Korai Zsolt

Generatív képalkotás

A Generatív képalkotás című tananyag elsősorban a VVVV nevű grafikus programozási környezetet mutatja be. Ez egy valós idejű, vizuális programozási nyelv, amit többnyire installációkhoz és különféle vetítésekhez használnak. Az egyes leckék során megismerkedhetsz a program kezelőfelületének használatával, alapvető matematikai- és üzenettípusok feldolgozásával, 2D és 3D objektumok megjelenítésével és azoknak a transzformációjával, kép- és videófeldolgozással, a különféle külső perifériák irányításával (például kamerák, Kinect, MIDI kontrollerek, Wacom tablet, Arduino és DMX), és az Android és IOS alapú OSC vezérléssel is. A program használatához nem szükséges előzetes programozói tudás, csak némi alapvető számítógépes és grafikai-videós ismeret.

Kulcsszavak: VVVV, grafikus programozás, 2D, 3D, Grafika, Video, Render, Mozgás detektálás, Generált kép, Real time

Budapesti Kommunikációs és Üzleti Főiskola

Typotex Kiadó
2014

Copyright: 🅭 2014-2019 Korai Zsolt, Budapesti Kommunikációs és Üzleti Főiskola

Creative Commons NonCommercial-NoDerivs 3.0 (CC BY-NC-ND 3.0)
A szerző nevének feltüntetése mellett nem kereskedelmi céllal szabadon másolható, terjeszthető, megjelentethető és előadható, de nem módosítható.

Lektorálta: Dunajcsik Péter

ISBN 978 963 279 219 4
Készült a Typotex Kiadó gondozásában
Felelős vezető: Votisky Zsuzsa

Készült a TÁMOP-4.1.2/A/1-11/1-2011-0010 számú, „Digitális és kollaboratív művészet” című projekt keretében.

Nemzeti Fejlesztési Ügynökség
www.ujszechenyiterv.gov.hu
06 40 638 638

MAGYARORSZÁG MEGÚJUL

A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

Tartalomjegyzék

Generatív képalkotás 2

1. A VVVV nevű program használata 5

2. A VVVV program kezelőfelülete, nyelvezete, menürendszere, az Inspektor és a segédfájlok használata 23

2.1. Első lépések, a program telepítése és hasznos tanácsok 23

2.2. A kezelőfelület 26

3. Az üzenettípusok, ki- és bemenetek ismertetése egyszerűbb példák segítségével 31

4. Komplexebb matematikai feladatok, fájlba írás és olvasás 37

5. A rendertípusok 47

6. A rétegek *(layer)* kezelése 53

7. Háromdimenziós megjelenítők és azok egyszerűbb editálása 61

8. Transzformációk és csoportok kezelése 68

9. A filterek és a beépülő modulok 78

10. Hangvezérlés 87

11. A kamera használata – kinyerhető adatok felhasználási lehetőségei 93

12. Külső kontrollerek használata 105

3. Az Arduino és elektronikai ismeretek 116

14. A gépek közti kommunikáció és különböző vezérlési lehetőségek 134

15. Kezelőfelületek készítése 150

Node list 162

1. A VVVV nevű program használata

A VVVV egy német fejlesztésű vizuális programozási környezet. 1998-ban Sebastian Oschatz és Max Wolf kezdte el fejleszteni a MESO (http://www.meso.net/vvvv) keretein belül. Nem sokkal később csatlakozott hozzájuk Sebastian Gregor, majd 2000-ben Joreg, aki a kezelőfelülettel kezdett el foglalkozni. Az első nyilvános verzió 2002-ben jelent meg, és azóta is Joreg és Sebastian Gregory vezeti a fejlesztést.

Hasonló kezelőfelülettel találkozhattál már videós, animációs, illetve 3D-s programoknál, mint például az Eyeon Fusion, a Blender, a 3DS Max, a Nuke, az UDK, a Unity stb. Ezeken kívül vannak még a VVVV-hez közelebbről hasonlító programok, mint például a Mac-en és Windows-on egyaránt működő Max-MSP, a multiplatform PureData (PD), a Quartz Composer (ami integrálva van az OSX-be), az Eyesweb és a csak Windows-on használható TouchDesigner. Ezek közül a Max-MSP (http://cycling74.com/products/max/) és a PD (http://puredata.info/) szinte testvéreknek számítanak, mivel mind a kettőt Miller Puckette hozta létre: annyi a különbség, hogy amíg a Max egy fizetős, jól használható kezelőfelülettel és extra szolgáltatásokkal rendelkező program, addig a PD egy nyílt forráskódú (Open Source) alkalmazás. Mindkettőt leginkább hangvezérlésre és -generálásra szokás használni, de nagyon sok videós eszközt is beleépítettek az idők során (a Max-MSP-be például nemrég még saját, node-alapú Shader editort is).

Valaha az Eyesweb (http://www.infomus.org/eyesweb_ita.php) volt a legalkalmasabb program arra, hogy emberalakokat és mozgást ismerjen fel és kövessen le, így elsősorban táncelőadásokon és performanszokban alkalmazták. Itt általában csak mozgásdetektálásra használták magát a programot, az ebből kinyert adatokat pedig hálózaton keresztül továbbították a megjelenítésért felelős gépnek, ami egy másik programozási környezetben generálta a látványt, de sajnos mára már nagyon háttérbe szorult ez a szoftver.

A VVVV-hez teljesítményben leginkább hasonlítható program a Touch Designer (http://www.derivative.ca/). A kezelőfelülete teljesen más, leginkább a 3D-s programokban megszokotthoz hasonlít. Ezenkívül jelentősen drágább, mint a VVVV: 600 és 2200 dollár között mozog az ára, míg a VVVV-t magáncélokra ingyen lehet használni. A Touch Designer nagy újítása az, hogy Cuda-t használ a kezelőfelület

legenerálásához és az effektek kiszámításához, így a teljesítménye többszöröse az eddigi programokénak, mivel ezeket a látványokat videokártyán sokkal gyorsabban lehet kiszámolni, mint a számítógép processzorán.

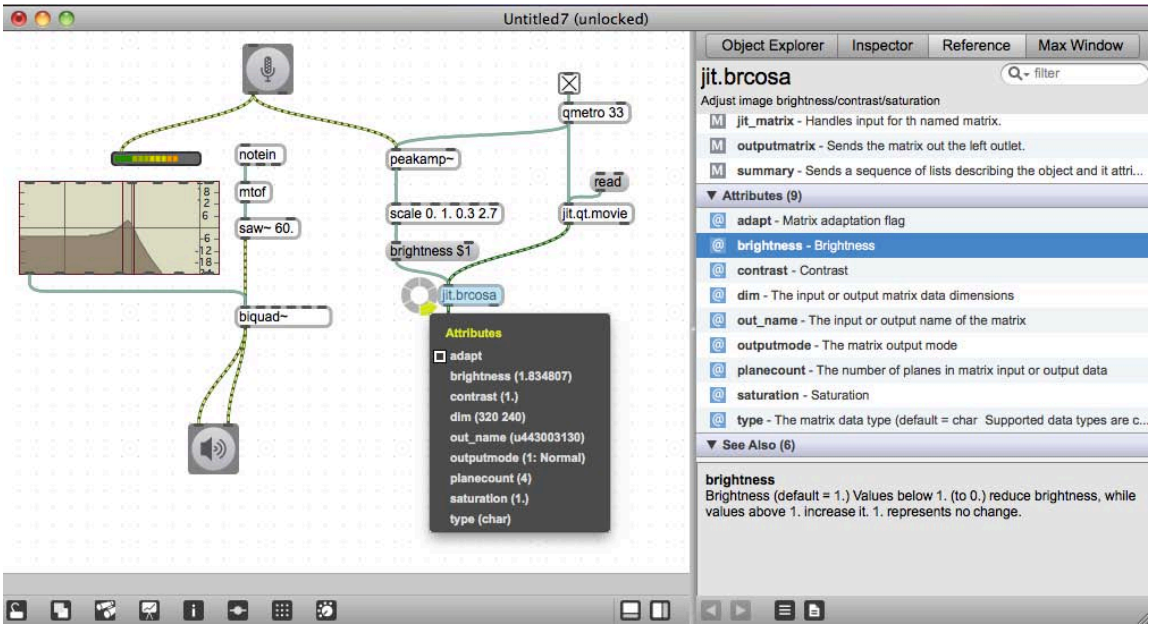
A VVVV-t elsősorban különféle művészeti projektekhez, video-installációkhoz, színházi előadásokhoz, kiállítások kezelőfelületeihez, épület- és tárgyvetítésekhez (mapping) és koncertek vetítéseihez szokták használni. Nem csak videók készítésére alkalmas, hanem akár fizikai eszközök: motorok, relék, LED-ek és egyéb lámpák vezérlésére is (Arduino segítségével). Színházi- és koncertvilágítási rendszerek kezelését is meg lehet oldani vele, mivel Artnet- és DMX-képes, és van beleépítve *Timeline,* amivel könnyen lehet zenéhez vagy időpontokhoz kötni a változtatási pontokat egy teljesen automatizált, akár hangra vagy más szenzorok adataira valós időben reagáló, interaktív rendszert hozva létre ezzel.

A VVVV-t külső eszközökkel is lehet vezérelni: ezek lehetnek külső vagy belső hangok (hangerő vagy frekvencia alapján történő vezérléssel), tabletek, joystickek vagy Wii-kontrollerek, Microsoft Kinect vagy Primesense 3D-s kamerák, egyszerű hagyományos kamerák (itt fényerő, szín- vagy mozgásalapú vezérléssel), MIDI kontrollerek, szenzorok és még sok más. Szerencsére a felhasználók folyamatosan töltik fel a program saját weboldalára az újabb és újabb kiegészítőket, amikkel különféle eszközöket tudunk használni. A feltöltések között azonban nemcsak eszközök találhatóak, hanem új effektek és különféle szimulációk is (például a *Bullet Physics,* ami egy játékoknál, illetve 3D-ben használt fizikai szimuláció, vagy akár a mostanában nagyon népszerű különféle *GPU Particle System*-ek). Ráadásul, ha elég magas szinten tudunk programozni, akár saját node-kat is írhatunk a VVVV-hez C#-ban, illetve a Cuda-hoz hasonlóan Gpu-n is kiszámoltathatjuk a különféle transzformációkat, vagy akár saját effekteket is írhatunk (szintén Gpu-ra, Hlsl Shaderben).

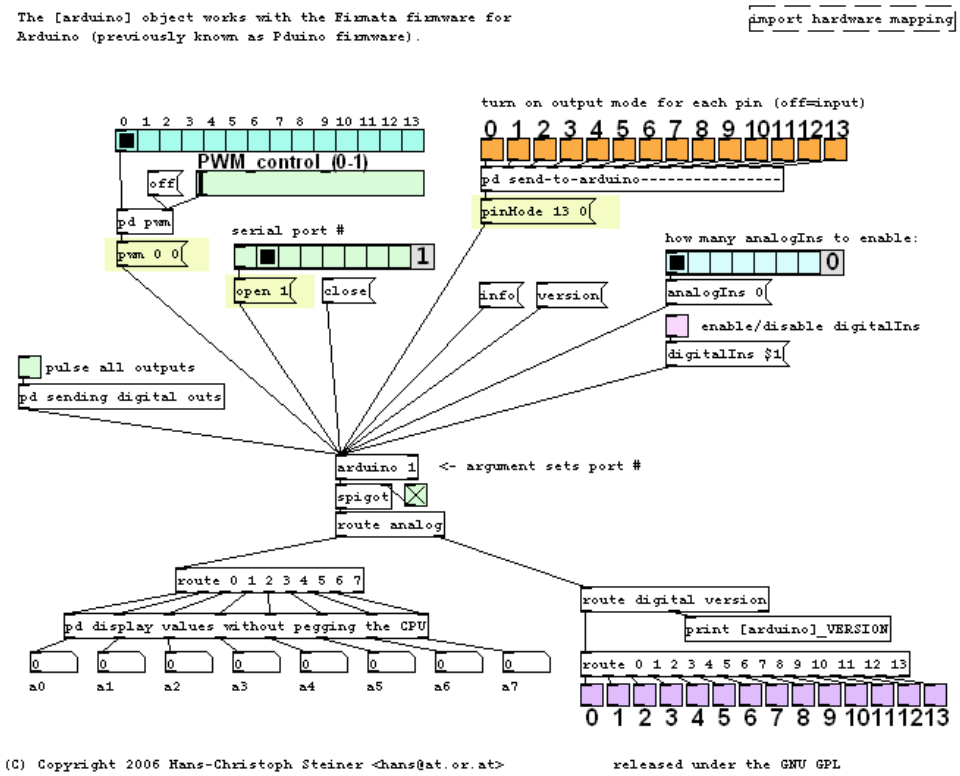
A VVVV nagy előnye a többi 3D-s programhoz képest, hogy nincsen renderidő, mivel mindent valós időben hajt végre, ezzel jelentősen meggyorsítva a folyamatot. A hátránya viszont, hogy a render minőségét nem szabad összehasonlítani egy direkt erre specializált szoftverekével, mivel számos olyan funkció (árnyékolás, világítás és egyéb effektek), amiket más programoknál megszokhattunk, itt még nem működnek a legjobb minőségben. Ezek javításán viszont folyamatosan dolgoznak a felhasználók, és próbáljak bővíteni a lehetséges effektek körét.

A mellékelt videóban az elmúlt években VVVV-ben készült legjelentősebb videómunkákból láthatsz egy összeállítást.

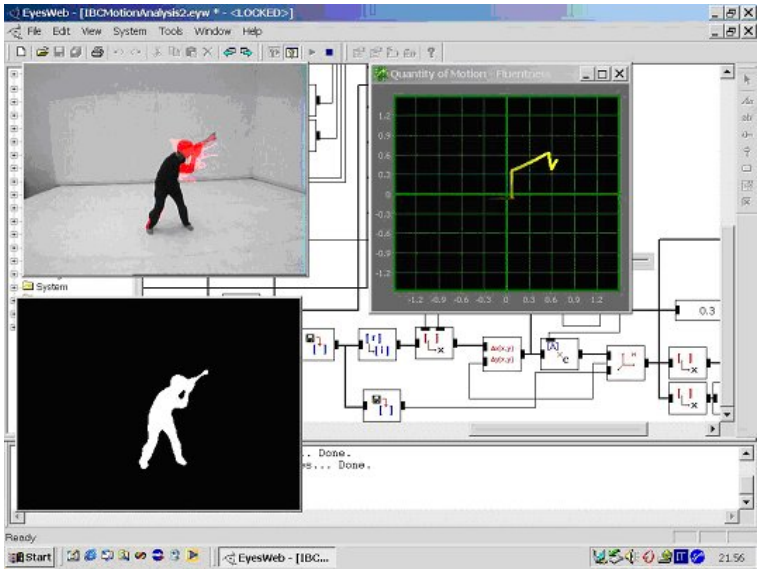
Jó szórakozást a videókhoz, és jó tanulást a következőkben!



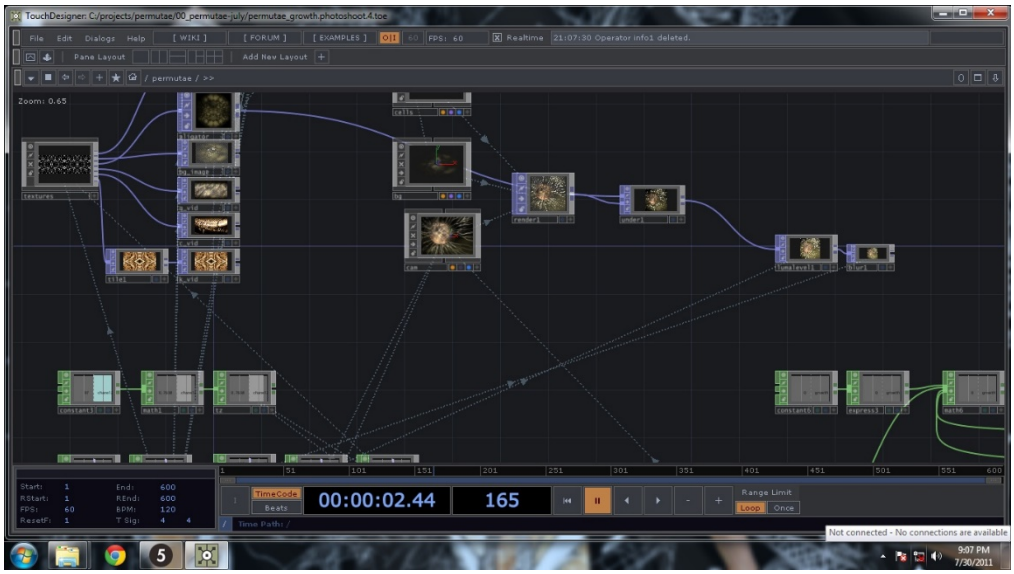
1.1. ábra: max



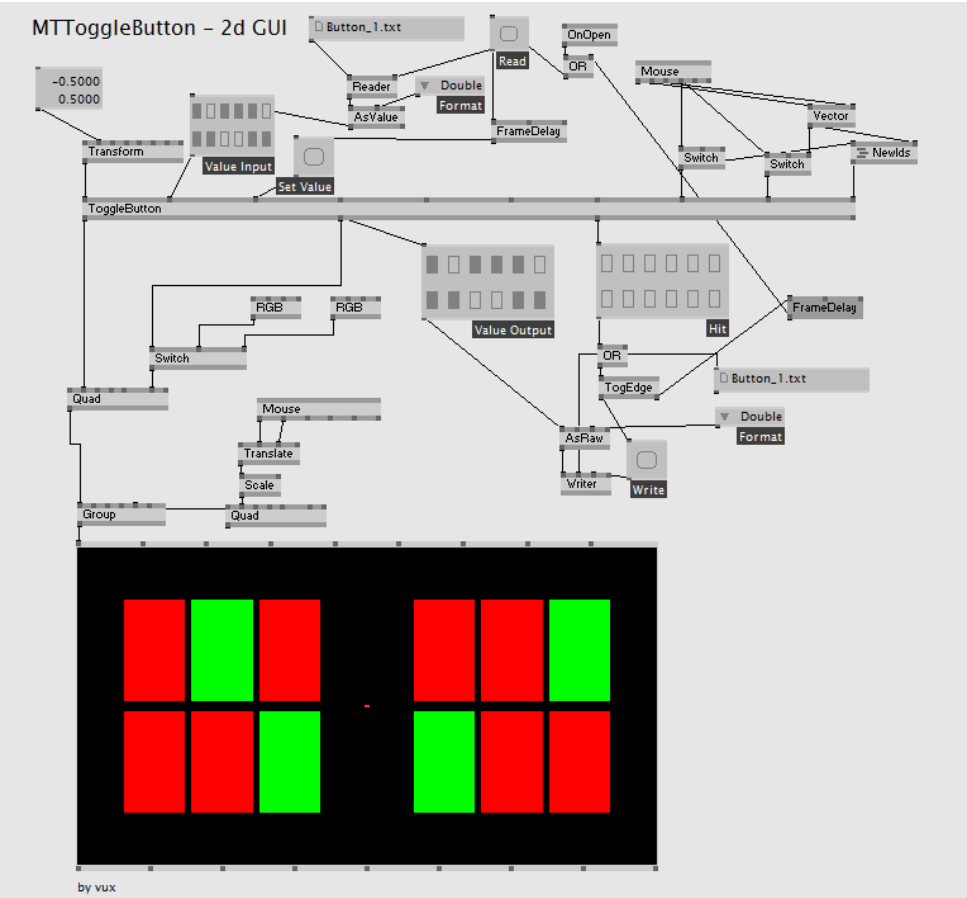
1.2. ábra: PD



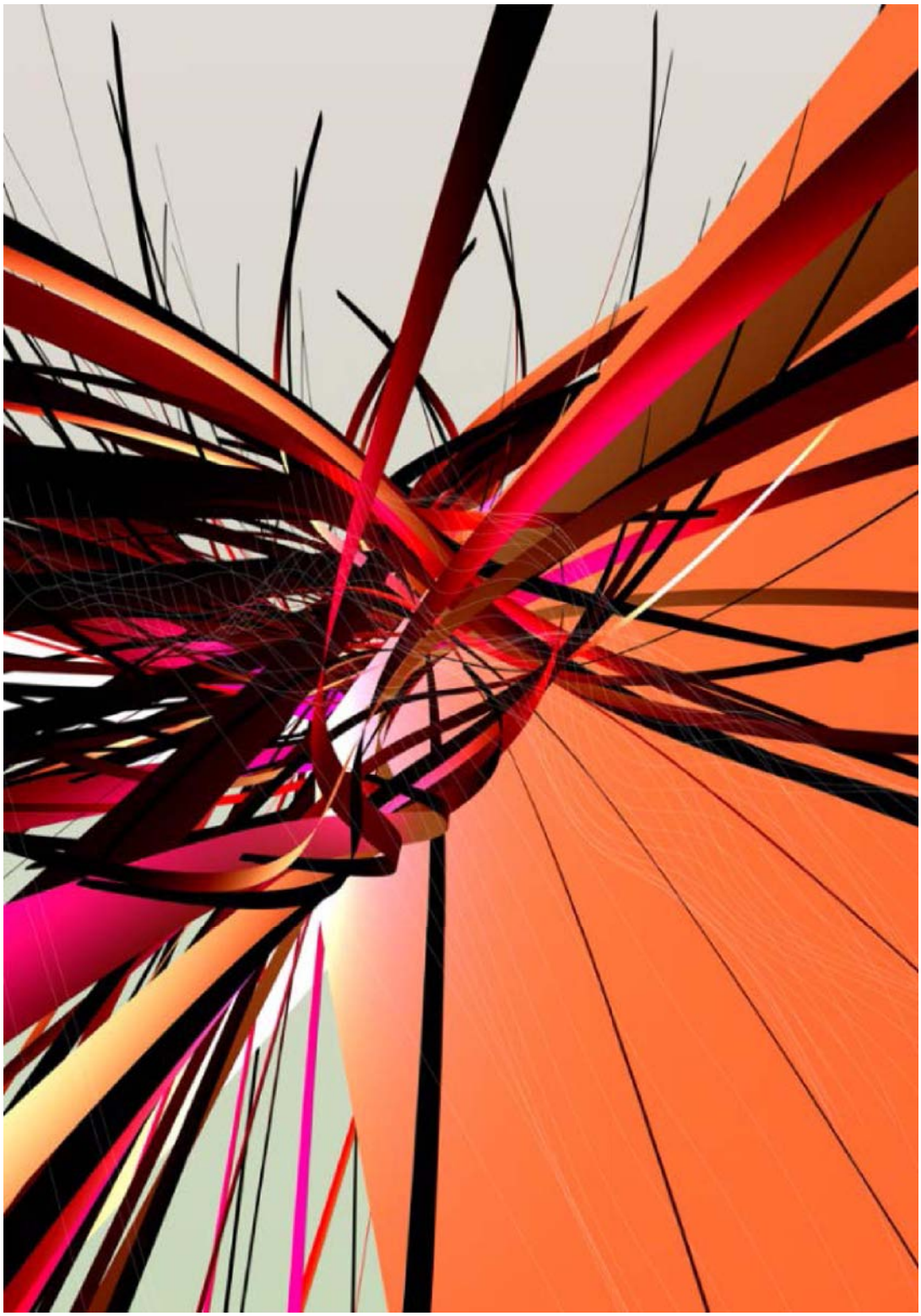
1.3. ábra: Eyesweb



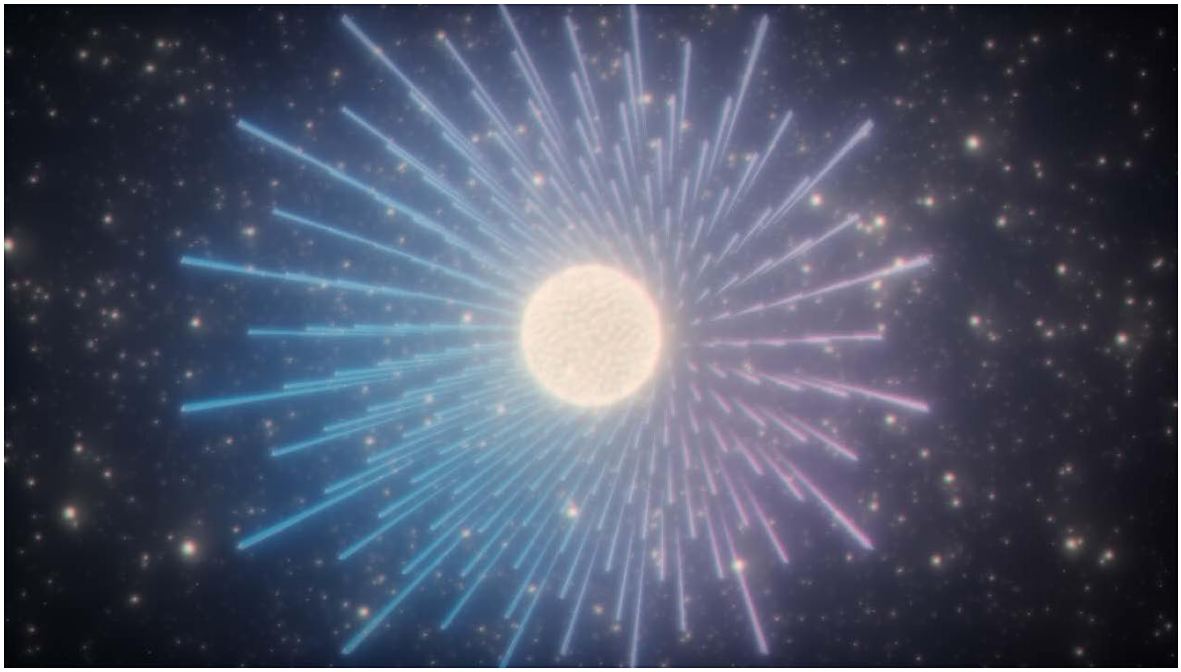
1.4. ábra: Touchdesigner


MTToggleButton – 2d GUI
Button_1.txt
Read
OnOpen
OR
Mouse
-0.5000
0.5000
Reader
Double
Format
AsValue
FrameDelay
Vector
Transform
Value Input
Set Value
Switch
Switch
NewIds
ToggleButton
RGB
RGB
Value Output
Hit
FrameDelay
Switch
OR
TogEdge
Button_1.txt
Quad
Mouse
Double
Format
AsRaw
Translate
Scale
Writer
Write
Group
Quad
by vux

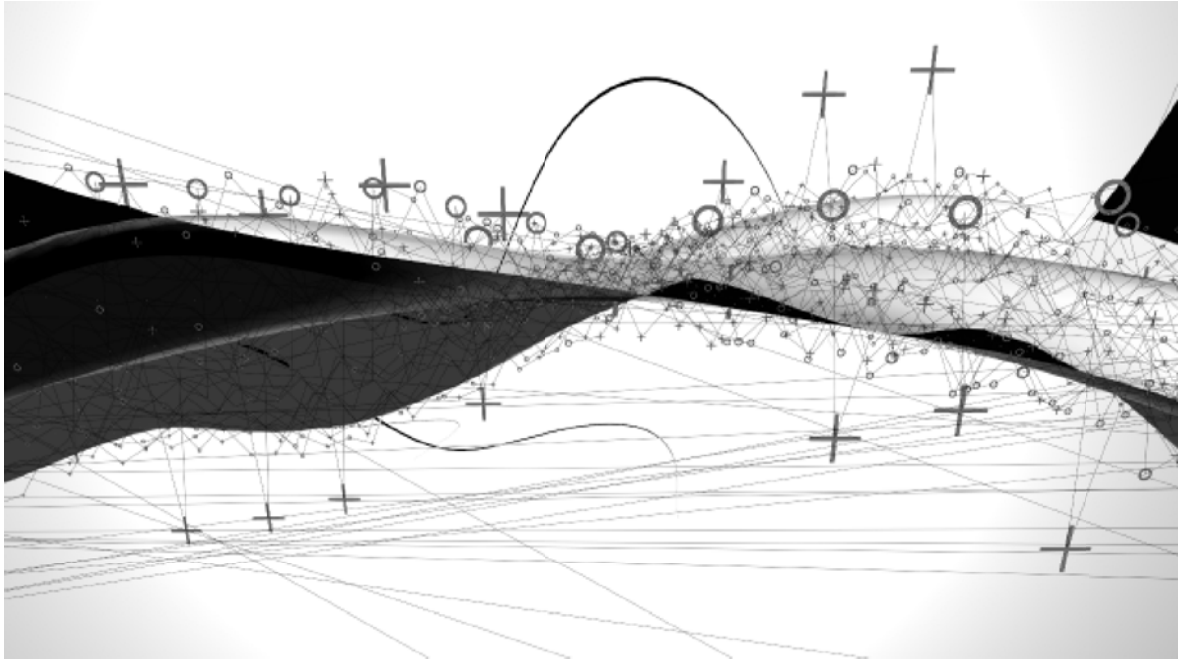
1.5. ábra: VVVV



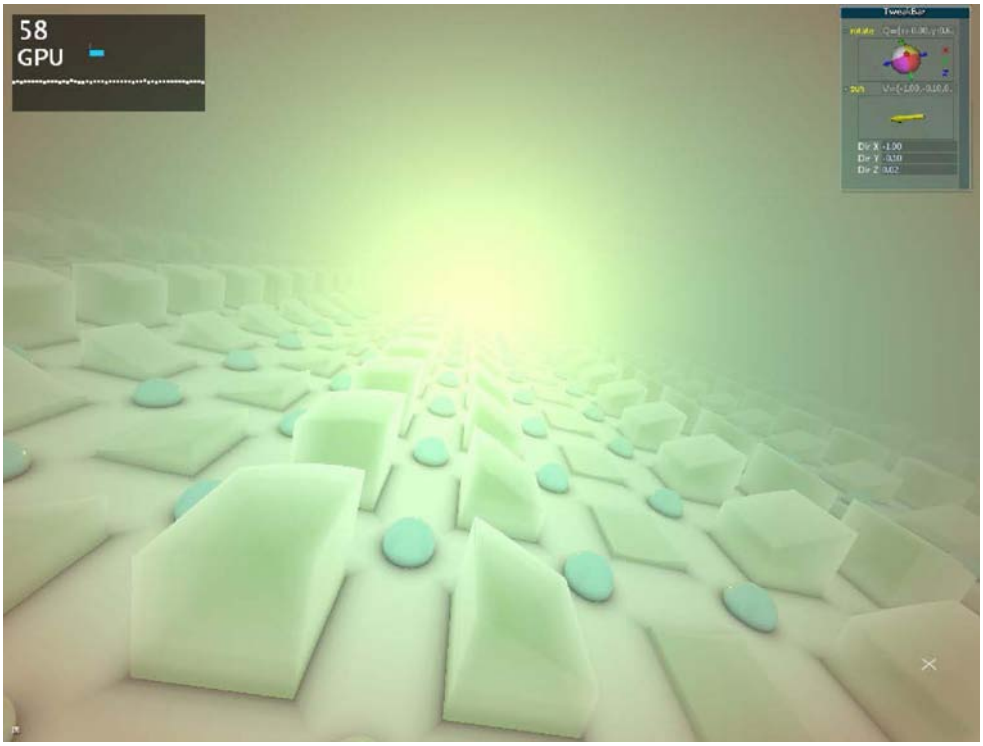
1.6. ábra: Cybear_screenshot



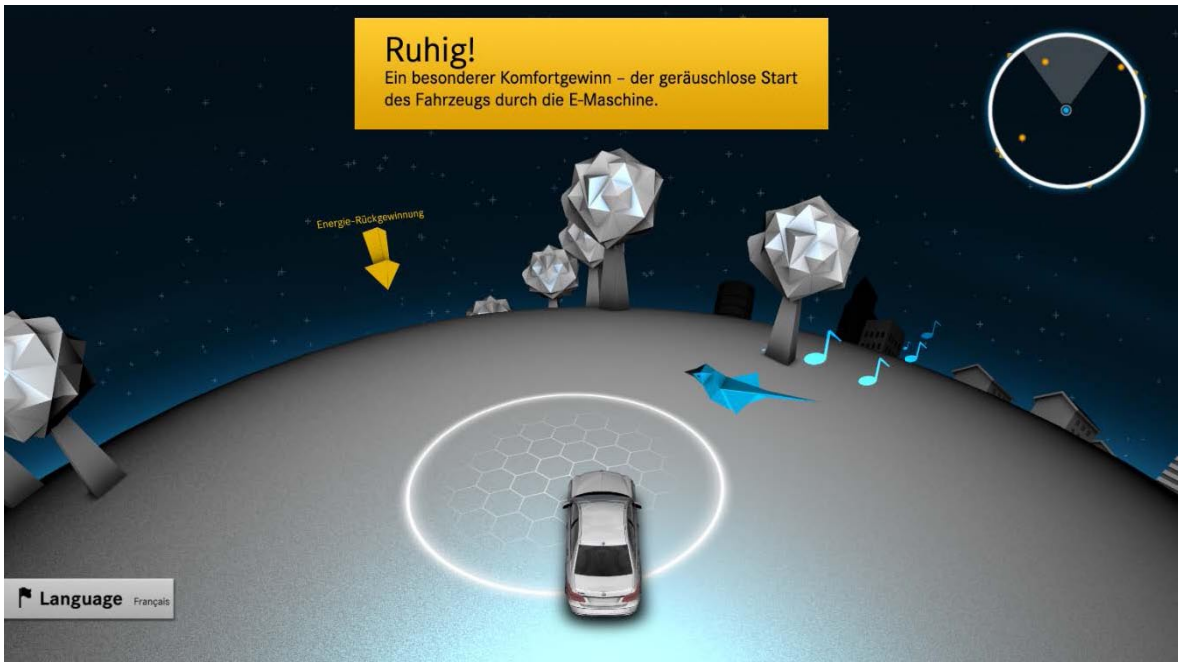
1.7. ábra: Lightmare_screens



1.8. ábra: Lanvideosource



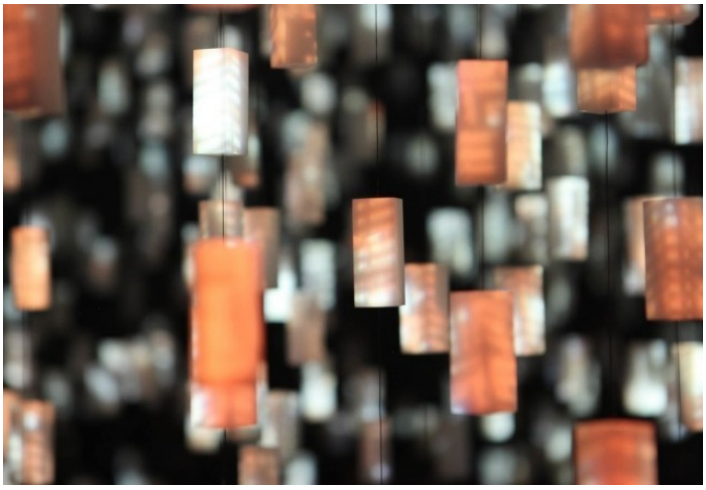
1.9. ábra: Microdee_screens



1.10. ábra: u7angel_still

1.11. ábra: Unc_sreenshot13

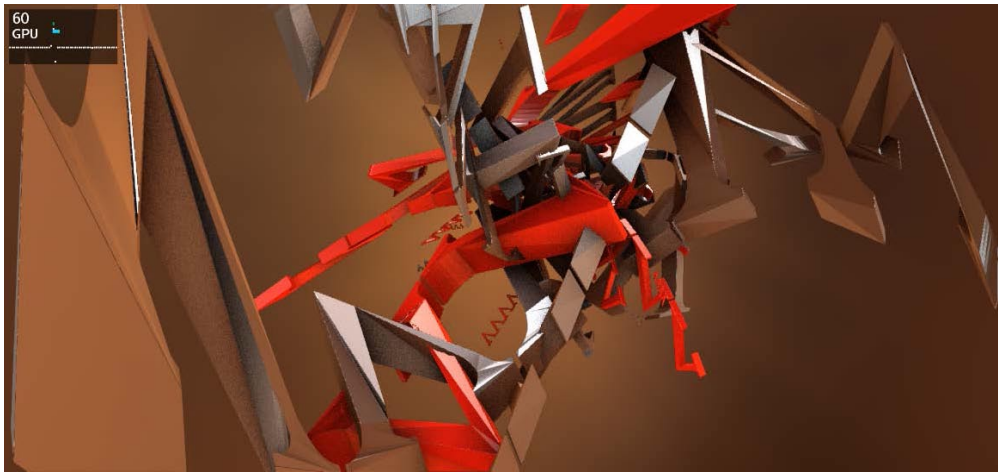
1.12. ábra: Mr_prudence_Parhelia



1.13. ábra: Elliotwoods_Asse

1.14. ábra: Patxi7 Ehime Daruma

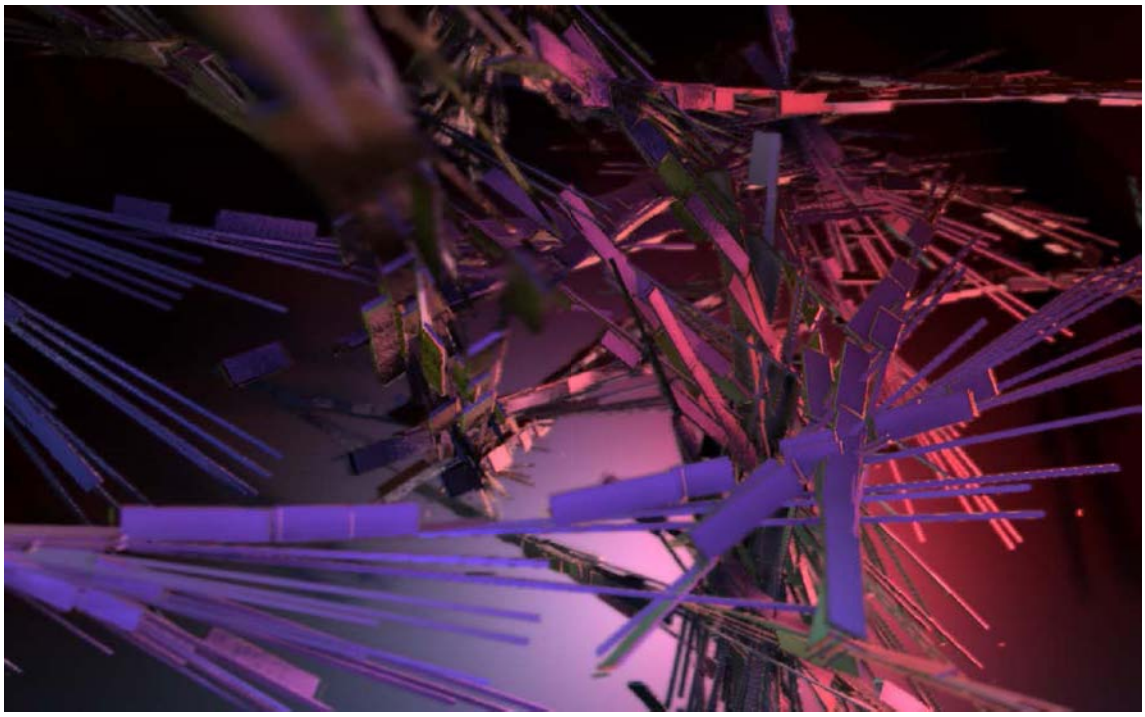
1.15. ábra: Unc_screenshot 13



1.16. ábra: Unc_screenshot 13

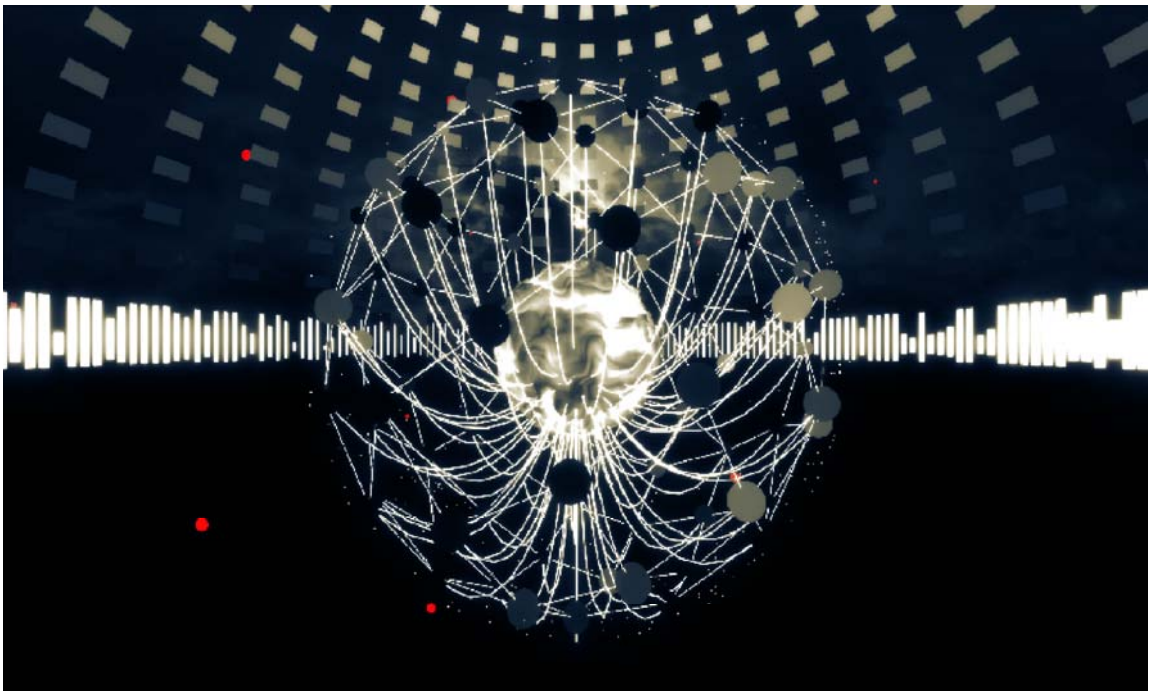
1.17. ábra: Lecloneur_screens

1.18. ábra: Sebl



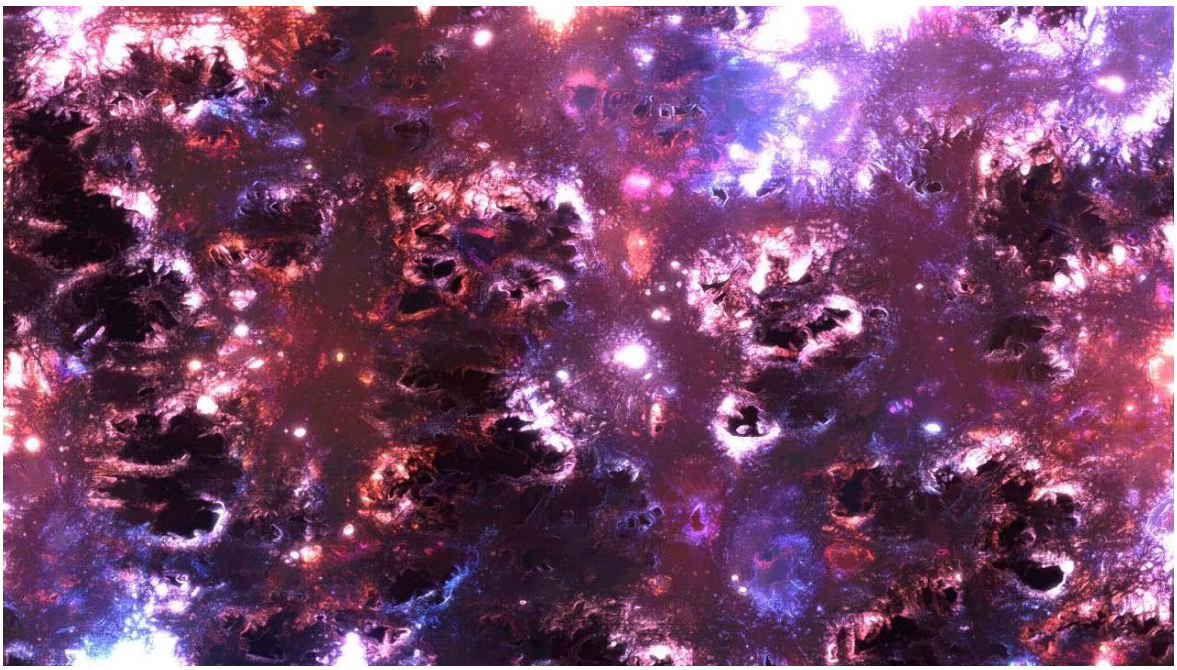
1.19. ábra: Lecloneur_screens

1.20. ábra: Lorenz_2



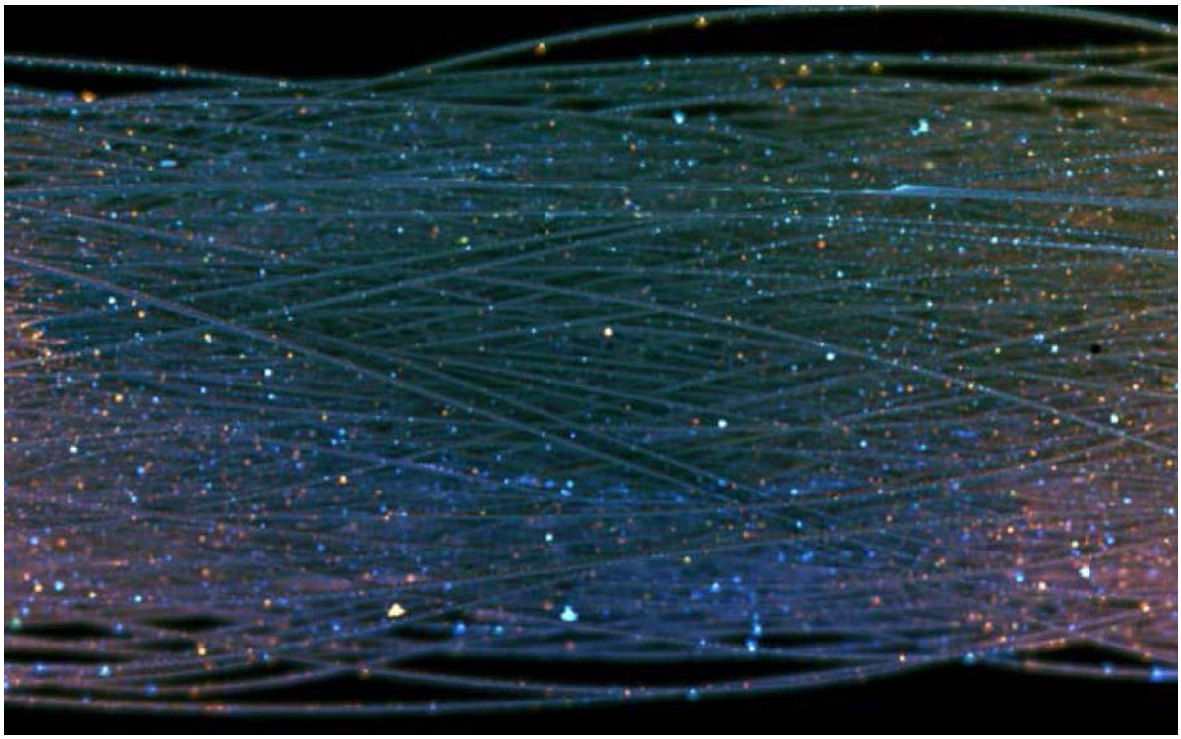
1.21. ábra: Brozzmical_screen

1.22. ábra: Lechloneur_screens



1.23. ábra: Lechloneur_screens

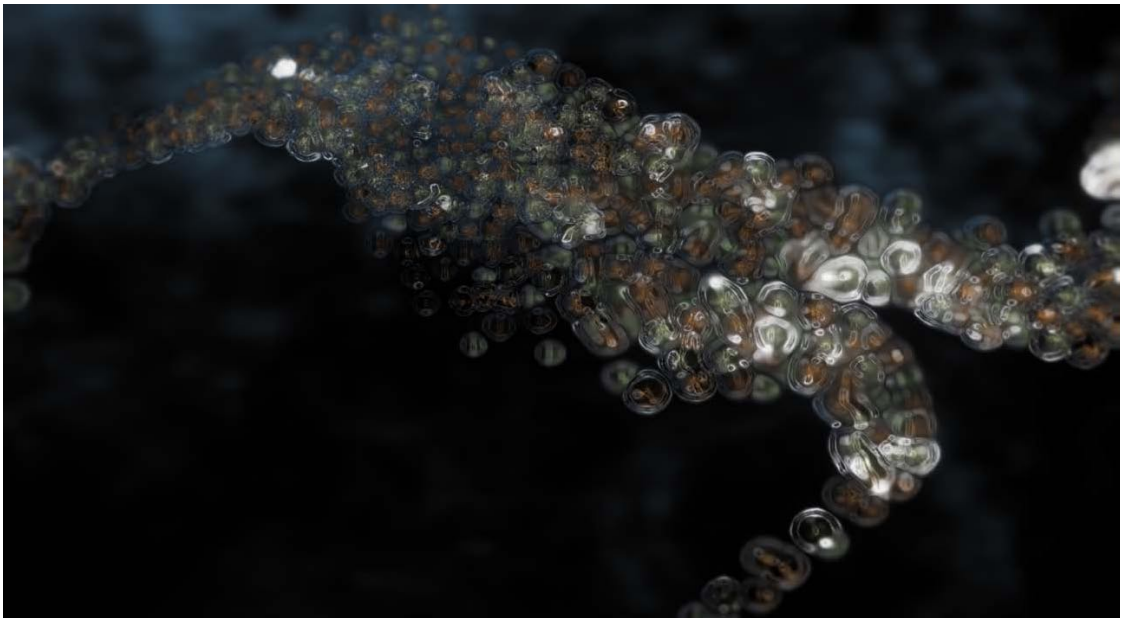
1.24. ábra: Vux_screenshot



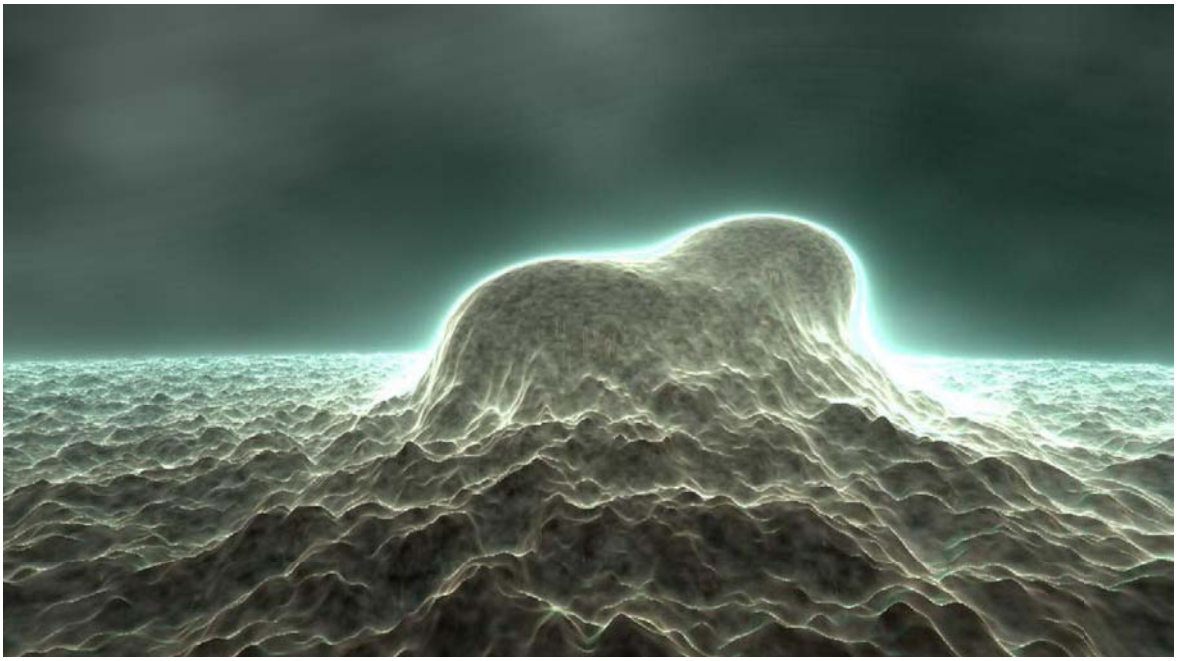
1.25. ábra: Vux_screenshot

1.26. ábra: WirmachenBunt, Mutabor Design G

1.27. ábra: Stain_MIMPI

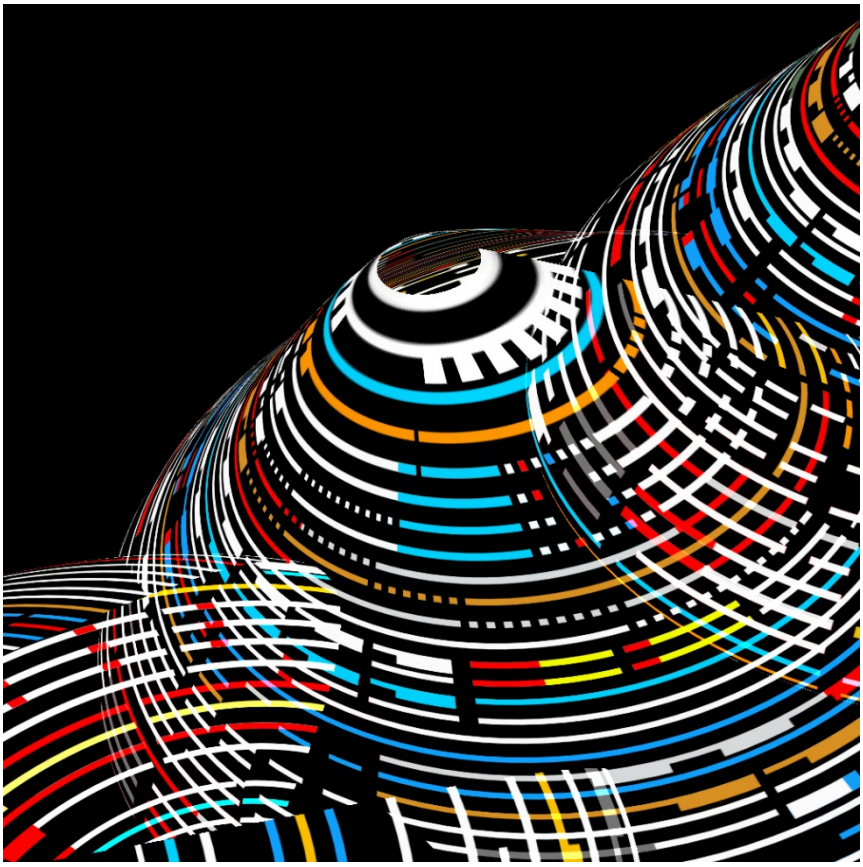


1.28. ábra: Eletromeier_screens



1.29. ábra: Vux_sreenshot

1.30. ábra: Dottore_screensh



1.31. ábra: mr prudence_Isograms

2. A VVVV program kezelőfelülete, nyelvezete, menürendszere, az Inspektor és a segédfájlok használata

2.1. Első lépések, a program telepítése és hasznos tanácsok

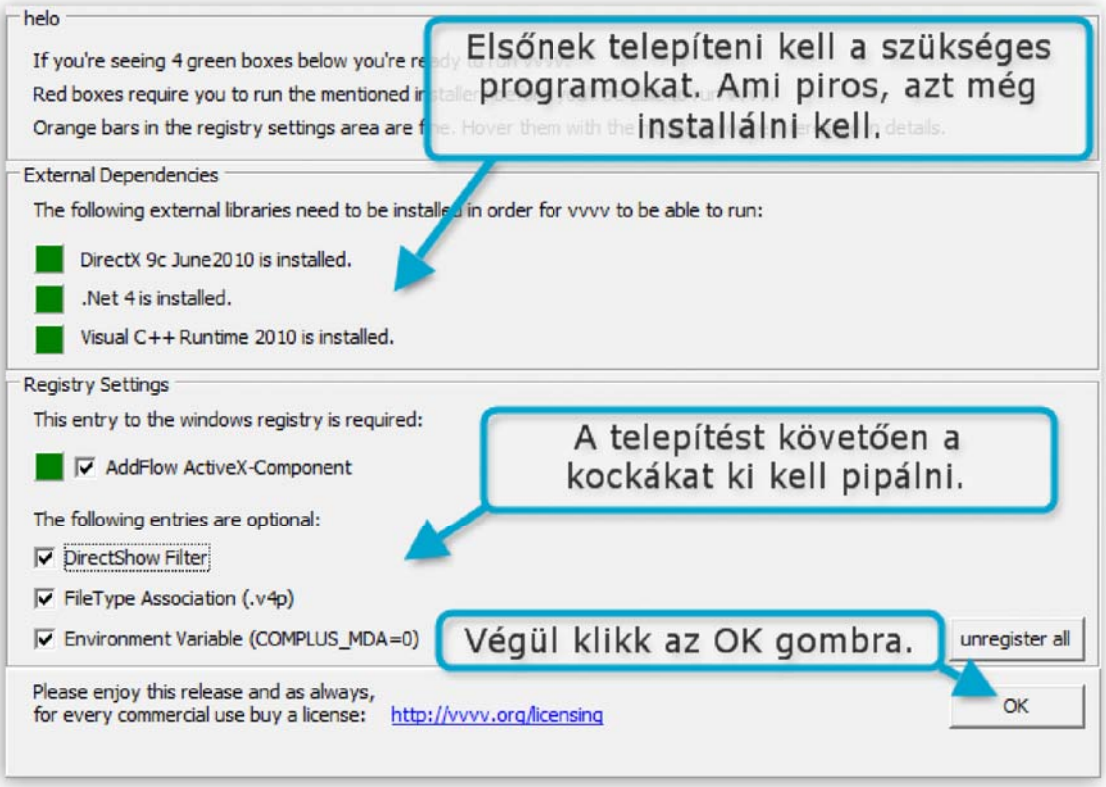
Ajánlott mindig a legfrissebb verziószámú VVVV-t használni, mert a fejlesztők folyamatosan javítják a hibákat, és újabb és újabb kiegészítőket építenek be a rendszerbe. Amennyiben mégis zavaró programhibákkal találkozol, mindig térj vissza az eggyel korábbi verzióra.

Első lépésként töltsd le a legújabb programot a http://vvvv.org/downloads oldalról. Fontos, hogy az *addon pack*-ot is letöltsd mellé, ebben ugyanis nagyon hasznos bővítményeket találsz, melyekre számos *patch* hivatkozik.

2.1. ábra: VVVV letöltése

A letöltést követően ajánlott nem a rendszer saját tömörítőszoftverét használni, hanem a Winrar, Winzip vagy 7zip nevű programok közül valamelyiket, ugyanis a Windows saját tömörítője sokszor nem bont ki bizonyos fájlokat a csomagból. Az addon pack zip tartalmát másold az előzőleg kicsomagolt VVVV könyvtár gyökerébe.

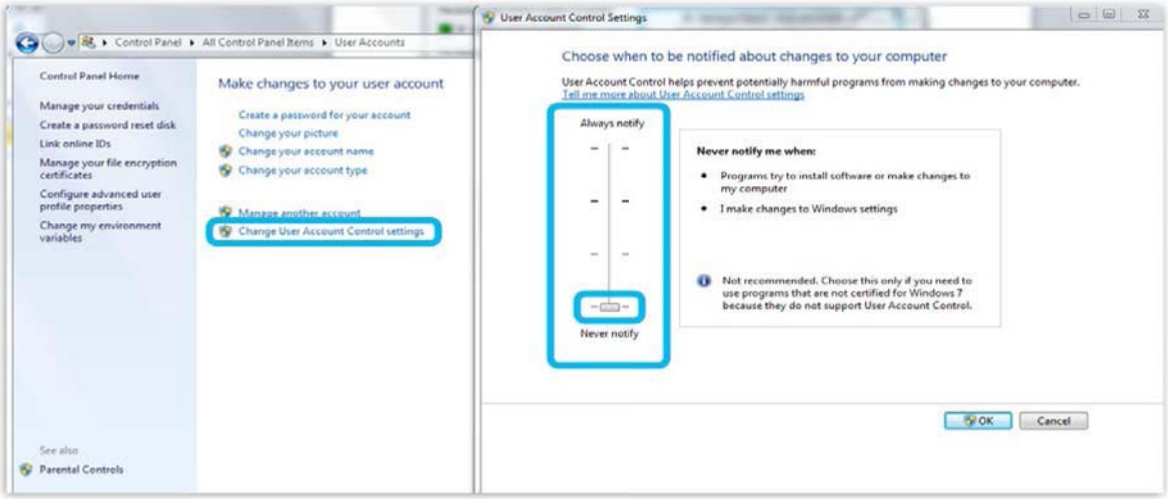
Miután mindez sikerült, először a VVVV könyvtárban található **crack.exe** nevű fájlt indítsd el. Ez azért szükséges, hogy a program alapértelmezettként kezelje a ***.v4p** kiterjesztést, emellett pedig azt is megmutatja, ha más, a VVVV futtatásához szükséges programok esetleg még nincsenek telepítve.



2.2. ábra: A crack.exe

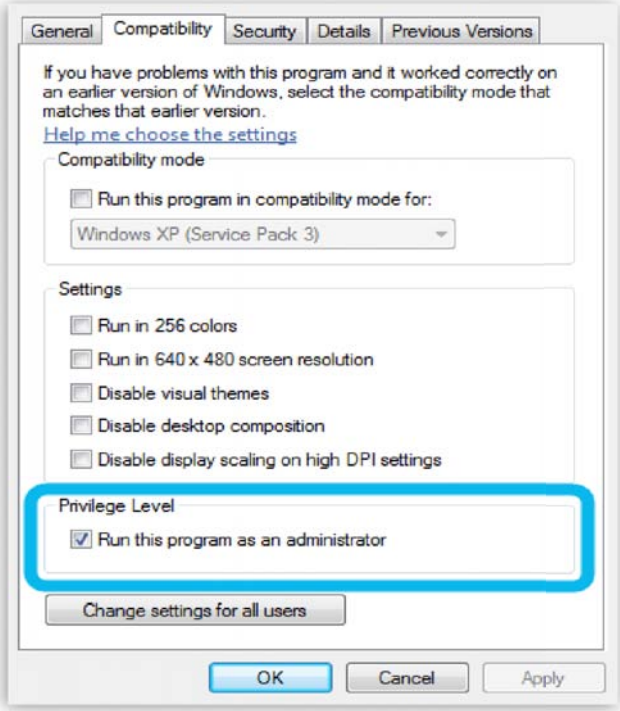
Ezek után már csak egy fontos lépés szükséges a program futtatásához: a rendszergazda-üzemmód beállítása. Ha ugyanis ez nincsen engedélyezve, a program beindul ugyan, de nem lehet használni. Ez a beállítás Windows Vista megjelenése óta létezik, és kétféleképpen lehet engedélyezni:

- A teljes operációs rendszer rendszergazda-üzemmódra való átállítása. Ehhez a *Vezérlőpult*ban a *Felhasználói beállítások*on belül válaszd ki a *Ne értesítsen soha* funkciót, majd indítsd újra a számítógépet. Ezt követően már használható a VVVV.



2.3. ábra: Rendszergazda-üzemmód

- Csak a VVVV programra vonatkozó beállítás. Ehhez klikkelj jobb gombbal a **VVVV.exe** fájlra, majd a *Beállítások* közül a *Kompatibilitás* menüpontban válaszd ki a *Program indítása rendszergazdaként* beállítást. Fontos, hogy ha ezt a lehetőséget használod, úgy minden egyes újabb verzió installálása után meg kell ismételned a beállítást.

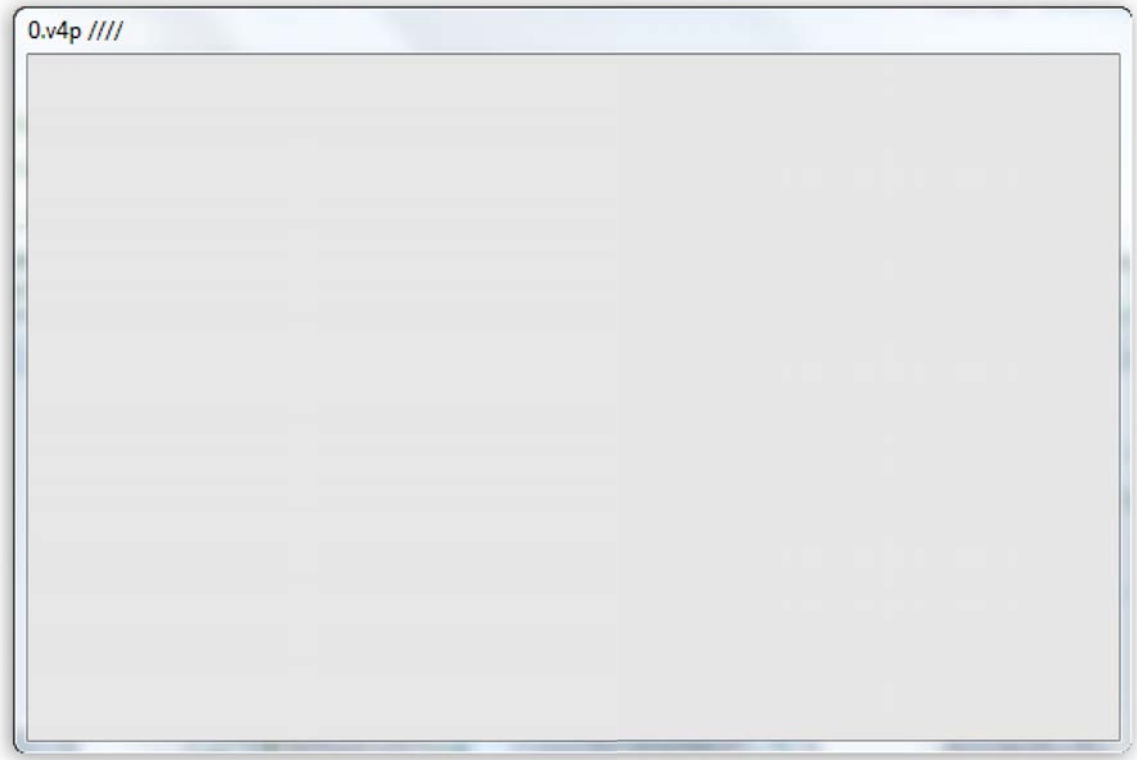


2.4. ábra: A VVVV.exe rendszergazda-üzemmódban való futtatása

Ha mindezeket sikerült beállítani, indítsd el **VVVV.exe** fájlt.

2.2. A kezelőfelület

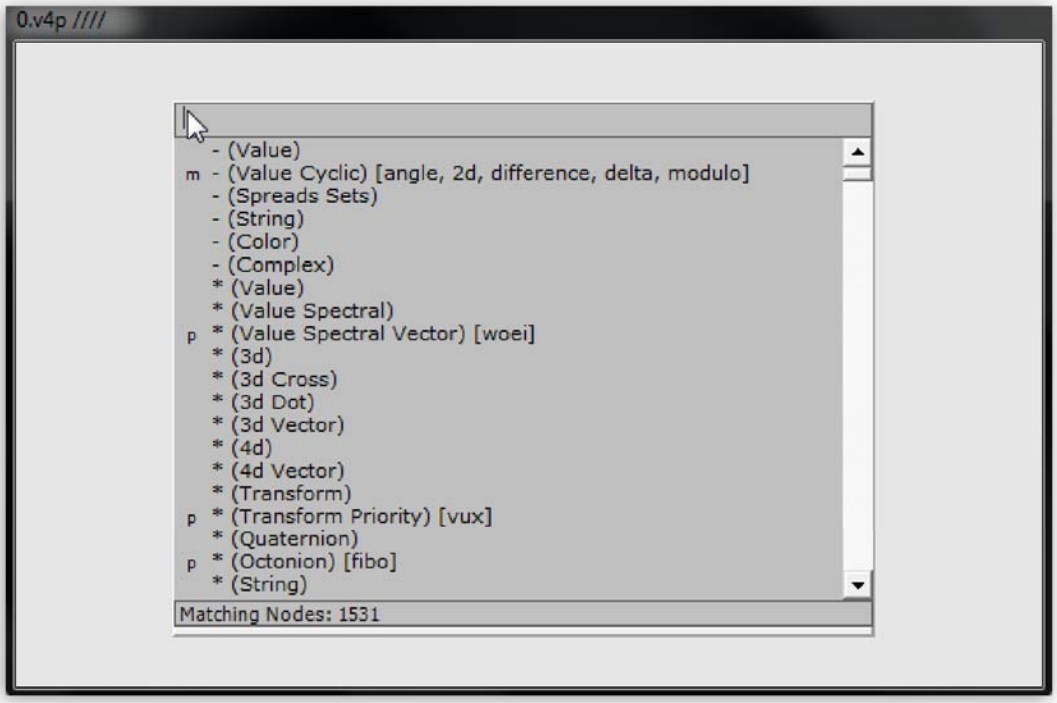
A program indítása után nem egy megszokott kezelőfelületet látsz majd, mivel itt nincsenek menük és gombsorok. Bár először csupán egy szürke ablaknak tűnik, ez ne rettentsen el! A VVVV nagyon is felhasználóbarát, mindössze arról van szó, hogy a menüket klikkeléssel lehet előhozni a felületről.



2.5. ábra: A VVVV indítás után

A kezelőfelületből háromféle menüt tudsz előhívni, más-más funkciókkal:

Az első a választható eszközök, *node*-ok listája. Ezt két balklikkel tudod előhívni. Itt van keresési lehetőség, vagyis ha a két balklikk után elkezdesz gépelni, a menü automatikusan csak az adott karakterekkel kezdődő találatokat mutatja meg. Ha viszont a két balklikk után egy jobbklikket nyomsz, akkor a menü típus szerint csoportosítva mutatja meg a *node*-okat. Ez akkor hasznos, amikor nem jut eszedbe a keresett eszköz pontos neve, azt viszont tudod, hogy melyik csoportba tartozik.



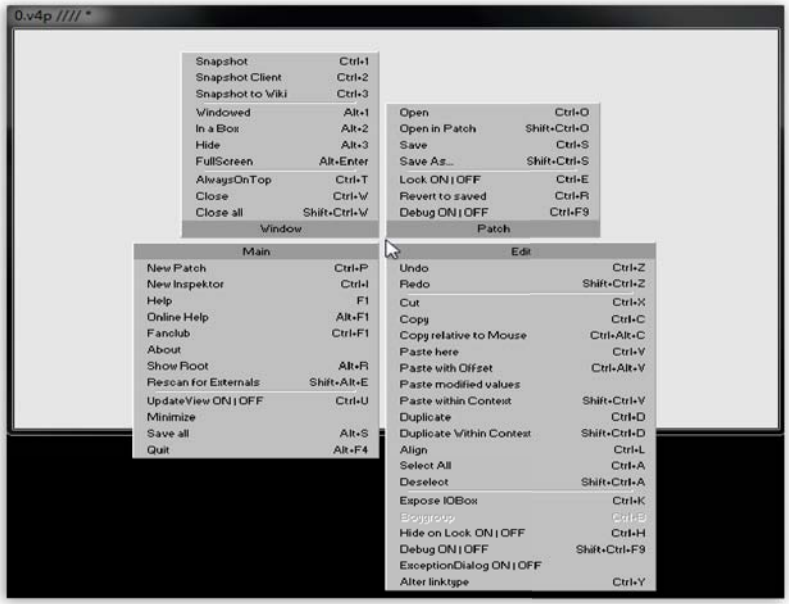
2.6. ábra: Két balklikk

2.7. ábra: Két balklikk + jobbklikk

- A második menüben a különböző üzenettípusok közül választhatsz. Ezt két jobbklikkel tudod előhozni.

2.8. ábra: Két jobbklikk

- A főmenü az egér középső gombjára klikkelve, vagy a Space és a jobbklikk együttes megnyomásával hívható elő. Ebben a legtöbb programban megszokott funkciókat mind megtalálod, némileg szokatlan elrendezésben. Az egyes funkciók mellett itt szerepel a hozzájuk tartozó billentyűkombinációk is, így némi gyakorlás után ezt a menüt egyre ritkábban fogod használni.



2.9. ábra: Főmenü

Végezetül ejtsünk szót néhány alapfogalomról. A VVVV-ben készített programokat *patch*-nek hívjuk, a programokon belül pedig a kis dobozokat, amiket összekötünk, *node*-nak. Mentéskor a *patch*-ek **.v4p* kiterjesztést kapnak. Ezek tulajdonképpen szöveges dokumentumok, melyek kissé hasonlítanak az **.xml* dokumentumokhoz, csak ezek a *node*-ok pozícióit és paramétereit, valamint a köztük lévő összeköttetéseket rögzítik. Minden *node*-nak vannak ki- és bemenetei, melyeket kis szürke négyzetek jelölnek. Ezt úgy kell elképzelni, mintha egy szövegalapú programnyelvben egy függvény lenne bedobozolva, és annak a változói és kimeneti paraméterei lennének kivezetve; itt viszont nem kell megírni magát a függvényt, csak hivatkozni rá.

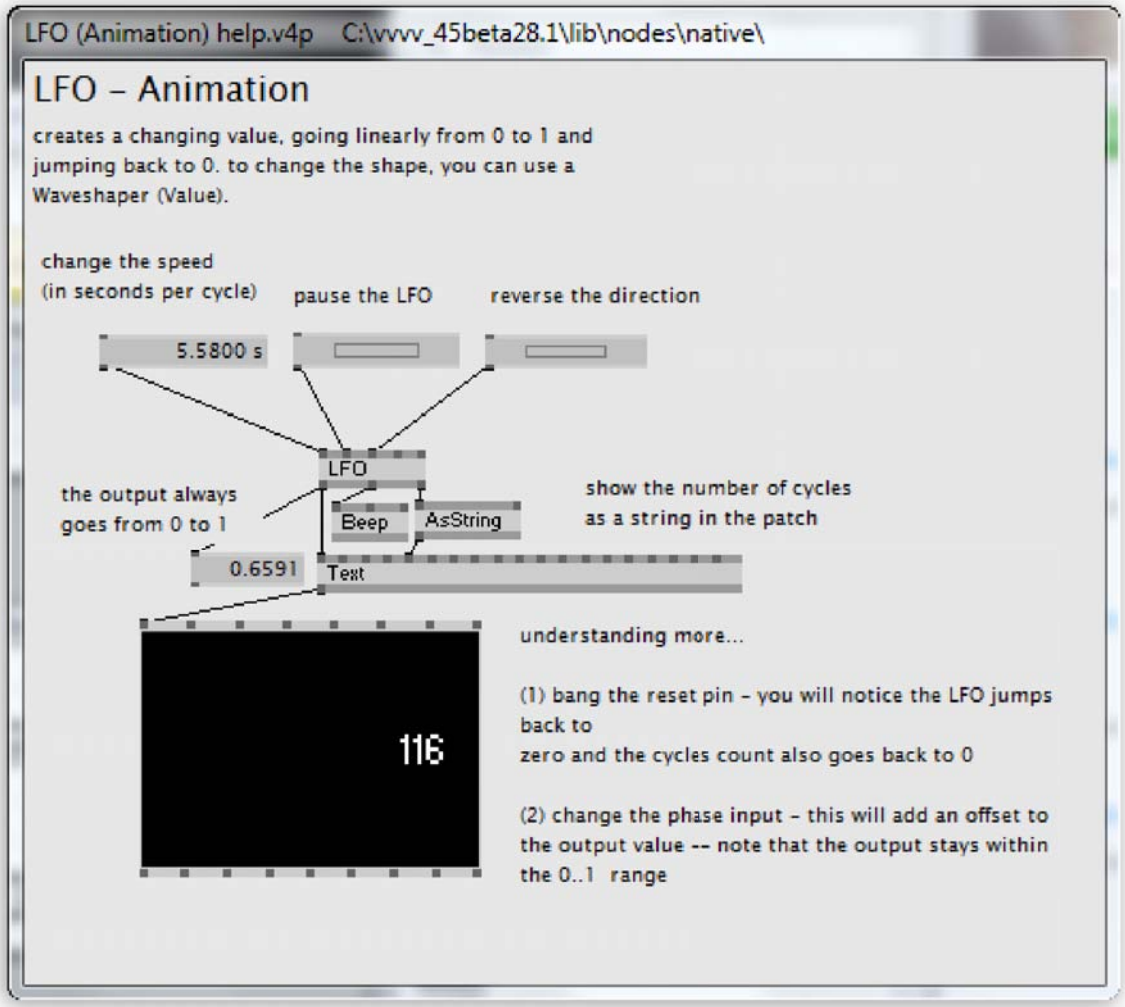
Most nyomj két balklikket, majd válaszd ki az *lfo* funkciót, és nyomj egy entert! A doboz tetején látni fogsz öt bemenetet, ha pedig fölé viszed az egeret, a program kiírja az egyes bemenetek neveit és a hozzájuk tartozó változókat: *Period: 1.00s, Pause: 0, Reverse: 0, Reset: 0, Phase: 0.00.* A doboz alján lévő kimeneteken pedig következőket látod: *Output: 0.xxxx~, Change: 0* (ez utóbbi másodpercenként 1-re vált, mivel 1 másodperces ciklusban van), a *Cycles* kimenetnél pedig valamilyen számot látsz.

A *node*-ok paramétereit és esetleges rejtett beállításait a *Ctrl* és az *i* billentyűk egyidejű lenyomásával tudod megtekinteni. Ezzel felugrik az *Inspektor* nevű ablak. Ha az *lfo*-ra klikkelsz, pontosan láthatod a be- és kimeneteket és paramétereiket. Az Inspektor további használatáról a későbbi leckékben lesz szó bővebben.

2.10. ábra: Lfo + Inspektor

Ha valamelyik *node*-ról többet szeretnél megtudni, például hogy hogyan működik és mire lehet használni, akkor az adott *node*-ra klikkelve és az F1 gombot megnyomva felugrik egy új ablakban a segédfájl *(help)*, ami további magyarázatokkal szolgál. (Az *lfo* esetében ez az *LFO (Animation) help* nevet viseli.) Szerencsére a VVVV-nek nagyon jól használható segédfájljai vannak, melyekből rengeteget lehet tanulni.

Amennyiben valamelyik olyan *patch*-et akarod használni, amit a segédfájlban találtál, de nem akarod újra elkészíteni, akkor kijelölés után a más programokból jól ismert *Ctrl + c* (másolás), majd *Ctrl + v* (beillesztés) billentyűkombinációkkal át tudod emelni a saját programodba.



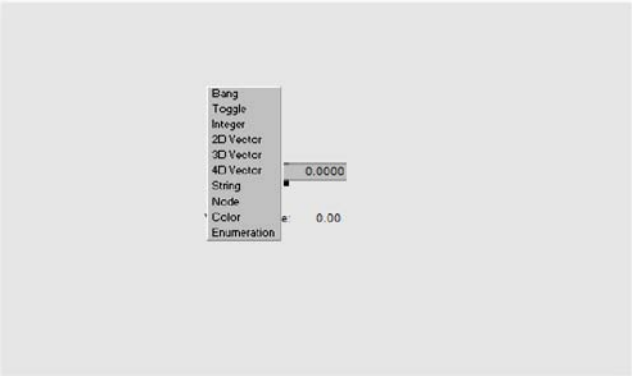
2.11. ábra: Lfo segédfájl

Ha a *patch*-en belül szeretnél mozogni, ezt vagy az egérrel scrollozva teheted meg, vagy ha az egér jobb gombját hosszan nyomva tartod egy szabad szürke felületen, majd azt elmozdítva mozgatod meg az egész *patch* tartalmát.

Ezzel nagyjából el is sajátítottad a VVVV-kezelőfelület használatának alapjait. A következő leckében az üzenetek típusait, a ki- és bemenetek használatát, és néhány alappéldát ismerhetsz majd meg, ami elősegíti a VVVV logikájának további megértését

3. Az üzenettípusok, ki- és bemenetek ismertetése egyszerűbb példák segítségével

A VVVV-ben az üzenetdobozokat *IOBox*-nak hívják. Ezekből többféle létezik, attól függően, hogy számra *(Value),* szövegre *(String),* színre *(Color),* adatra *(Node)* vagy legördülő menüre *(Enumeration)* vonatkoznak. Mindegyik *IOBox* paramétereit, vagyis hogy mennyi üzenetet tartalmazzon, milyen betűtípust használjon és még sok mást az Inspektorban pontosan be lehet állítani – ezt a későbbiekben fejtjük ki bővebben.

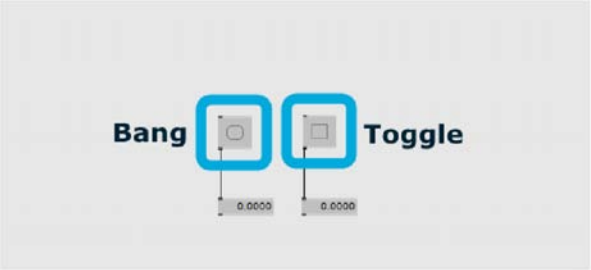


3.1. ábra: Az üzenet menü

Először vegyük a leggyakrabban használt, számalapú üzenettípusokat! A VVVV-ben számos előre elkészített variáció közül tudsz választani, ha két jobbklikket nyomsz a szürke háttér valamely szabad felületén. Ekkor felugrik az üzenetmenü, ahol a felsorolásban megtalálod a *Bang, Toggle, Integer, 2D Vector, 3D Vector* és *4D Vector* nevű funkciókat, amik mind számalapú üzenetek. A *Bang* és a *Toggle* hasonló funkcióval bírnak, mindkettő egyfajta kapcsoló, ami alapesetben 0 jelet küld ki.

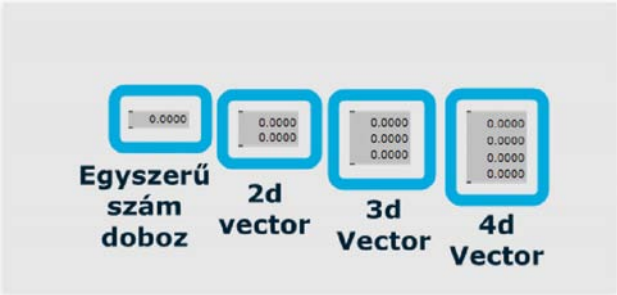
A *Bang* megnyomáskor egy pillanatra felvillan, és kiküld egy 1-es üzenetet, majd visszatér 0-ra. Ezt például léptetéshez, törléshez és hasonlókhoz lehet használni.
A *Toggle* megnyomásakor a kimenet 0-ról 1-re vált át és úgy is marad, újabb megnyomásra pedig visszatér az eredeti állapotába, vagyis 0-ra.

A kettőt úgy tudod a legegyszerűbben megkülönböztetni egymástól, hogy a *Bang*-nek lekerekített a széle, a *Toggle*-nek pedig szögletes. Érdemes észben tartani, hogy a VVVV-ben az üzenetek megváltoztatására elsősorban a jobbklikket kell használni és nem a megszokott balt. (Kétszeres balklikkel az üzenetdoboz billentyűs beviteli módra vált át.)
Most próbáld ki a gombok működését! A jobb egérgombbal klikkelj kétszer a háttérre, majd hozz lére egy *Toggle*-t és egy *Bang*-et. Mindkettő alá (szintén jobbklikkel) hozz létre két egyszerű számdobozt, vagyis csak klikkelj, és húzd el az egeret. Ahhoz, hogy lásd is a változást, össze kell kötnöd a kapcsolókat a számokkal. Ezt úgy tudod megtenni, hogy balklikkel a *Bang* és a *Toggle* alján látható kis sötétszürke kockákat, vagyis a kimeneteket összekötöd a számdoboz tetején található bemenettel. Ha most elkezded kapcsolgatni a gombokat, látható lesz a változás.



3.2. ábra: A Bang és a Toggle

A hagyományos számdobozokból, amiket egyszerű dupla jobbklikkel tudsz előhozni, szintén van néhány előre elkészített alapverzió: ezeket a vektorvégződésűek között találod az üzenetmenüben. Az ilyenekre elsősorban akkor van szükség, ha például egy bemenetnek több értékre van szüksége, mondjuk *x-y-z* koordinátákra. Ezt manuálisan is elő tudod állítani az Inspektorban, sőt akár sokkal több számot is hozzá tudsz adni.



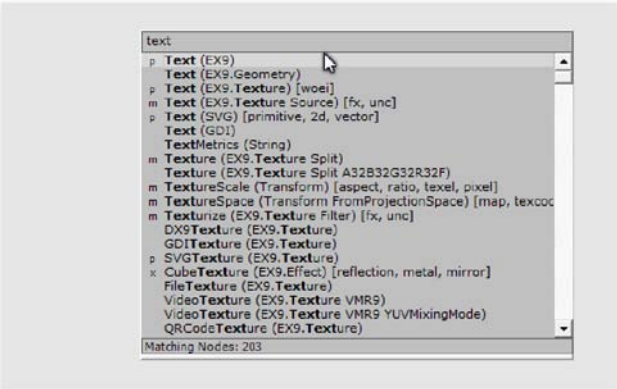
3.3. ábra: A Vector-fajták

A szövegdobozt elsősorban szövegbevitelre szokás használni, mivel vannak olyan *node*-ok, amik csak szöveges formátumot fogadnak el. Ettől függetlenül ebbe is tudsz

számokat írni, csak nem fogod tudni bekötni egy számalapú bemenetbe: ezt a legegyszerűbben úgy tudod kipróbálni, ha az üzenetmenüből kiválasztod a *string*-et, és melléteszel két egyszerű számdobozt. Ilyenkor, ha a *string* kimenetére klikkelsz, és próbálod hozzákötni a számokhoz, nem fog sikerülni. (Ha elkezdtél egy összekötést, de mégsem szeretnéd összekötni semmivel, egyszerűen klikkelj a háttérre a jobb egérgombbal, és eltűnik a vonal.) Ha viszont az egyik számdoboz kimenetére klikkelsz, akkor láthatod, hogy a másik számdoboz bemenete megvastagodik, ezzel jelezve azt, hogy abba kötheted, miközben a *string* bemenete ugyanolyan marad, vagyis oda nem tudod bekötni.

3.4. ábra: A String és a szám üzenet

A legjobb példa, amivel az eddig tanultakat ki tudod próbálni, ha dupla balklikkel begépeled a *text* funkciót: ekkor a program felajánlja az összes *text*-et tartalmazó *node*-ot. Ebből neked a legfelső kell, vagyis a *Text (ex9)*. Ha végighúzod az egeret a doboz bemenetein, láthatod, hogy nagyon sok paraméter változtatható rajta, hasonlóképpen a szövegszerkesztőkhöz.



3.5. ábra: A Text modul

A kimenetén a következő olvasható: *Layer: [Supports: ex9 layer].* Ez azt jelenti, hogy a kimenetén egy *ex9* megjelenítőt vár, vagyis ahhoz, hogy ezt a szövegmodult meg tudd jeleníteni, nyitnod kell egy renderablakot. Ehhez írd be a *node*-menübe, hogy *renderer*, és válaszd ki a legfelsőt, a *Renderer (ex9)* nevűt. Klikkelj a *text* kimenetére,

ami rögtön fel is villan, jelezve, hogy rá tudod kötni a *Renderer* bemenetére. Ha összekötötted őket, a renderablakban meg is jelenik a „vvvv" karaktersor, mivel a *Text* doboz szövegbemenetébe ez van beleírva.

3.6. ábra: Hello Renderer

Ahhoz, hogy ezt átírd, klikkelj a *String* doboz kimenetére és látni fogod, hogy melyik *Text* bemenetre kötheted, mivel az kiemelődik. Ezután klikkelj jobb gombbal a *String*-re, és írj bele bármilyen példaszöveget, mondjuk azt, hogy „HELLO". Korábban láthattad, hogy a *Text* doboz bemenetei között van olyan is, hogy *size* – viszont ha jobbklikket nyomsz erre a bemenetre, majd elkezded lefelé húzni az egeret, láthatod, hogy nem a mérete, hanem a megjelenített szöveg felbontása változik. Ide egy szimpla jobbklikkel írd be, hogy „300", és nyomj egy entert.

3.7. ábra: A felbontás 10 és 300

Ahhoz, hogy a szöveg méretét is változtatni tudd, egy *transform* dobozt is be kell kötnöd. Klikkelj a *node*-menübe és írd be azt, hogy *transform,* majd válaszd ki a *Transform 2d* funkciót. Ha a *transform* kimenetére klikkelsz, rá tudod kötni a *Text* második bemenetére. Ha egérrel végigmész a *transform* bemenetein, láthatod, hogy van x y pozíció, *scale, rotate* stb. Neked most a *Scale x, Scale y* funkciók fognak kelleni.

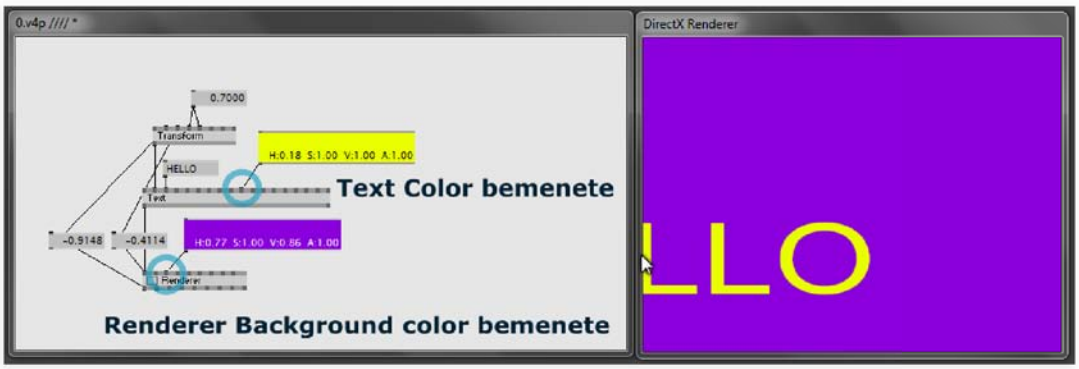
Hozz létre egy számdobozt, majd annak a kimenetét kösd rá mind a két *scale*-re. Ekkor eltűnik a felirat a *Renderer*-ből, mivel a *scale* mérete most még 0-ra van állítva. Változtasd meg mondjuk 0.5-re.

3.8. ábra: A Scale x, y

A szöveg pozíciójáért a *transform*-on belül a *Translate x, y* a felelős. Ha egérrel szeretnéd vezérelni a felirat helyzetét, a *Renderer* legelső két kimenetét, vagyis az *x*-et és az *y*-t be kell kötnöd a *Translate x, y*-ba.

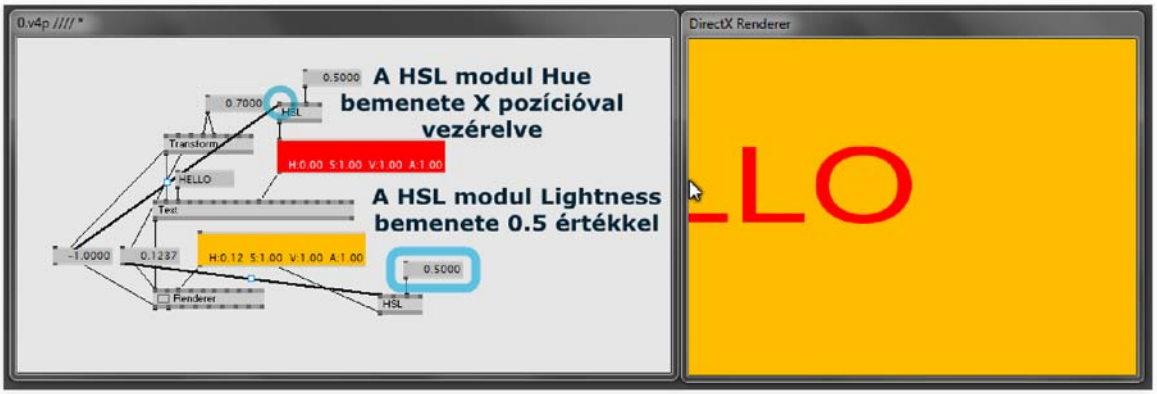
3.9. ábra: A felirat mozgatása egérrel

A színek vezérlését a *Color* üzenettípussal tudod megoldani. Hozz létre két *Color* dobozt, majd az egyiket kösd a *Renderer*-be, és változtasd meg a színét a jobb egérgomb folyamatos nyomva tartásával, miközben az egeret mozgatod. A másik *Color*-t kösd be a *Text* doboz *color* bemenetébe, hogy a szöveg színét is meg tudd változtatni.



3.10. ábra: A Hello színesen

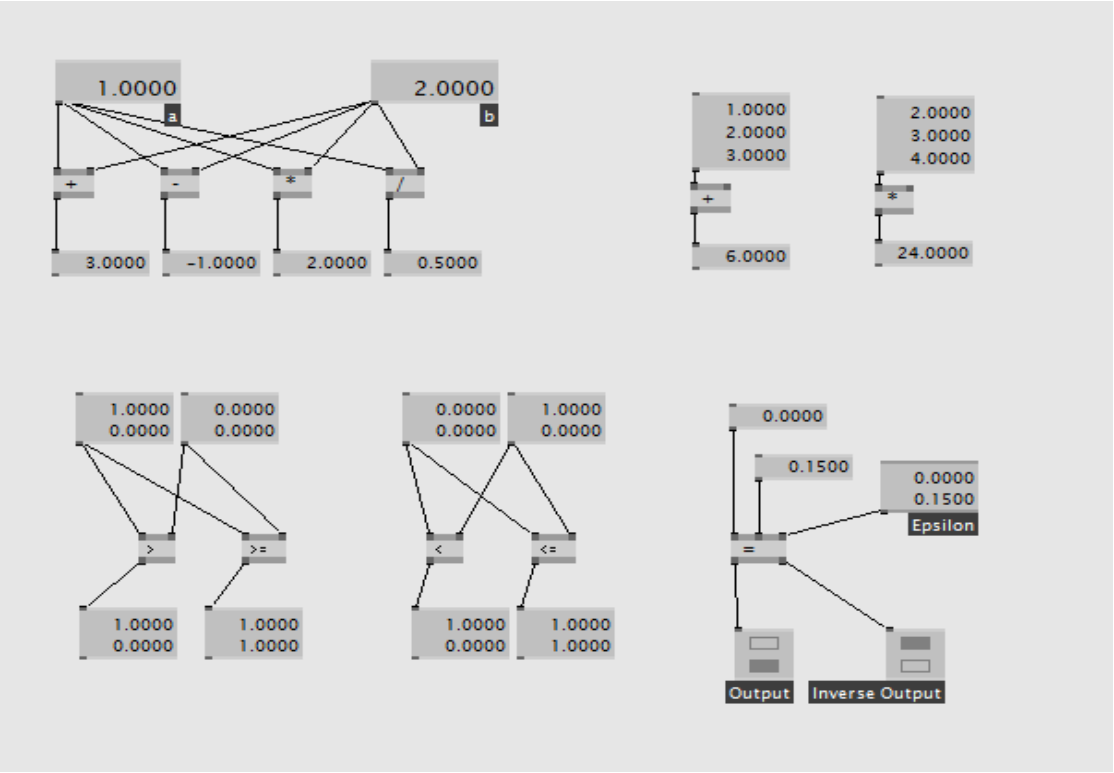
Ahhoz, hogy ezt is egérrel tudd vezérelni, szükséged lesz egy olyan dobozra is, ami a számokat színné alakítja. Erre többféle *node* is van, ezek közül használd most *HSL (Color Join)* nevűt. Ahogy a nevéből következik, ebben a *Hue, Saturation* és *Lightness* paraméterek változtatásával tudsz színeket előállítani. Ha ennek a kimenetét rákötöd a *Text* dobozra kötött *Color* doboz bemenetére, a szöveg fehérre változik, mivel alapesetben a *Lightness* 1-re van állítva, így hiába állítgatod a *Hue*-t, a szöveg továbbra is fehér marad. Ha viszont a *Lightness*-t lehúzod 0.5-re, akkor már teljes színeket tudsz kapni. Ezután már a pozíciótól is függővé tudod tenni a színeket, csak kösd össze a *Renderer* doboz *x* kimenetével a *HSL (Color Join)* doboz *Hue* bemenetét.



3.11. ábra: A HSL mozgatással

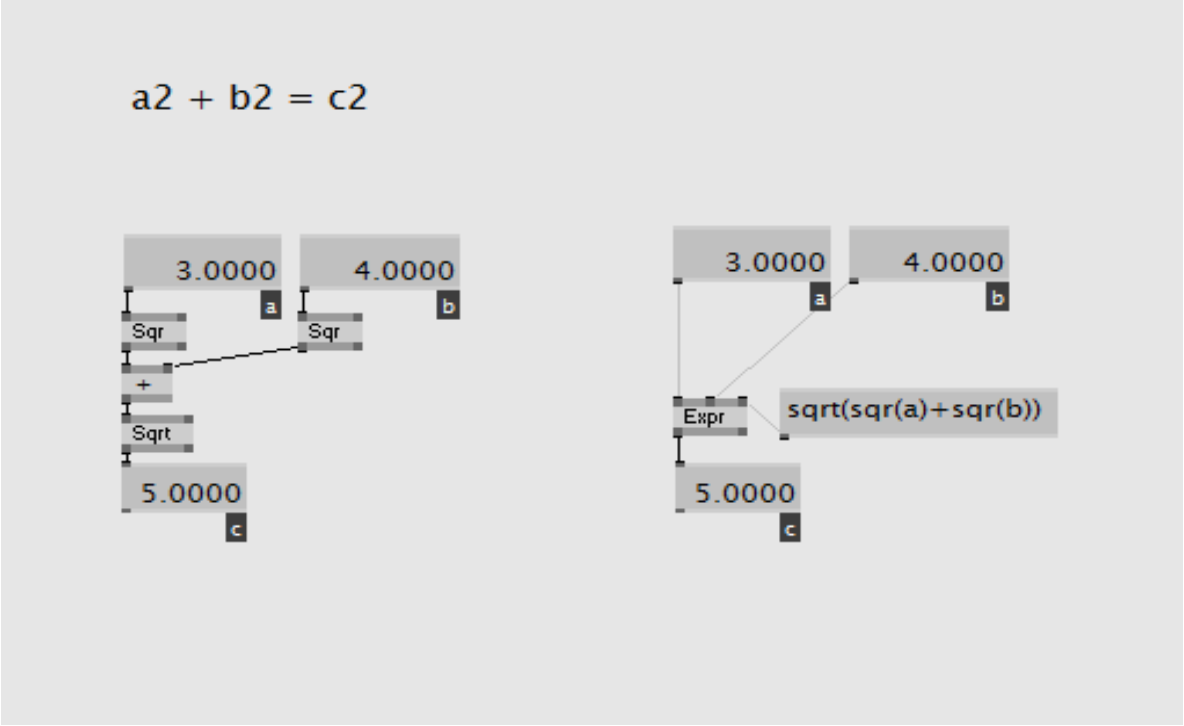
4. Komplexebb matematikai feladatok, fájlba írás és olvasás

Elsőként vegyük a leggyakrabban használt alapműveleteket, az összeadást, kivonást, szorzást és osztást. Ehhez hozz létre mindegyikből egyet a mellékelt ábra alapján! A node menüben ezekre a műveletekre rengeteg különböző típust ajánl fel a program; most itt nekünk csak a *value* végződésűekre lesz szükségünk. Ezek használata eléggé magától értetődik. Ha több számból álló vektort szeretnél összeadni vagy esetleg szorozni, erre a *Value Spectral* node-ot tudod használni.



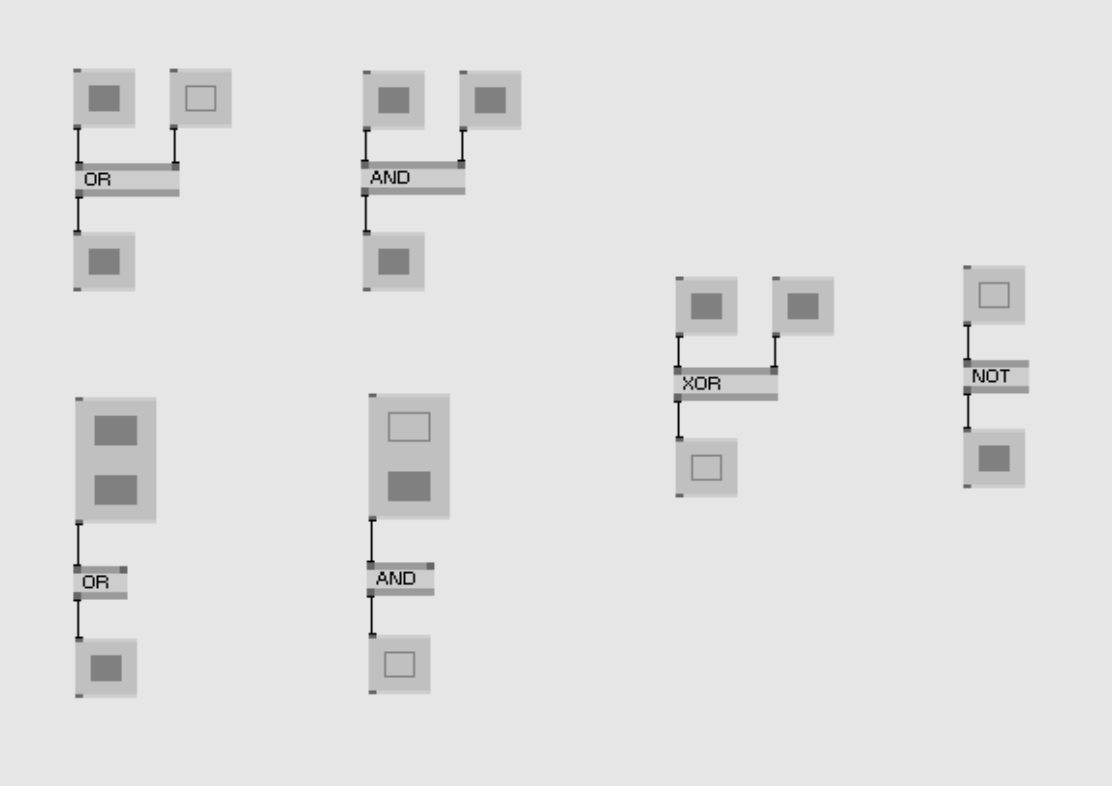
4.1 ábra: Matematikai alapműveletek

Képletek megoldására két lehetőséged is van. Egyrészt összerakhatod magát a képletet a benne szereplő műveletek moduljaiból is, másrészt használhatod az *Expr* modult is, amibe a teljes képletet kell betáplálni.



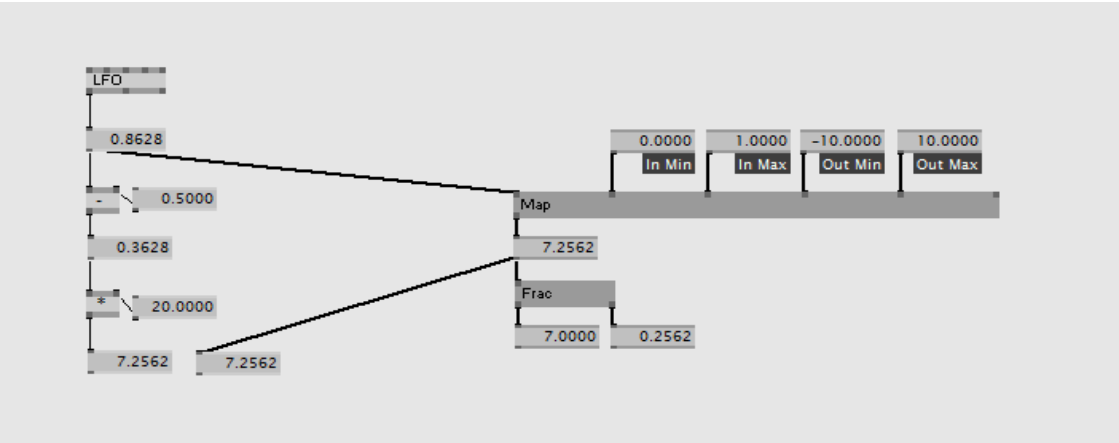
4.2. ábra: Komplexebb matematikai műveletek

A logikai kapuk használata szintén alapvető fontosságú, mivel számos esetben egyszerre több dolognak kell teljesülnie ahhoz, hogy valami elindulhasson, illetve lehet, hogy triggerelni kell a bemenetet. A logikai kapumodulok a következők: az *OR* modul kimenete rögtön 1-re vált, ha bármelyik bemenetén kap egy 1-est – ezt például akkor érdemes használni, ha egy bemenet (például egy modul reset bemenete) több helyről kaphat impulzust, hogy végrehajtsa a műveletet. Az *AND* kapu kimenete csak akkor vált 1-re, ha az összes bemenet 1-en van, míg a *XOR* ennek az ellentéte, addig küld ki csak 1-et, ha nem azonosak a bemenetek. A *NOT* modul pedig invertálja a bemenetét, vagyis ha a bemenetén 1 van, akkor a kimenetén 0.

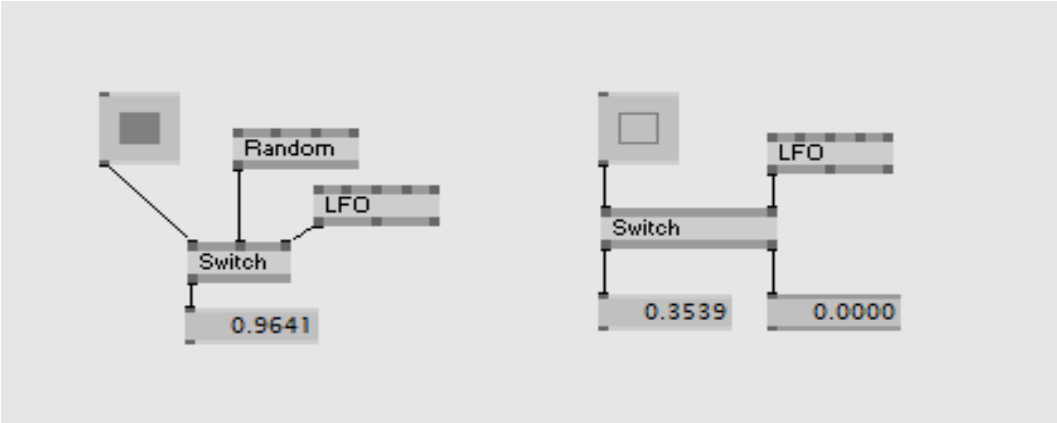


4.3. ábra: Logikai kapuk használata

Amennyiben a kapott eredmény nem a nekünk megfelelő tartományba esik, további szorzással, osztással vagy hozzáadással állíthatod be az ideális pozícióra. Ennek megkönnyítésére érdemes a *Map* modult használni, ami kifejezetten erre szolgál. A *Map* 2. és 3. bemenetén állíthatod be, hogy a forrásod mely két érték között változik, míg a 4. és az 5. bemeneten a kimeneti tartományt lehet beállítani.

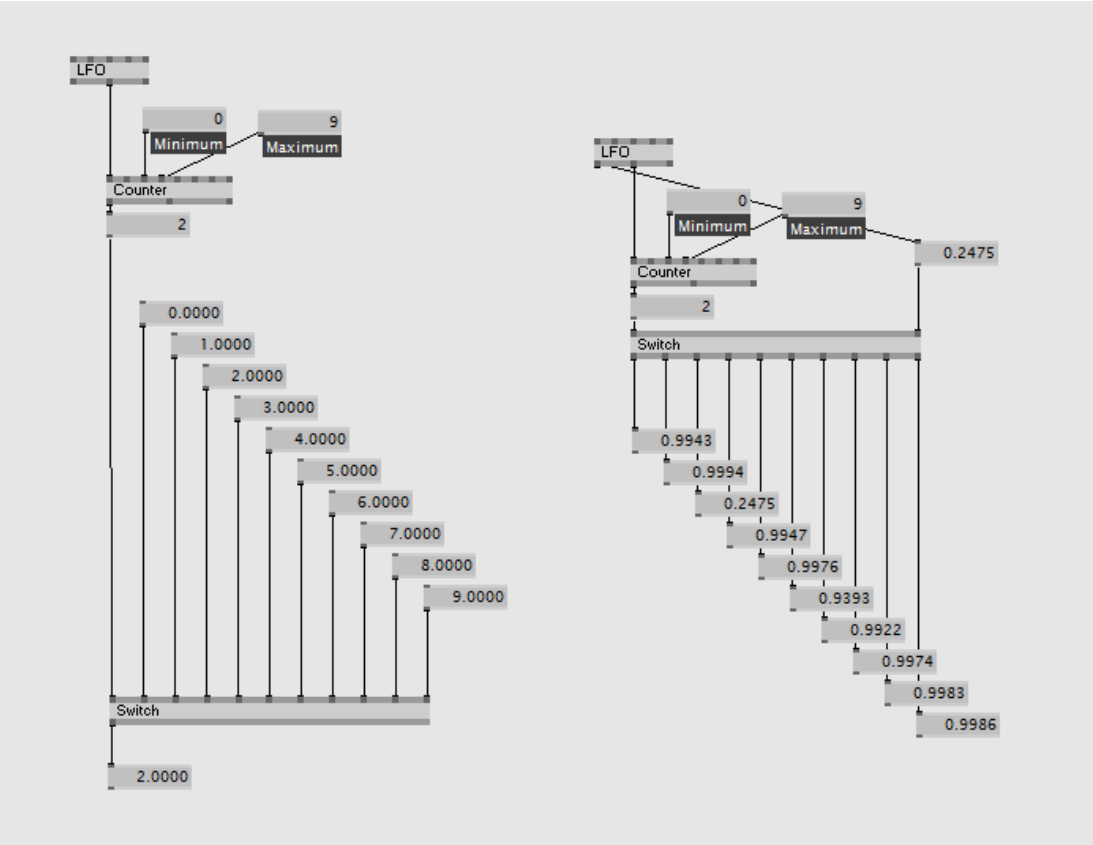


4.4. ábra: Átméretezés



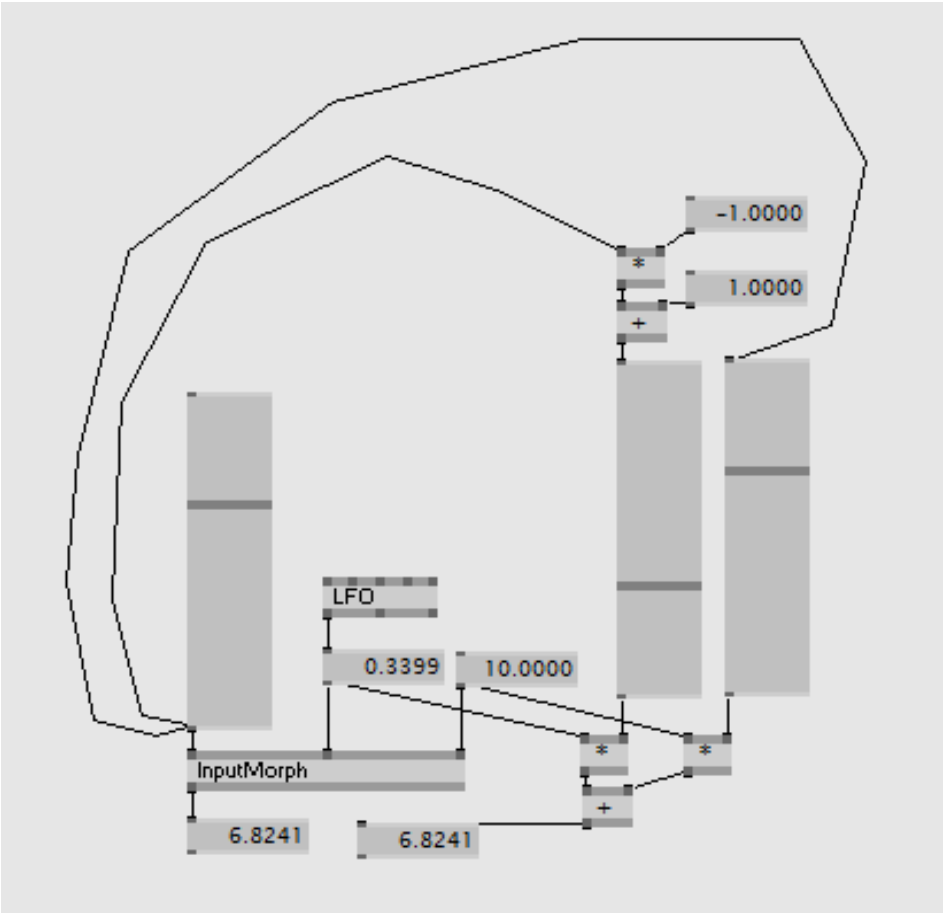
4.5 ábra: Bemenet- és kimenetkapcsolók

Ahhoz, hogy két érték között váltani tudj, illetve hogy választhass egy forrás két bemenete közül, *Switch*-re van szükség. A kapcsolófajták bemenetei és kimenetei könnyen többszörözhetőek az *Inspektor* segítségével, ahogy az a 4.6-os ábrán látszik.



4.6. ábra: Kapcsolók több ki-, ill. bemenettel

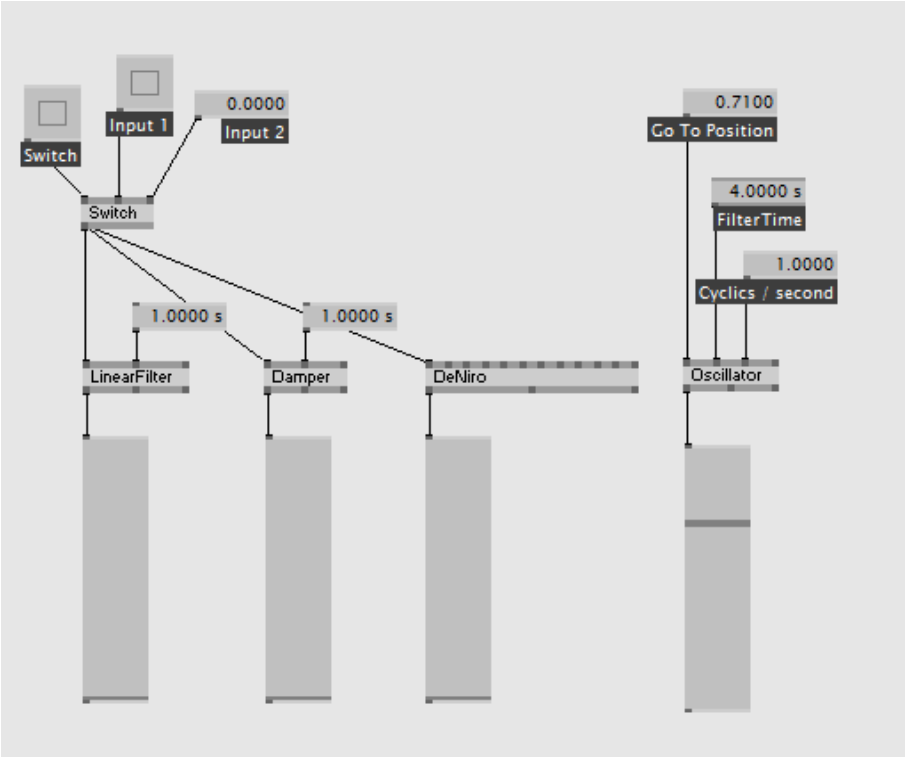
Ha két bemenet között szeretnél átmenetet képezni, arra a legegyszerűbb mód az *InputMorph* használata. Ezt azonban megoldhatod hagyományos modulok használatával is, ahogy az a 4.7-es ábrán látható.



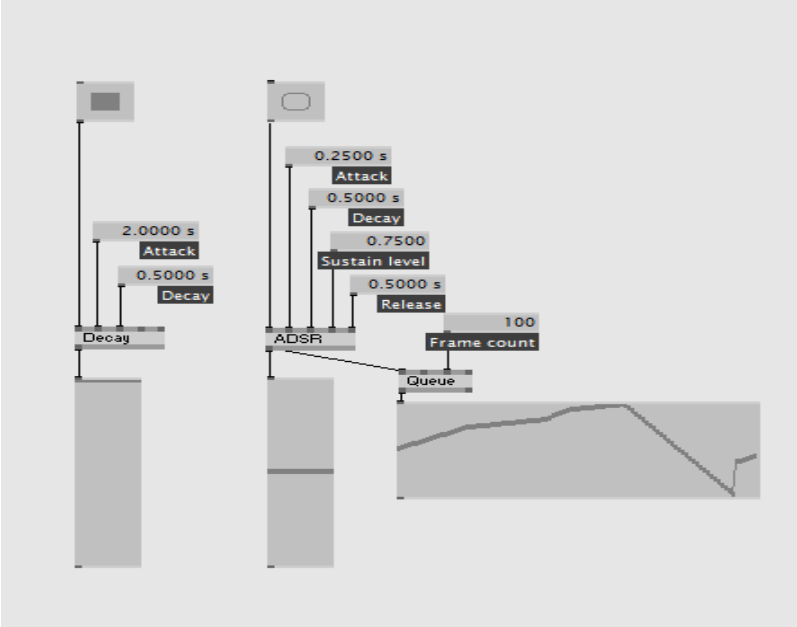
4.7. ábra: Az InputMorph

Ha a bemeneted változik, de a két váltás között nem elég szép az átmenet, ezen a *LinearFilter, Damper, DeNiro, Oscillator* és a *Decay* nevű filterekkel tudsz változtatni.

A *LinearFilter* lineáris átmenetet képez két változás között adott idő alatt. A *Damper* gyors felfutási idővel és lassú lefutási idővel oldja meg ugyanezt (hasonlóan egy rugó kiengedéséhez), míg a *DeNiro* mozgása lassan indul el és gyorsabban áll le (hasonlóan például a járművekhez). Az *Oscillator* hasonlóan működik, csak ennek utómozgása is van. A *Decay* szintén egyfajta *LinearFilter*, viszont ennek a felfutási idejét (*Attack,* 2. bemenet) és a kifutási idejét (*Decay*, 3. bemenet) is külön-külön tudod állítani. A szintetizátorokból jól ismert burkológörbe-generátorral vagyis *ADSR*-el már sokkal több paramétert tudsz változtatni.

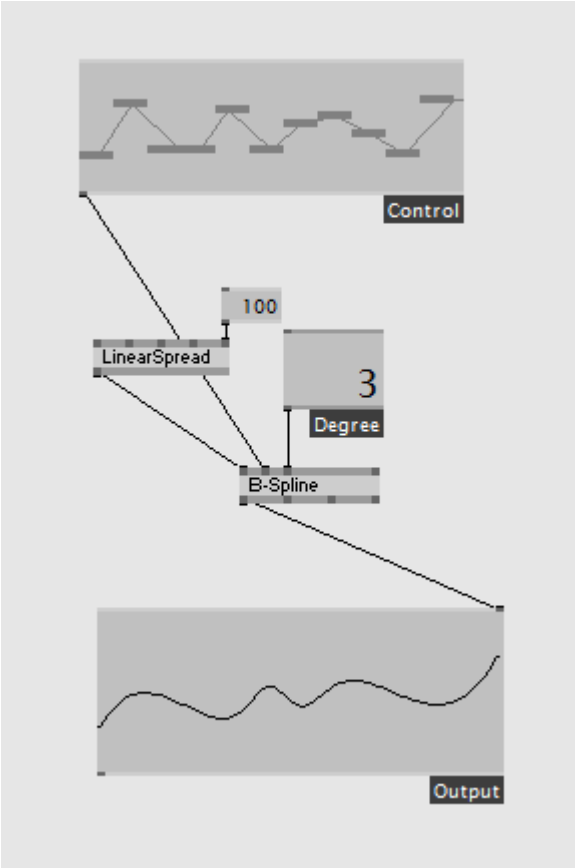


4.8. ábra: Filterek



4.9. ábra: Decay és az ADSR

Ha a bemeneted több értékből áll, és azt szeretnéd simítani, ahhoz a *B-Spline* modult kell használnod. Ez az 1. bemenetén egy *LinearSpread*-et vár, amivel azt tudod beállítani, hogy a kimenet összesen hány számból álljon. Vagyis ha az eredeti forrásod mondjuk 12 számértékből tevődik össze, a *LinearSpread* pedig 100-ból, akkor a kimeneted is egy 100 értékből álló forrást közvetít. A modul a 2. bemenetén várja a kontrollforrást, míg a 3. bemeneten tudod beállítani a simítás mértékét. Ha ez utóbbit 0-ra állítod, a végeremény szögletes lesz, ha 1-re, akkor marad az eredeti, míg 3-as értéknél már egy lágy kimenetet kapsz.

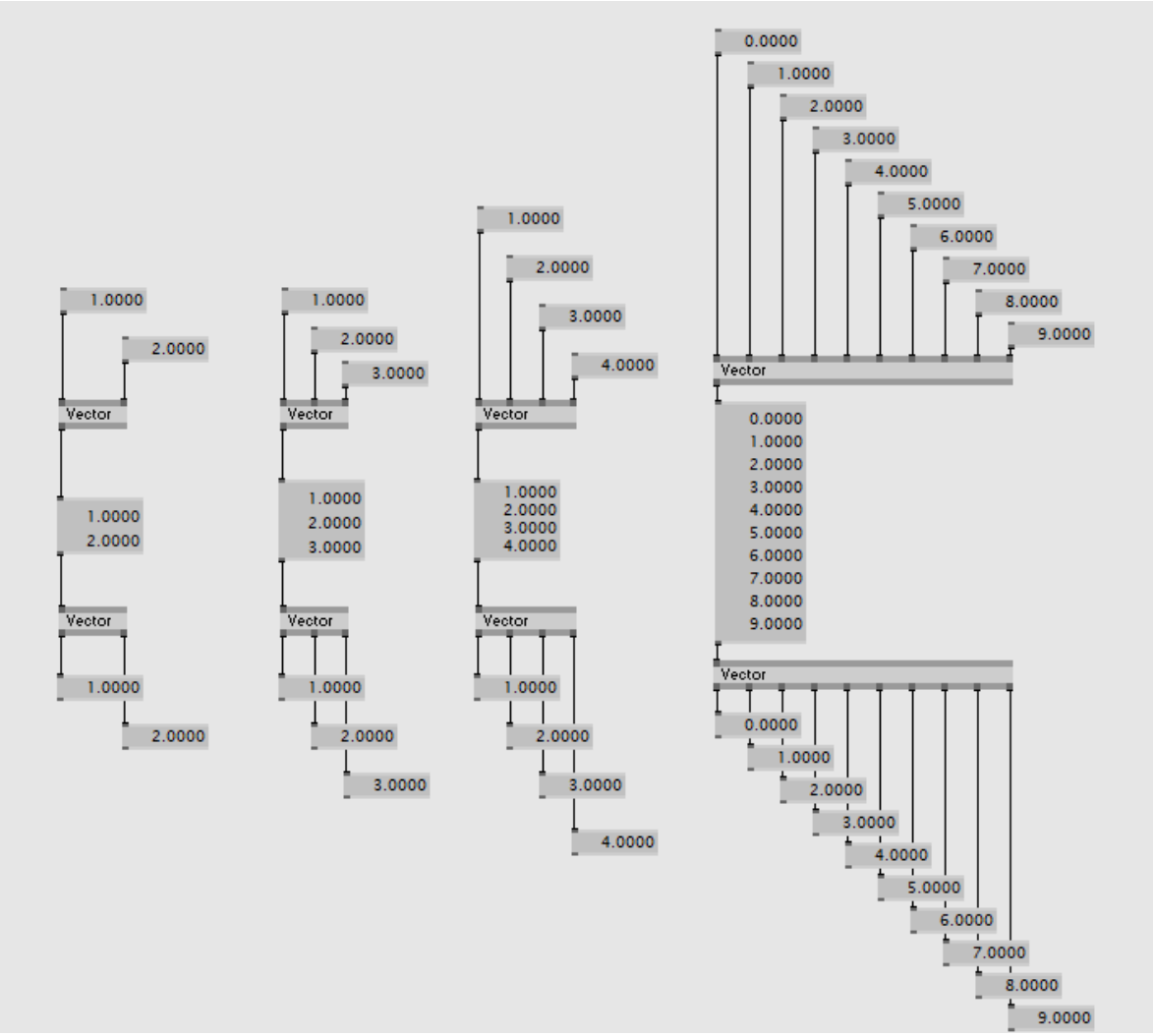


4.10. ábra: B-Spline

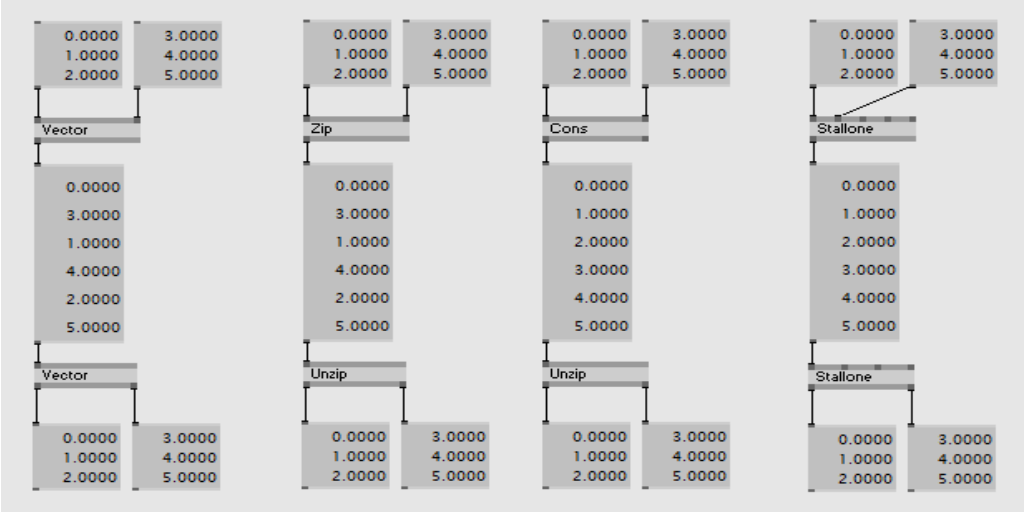
A vektorokról már ejtettünk néhány szót, de most nézzük meg részletesebben! Minden üzenettípus több üzenetből is állhat, ezek összetételére és szétbontására pedig több módszer is létezik. Eddig elsősorban a *Vector (joint), Vector (split), Cons,* és *Stallone* modulok voltak elérhetők. A *Vector*-ból négyféle van: *2s, 3d, 4d,* és *Spread*, az első három fix számú bemenettel vagy kimenettel rendelkezik, míg a *Spread* változatban te adhatod meg a bemenetek és kimenetek számát. A *Vector* modul úgy működik, hogy ha több értéked van a bemeneteiden, akkor mindegyikből csak az elsőt, majd a másodikat stb. teszi egymás után. Ha azt szeretnéd, hogy a kimeneti

listádban először az első bemenet összes értéke legyen sorba rakva és utána a második bemenetedé következzen, akkor a *Cons* nevű modult kell használnod. A *Stallone* egy modulárisabb node, itt *Inspektor* segítségével beállíthatod, hogy hány ki- és bemenetet akarsz használni, és azt, hogy a lista hogyan álljon össze.

Nemrégiben jelentek meg a *Zip* és az *Unzip* modulok. Ezek jóval hatékonyabbak, ha 25-nél többel dolgozol egyszerre, így sok szám listázásánál ezt érdemes használni. Ezeknél a moduloknál is *Inspektor*-ral tudod változtatni a bemenetek és a kimenetek számát.

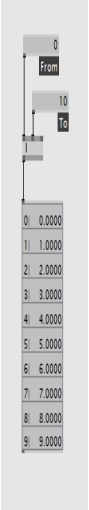


4.11. ábra: Vektorok fajtái

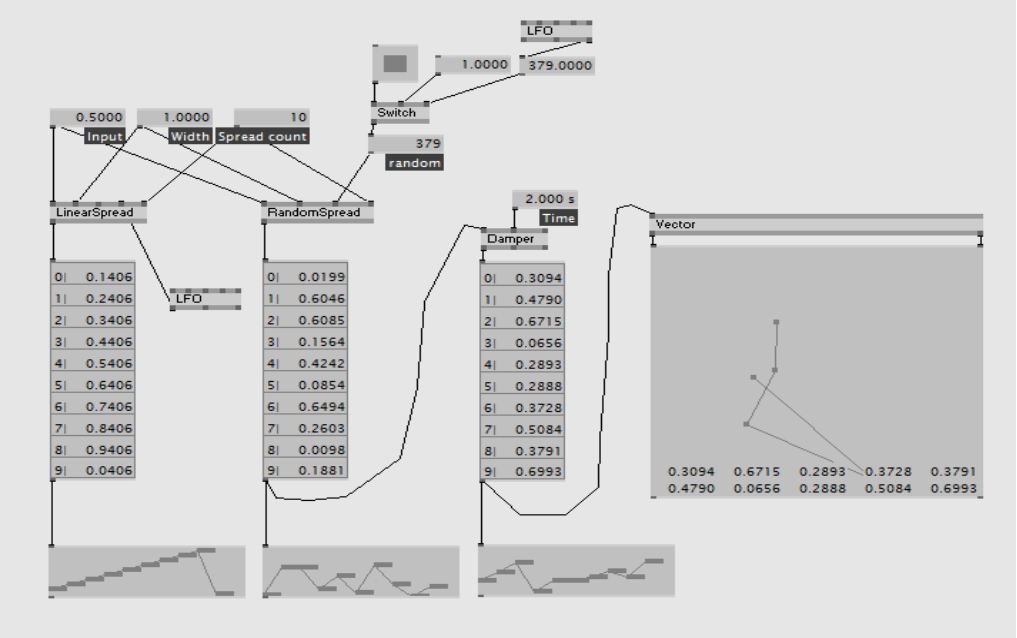


4.12. ábra: A Vector, Zip, Cons és a Stallone

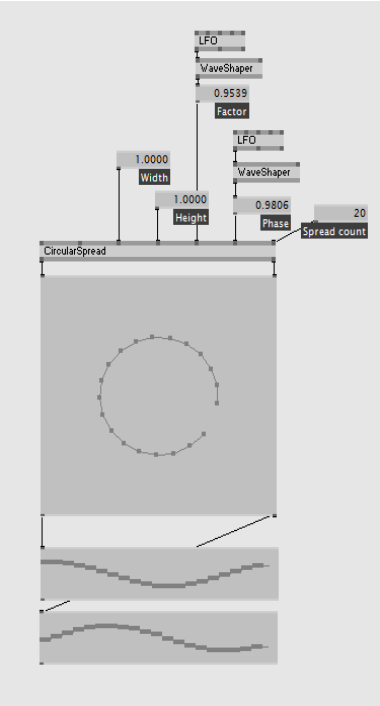
Ha listát szeretnél létrehozni, mondjuk 0-tól 100-ig terjedően, arra az *I* modult kell használnod. Ez a *LinearSpread*-hez hasonlóan működik: ha az 1. bemenetére beírsz egy 0-t, a 2. bemenetre *(Width)* pedig egy 1-est, míg a spread count bemenetére pedig 10-et, akkor -0.5 és 0.5 között kapsz egy lineáris listát. Ugyanezt tudod elérni, csak véletlenszerű értékekkel, ha a *RandomSpread*-et használod. A modul randomját a *Random Seed* nevű bemenetén tudod állítani. A *CircularSpread* pedig egy körvonal mentén hoz létre számokat, amiknek *x* és *y* értékük van. Ebben nemcsak a kör középpontjának *x* és *y* paraméterét, de a kör szélességét és magasságát, és a kitöltését is be lehet állítani.is be lehet állítani.



4.13. ábra: Az I



4.14. ábra: A LinearSpread és a RandomSpread



4.15. ábra: CircularSpread

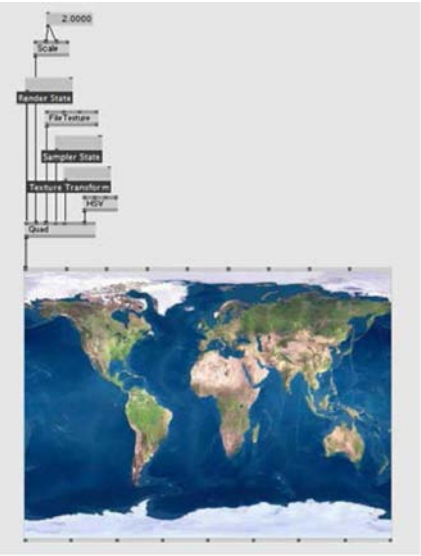
A fájlba írás és olvasás az utolsó lecke témája lesz.

5. A rendertípusok

Elsőként vegyük a node-menüben található legelső, és egyben leggyakrabban használt rendermodult, a Renderer EX9–et! Ezzel az összes két- és háromdimenziós modul megjeleníthető.

A modul kipróbálásához előbb vegyél egy quad DX9-et, és kösd rá a rendermodulra. Ekkor a rendermodulban megjelenik egy fehér négyzet, amit át kell méretezni, hogy teljes képet kaphass. Ezt a transform segítségével tudod megtenni, ha a scale x scale y paramétereit 2-re írod át. A Quad egy szimpla lap, ami mind képek, mind videók megjelenítésére képes. A képeket a Filetexture nevű modullal tudod betölteni. Ha a modul kimenetét rákötöd az arra alkalmas bemenetre, a lap üres állapotában először feketére vált, amíg jobb klikkel ki nem választod a betölteni kívánt képet a Filetexture legelső bemenetén.

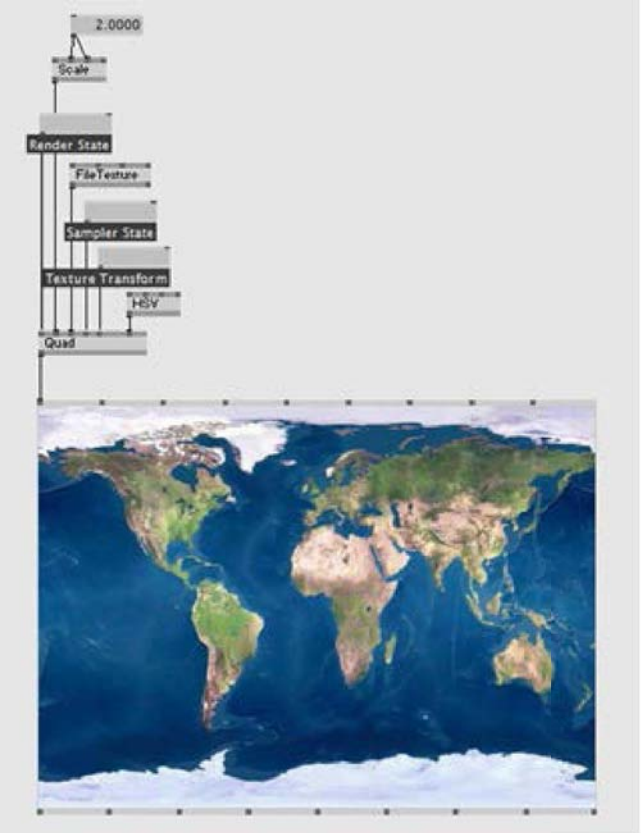
Ha előre tudod, hogy végig teljes méretű képre lesz szükséged, amit nem feltétlenül akarsz átméretezni, erre a legjobb választás a FullscreenQuad modul, ez ugyanis rögtön a megfelelő méretben jelenik meg. Ha később mégis szeretnéd átméretezni, ezt ugyanúgy a transform bemenettel tudod megtenni, mint a Quad esetében.



5.1. ábra: Full Screen Quad Render

Mindkét modul rendelkezik még color bemenettel is, amivel egyszerűen tudod beállítani a kép színét és átlátszóságát. A sima transform bemeneten kívül pedig érdemes megemlíteni még a Texture Transform bemenetet (lásd a példafájlban), ami a textúrák méretezésére szolgál. Ez akkor lehet érdekes, ha a textúrád kisebb, mint maga a megjelenítő: ekkor a program olyan funkciókat ajánl fel, amivel megadhatod, hogy ilyenkor a képet ismételje-e, vagy az utolsó sornyi pixelt húzza-e el stb. Ezt a funkciót a sima Quad-nál a Sampler State funkcióra kötött Adress-el tudod előhozni, míg a FullscreenQuad modulnál az utolsó előtti füllel, a Texture Adress Mode-dal.

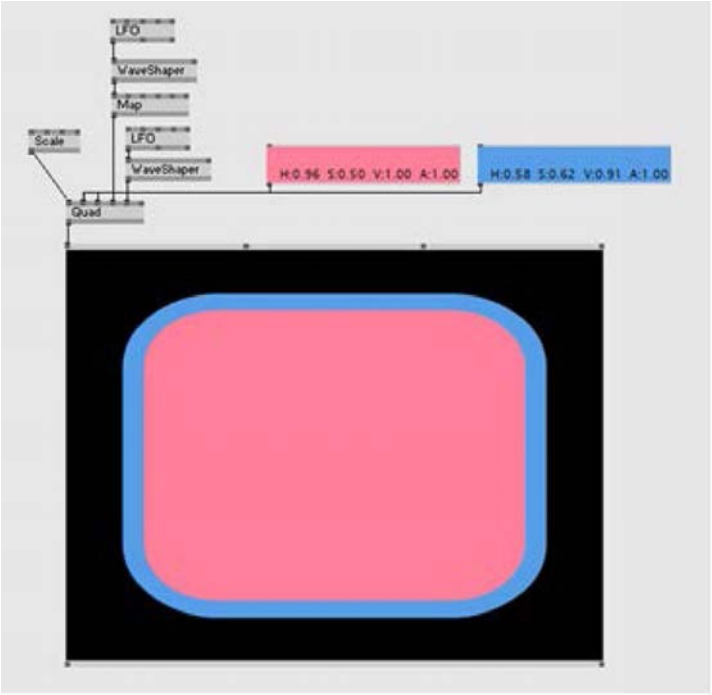
Ezek a rendermodulok alapvető funkciói, melyeket továbbiakkal bővítünk majd a következő leckékben.



5.2. ábra: Quad Render

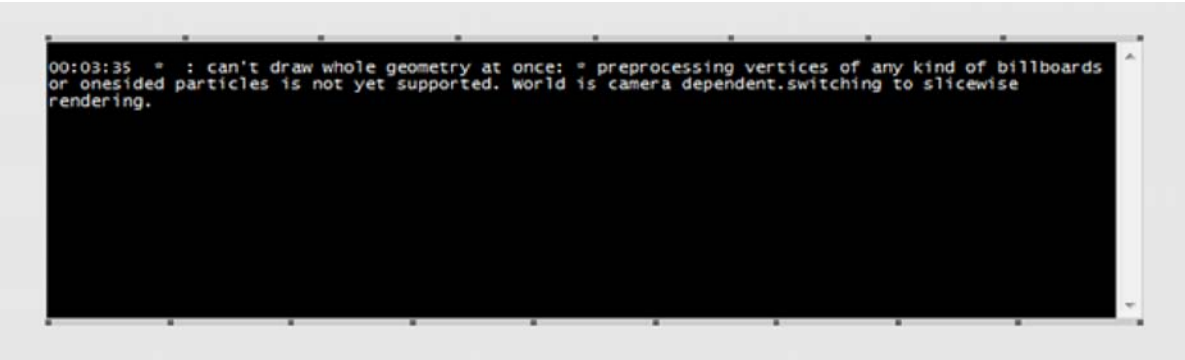
A soron következő rendermodul az SVG, vagyis Scalable Vector Graphics, ami egy xml alapú vektoros megjelenítési forma. Ez elsősorban webre lett kitalálva, és hozzá is tartozik Quad, aminél a transzformálás és a színezés hasonló módon működik, mint az EX9 típusú Quad-nál, csak ez a modul nem képes a pixelgrafikus képek megjelenítésére. Ha a node-menüben rákeresel az SVG-hez kapcsolódó modulokra,

látni fogod, hogy elég kevés modul elérhető hozzá; ez a program leginkább egyszerű, kétdimenziós grafikák generálására alkalmas, ráadásul nagy hátránya, hogy erősen igénybe veszi a processzort. Így ezzel a modullal egyelőre nem érdemes behatóbban foglalkoznod, kivéve, ha az általad generált grafikából kifejezetten xml formátumú mentésre van szükséged.



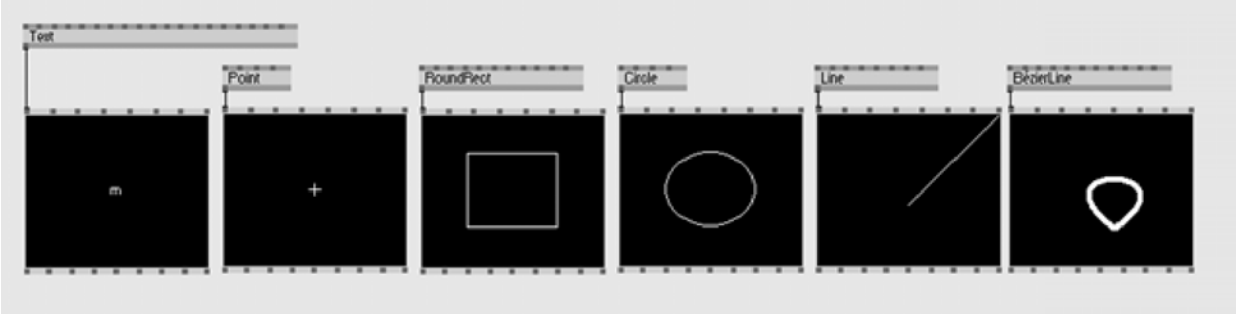
5.3. ábra: SVG Render

Egy másik a TTY render, ami üzenetek megjelenítésére alkalmas. Elsősorban a hibaüzeneteket vagy cselekményeket jelenítik meg vele, de saját üzenet betáplálására is alkalmas egy TTY típusú Write modul segítségével.



5.4. ábra: TTY Render

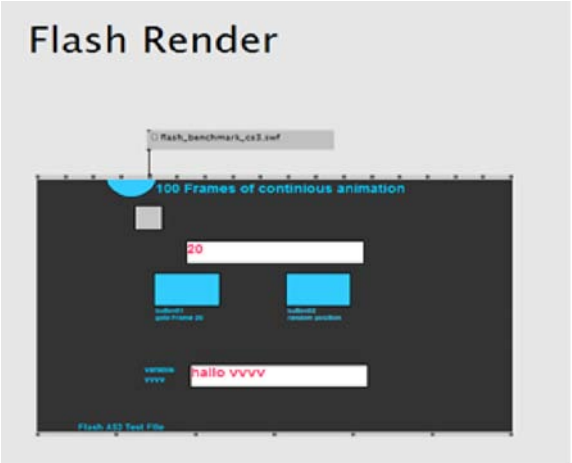
A GDI, vagyis Graphic Device Interface render szintén egy kétdimenziós renderer, vonalalapú grafikák megjelenítésére. A használható modulok listája a következő: Point, Text, RoundRect, Circle, Line és a BézierLine. A többivel ellentétben ennek már nincs transform bemenete, hanem minden paraméter a modulon belül állítható, így például az x és y pozíció, a szélesség, a magasság, a szín stb. Egyes modulokban a vonalvastagság is állítható. Ritkábban használják.



5.5. ábra: GDI Render

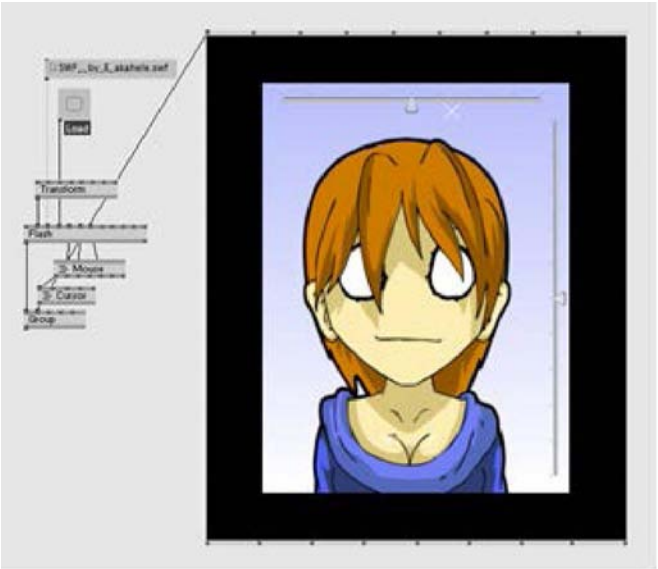
Végül ejtsünk szót az SWF fájlok lejátszására szolgáló Flash-megjelenítőkről! Az ilyen típusú fájlok megjelenítésére két lehetőség van.

Az első a Flash renderer, ami külön modul nélkül is alkalmas flashfájlok megjelenítésére: itt magába a rendermodulba kell betölteni a fájlt, ahol meg tudod adni a megjelenítés méretét, hogy a lejátszást melyik kulcskockától kezdje és melyik kulcskockánál fejezze be, valamint hogy ismétlődjön-e. Programozott SWF-ek esetében a változók kívülről is vezérelhetők, így akár interaktív flash is létrehozható vele.



5.6. ábra: Flash Render

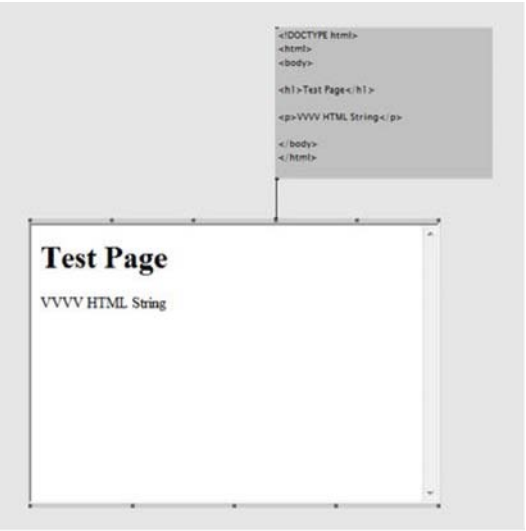
A másik megjelenítési lehetőség a Flash EX9, amihez az EX9-es rendermodult kell használnod. Teljesen hasonlóan működik, mint egy Quad, csak SWF fájlt játszik le. Külső vezérlésre a fentebb említett változók megadásán kívül itt már az egér pozícióját, vagy az egérgombok jelét is használhatod, ami később nagyon hasznos lehet.



5.7. ábra: Flash EX9 Render

Weboldalak betöltésére is van lehetőség, ebből is kétféle lett integrálva a vvvv -be

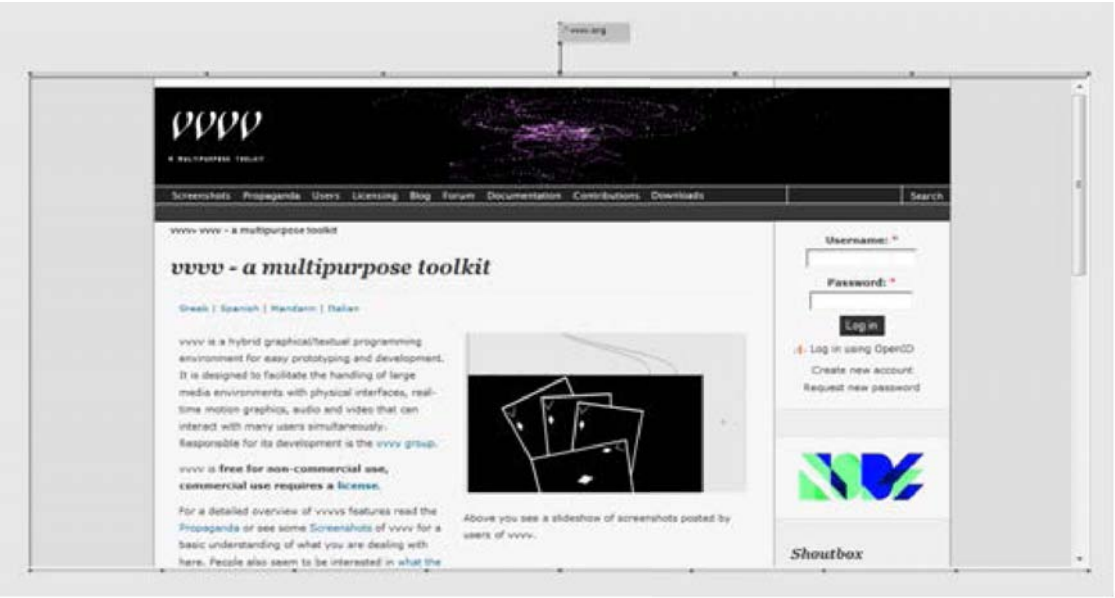
Az első a HTML Render String. Ezzel a modullal Html kódokat tudsz lefuttatni, így a végeredményben egy komplett weboldalt is láthatsz. Akár arra is használhatod a Html Render String-et, hogy egy beágyaz, például Youtube vagy egyéb video forrást lefuttass. Mindegyik webbel kapcsolatos modul a Windows alapértelmezett böngészőjét használja, vagyis az Internet Explorert.



5.8. ábra: HTML Render String

A másik Html alapú renderer modul a sima HTML Render. Ebben a modulban létező weboldalakat tudsz lefuttatni. Ezekből az oldalakból pedig kép és szöveges információt egyaránt kitudsz nyerni.

A VVVV legújabb fejlesztése, amit már az alpha verziókba is integráltak, az a Chrome böngésző, ami jelentős fejlődést jelent ezen a részen.



5.9. ábra: HTML Render

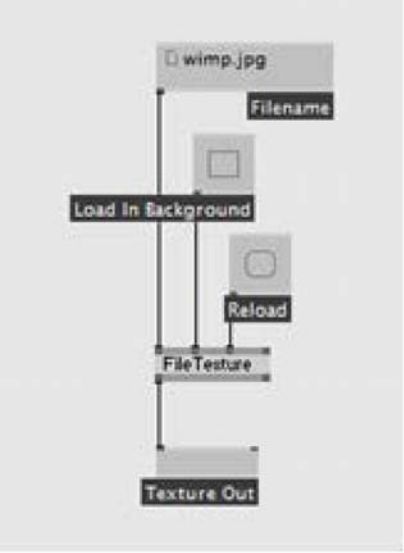
6. A rétegek *(layer)* kezelése

Ehhez először hozz létre két Ex9 Quad-ot és egy rendert, és mindkét Quad-ba köss bele egy-egy 2d Transform-ot!

Ahogy a számok esetében már tapasztalhattad, egy bemenethez itt is csak egy réteg adható meg, kivéve, ha azt összevontad egy vektorral. A képeknél hasonló a helyzet: ahhoz, hogy egy renderablakban két vagy annál több Quad-ot tudj megjeleníteni, Group-ra lesz szükséged (a bemenetek számát Inspektorban tudod beállítani).

Hozz létre egyet, és kösd bele a két Quad-ot! Ekkor a renderablakban csak egyetlen négyzet jelenik meg, mivel a két Quad most még egymás fölött van. Ahhoz, hogy egymás mellé helyeződjenek, a Quad modul Transform-jának az X pozíciójába írj -0.5-öt, a másodikéba pedig 0.5-öt!

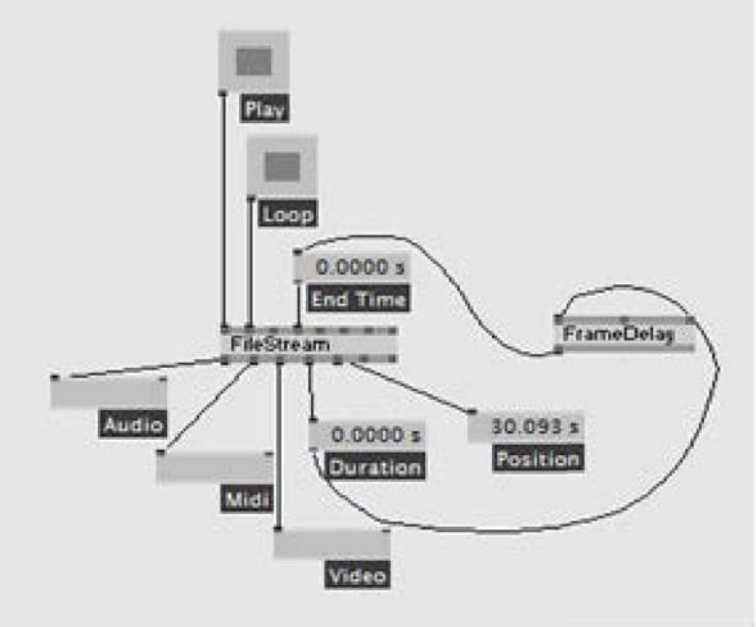
A textúrabetöltő moduloknak két alaptípusa létezik: az egyik az állóképek betöltésére használható FileTexture, amivel a korábbi leckékben már megismerkedhettél. A másik videó-, hang- és midifájlok betöltésére szolgál, és a FileStream (DShow9) nevet viseli.



6.1. ábra: File Texture

Ha létrehozol egy FileStream-et, a következő fajta bemeneteket találod rajta: lejátszás, loopolás, kezdés ideje, befejezés ideje, ugrás, ugrás pozíciója, sebesség és fájlnév. A kimenetek közül a következők lesznek hasznosak: az audiokimenet, a midikimenet, a videókimenet, a hossz és az aktuális pozíció. Ahhoz, hogy a videókimenet beköthetővé váljon egy Quad-ba, először át kell alakítani a VideoTexture modullal, mivel ez a kimenet alapesetben DirectShow típusú, neked pedig Texture Out-ra van szükséged.

Fájl betöltése és a videó elindítása után előfordulhat, hogy nem látsz képet: ennek az lehet az oka, hogy a bemeneteknél a befejezés ideje nem lett beállítva. Ezt a legegyszerűbben a Duration kimenet EndTime-al való összekötésével tudod elérni, ez viszont nem működik egyből, mivel így feedback jönne létre, ezt pedig a rendszer nem engedélyezi. Ezért először a FrameDelay modullal késleltetni kell 1 frame ideig (ezt a későbbiekben sokszor fogod használni, mivel van belőle textúrák és szöveges üzenetek kezelésére szolgáló típus is). Ha ez megvan, már össze tudod kötni a ki- és bemenetet, és indulhat a lejátszás.



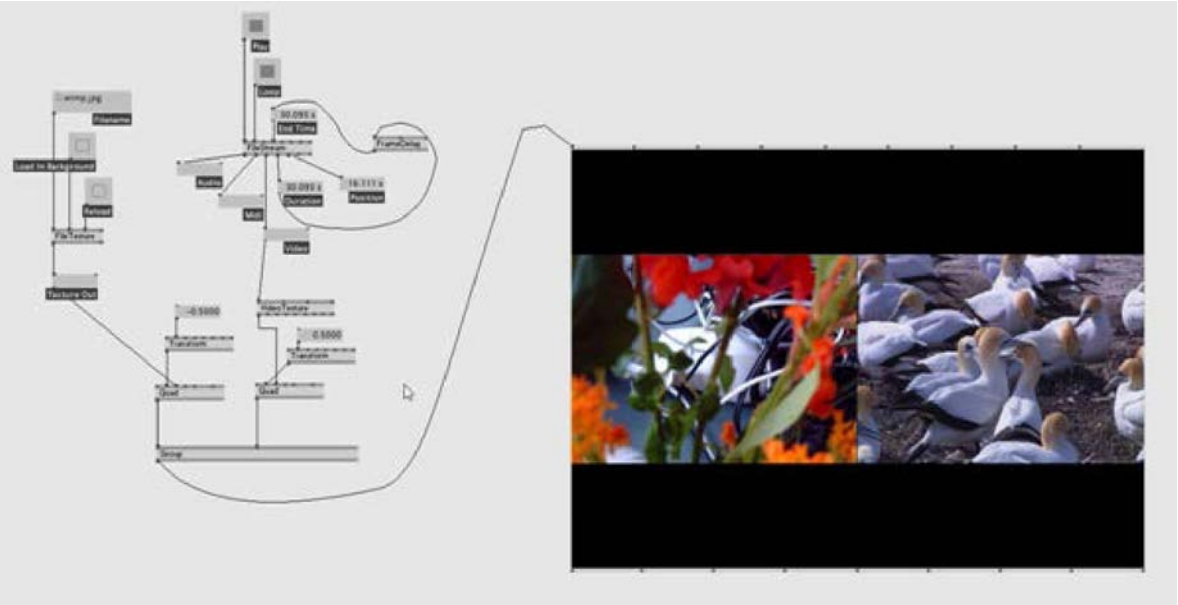
6.2. ábra: FileStream

A FileStream viszonylag sok formátumot kezel, de beletelhet némi időbe, amíg megtalálod a számodra ideálisat. Ha esetleg tudod, hogy nem akarsz se hangot lejátszani, se a kezdés, illetve a befejezés idején változtatni, akkor a legegyszerűbb, ha a MediaPlayer modult használod, mivel ebbe már beépítették a FrameDelay-t és a VideoTexture-t is.

A Videolan lejátszóját, a VLC Player-t nemrég kezdték el integrálni a rendszerbe: ez elég nagy változásokat fog hozni, mivel szinte minden fájltípust képes lejátszani, és nagyon sok funkció érhető el benne, amit már a lejátszóból ismerhetsz. Ezzel azonban itt nem foglalkozunk, mivel a modul még csak béta verzióban elérhető, és nem mindig működik tökéletesen.

http://vvvv.org/contribution/vlc-plugin-%28beta%29-a-new-audiovideo-player-to-texture-plugin

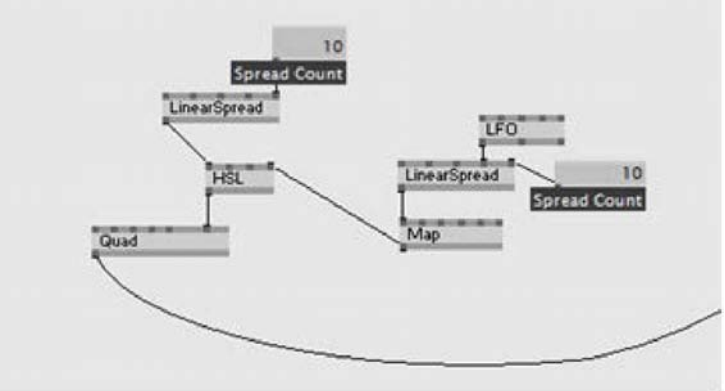
A rétegek használatának másik módja, ha egyetlen Quad-ot használsz, viszont több paraméterrel vezérled. Ekkor több réteg alakul ki, és a Group használatára nincs szükség.



6.3. ábra: Texturázott Quad-ok

A következő példa azt mutatja meg, hogyan tudsz váltani egy több textúrából álló Quad képei között a HSL modul segítségével.

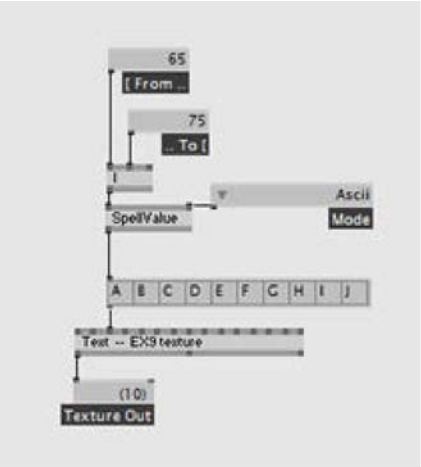
A lényeg, hogy átmenetet kell képezni a különböző rétegek között. Ehhez először hozz létre egy Quad-ot, azt kösd be egy renderbe, a Quad modul color bemenetére pedig köss rá egy HSL Color Join modult! Ezek után a HSL modul Hue és Alpha bemeneteire köss egy-egy LinearSpread-et, és állítsd az első bemeneteiket 0.5-re! Így az értékek 0 és 1 közé fognak esni. Majd emeld fel a Spread Count-ot 10-re az utolsó bemeneten: így már 10 értéked lesz. Ugyanezt végezd el a másik LinearSpread-en is!



6.4. ábra: HSL LinearSpread

Most a renderablakban egy fehér négyzetet látsz, ami azt jelenti, hogy a Lightness 1-en van: ezt írd át 0.5-re! Ezzel kaptál egy pirosas színt, amit úgy tudsz megváltoztani, ha az Alpha bemenetre kötött LinearSpread modul Phase (eltolás) bemenetét átírod, vagy akár rákötsz egy 3 másodpercre lelassított LFO-t. Ha most generálunk hozzá egy textúrát, rögtön jobban fog látszani, mit csináltunk.

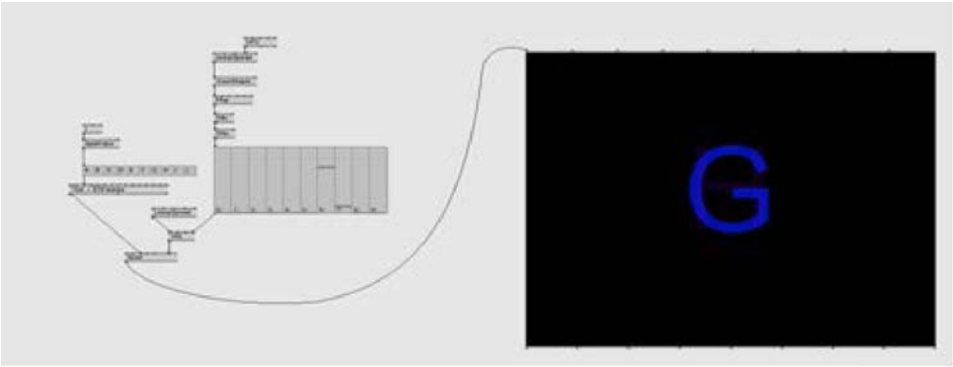
Ehhez hozz létre egy Text Ex9 Texture woei modult, és kösd rá a Quad textúrabemenetére! Ezzel megjelenik egy „VVVV" kiírás a megjelenítőn. Ahhoz, hogy ebből a Text modulból 10 különböző textúraréteget kapjunk, egy i modulra és egy SpellValue-ra lesz szükség. Az i egész számokat hoz létre két adott érték között, a SpellValue pedig ezekből a számokból generál betűket. Kösd össze a kettőt, és az i első két bemenetére írj 65-öt és 75-öt, a SpellValue legördülő menüjében pedig válaszd ki az Ascii funkciót!



6.5. ábra: Text textúra és Spell Value

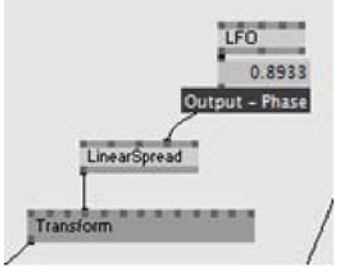
Ekkor már A-tól Z-ig tudod használni a betűket, mivel a karaktertáblázatban 65-től kezdődnek az ábécé betűi. Ha ezt bekötöd a Text megfelelő bemenetére, láthatóvá

válik, hogy mindegyik betű más és más színű. Ahhoz, hogy egyszerre csak két betű látszódjon, átméretezés szükséges: ehhez kösd az Alpha bemenetre kötött LinearSpread-et egy Map-re, a Destination Minimum-ot pedig állítsd át mondjuk mínusz 5-re!



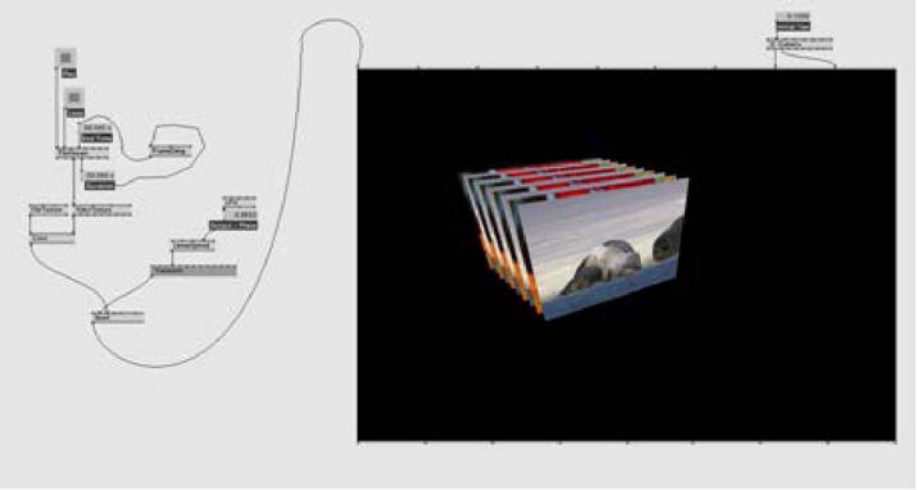
6.6. ábra: Renderelt betűréteg

A következő példában az előzőhöz képest annyi eltérés van, hogy nem HSL-lel van eltolva, hanem térben, viszont az eddigi egyetlen textúraforráshoz képest most egy FileTexture-rel és egy FileStream-mel dolgozunk. Ahhoz, hogy ezeket össze tudd adni, a számokhoz hasonlóan szintén a Cons modulra van szükséged, mivel itt csak két textúránk van, így a Quad most még ötször ismétli a textúrákat ahhoz, hogy a szükséges tíz meglegyen.



6.7. ábra: LinearSpread Transform-mal

Normál esetben a megjelenítő nem ábrázol perspektívát: ehhez a Camera Softimage modul használata szükséges. Nagyon fontos még, hogy ha több réteggel dolgozol, a rendermodulon Inspektorban be kell állítani, hogy a Windowed Deptbuffer Format értéke D24X8 legyen, mivel normál esetben nem megfelelően jeleníti meg a képeket.

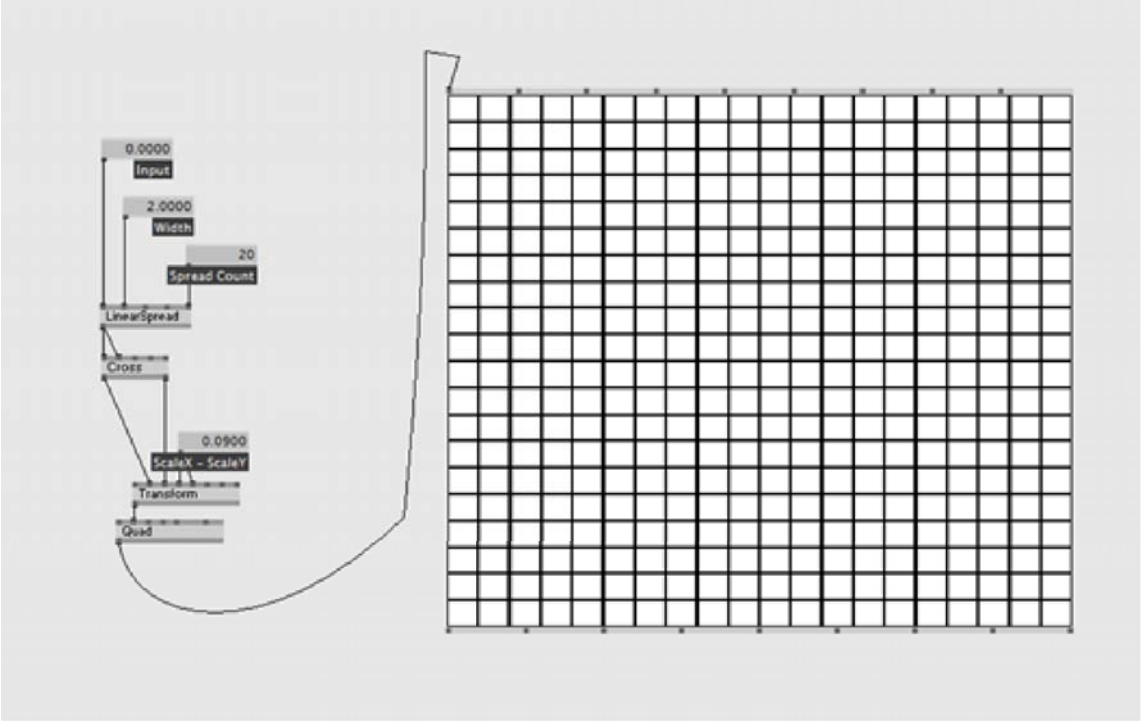


6.8. ábra: Perspektivikus rétegek

Az utolsó példánk egy fokkal bonyolultabb lesz, mint az eddigiek: ebben azt nézzük meg, hogyan lehet az egér távolsága alapján átméretezni objektumokat, és azokat gombnyomásra torzítani.

Első lépésként egy 25x25-ös Quad-hálót kell létrehoznunk a Cross modul segítségével (e nélkül csak átlóban elhelyezkedő négyzeteket kapunk). A Cross nem tesz mást, mint hogy veszi a modul első bemenetének pozícióit és elkezdi ismételni azt a második bemenetének értékeivel, vagyis ha az első bemeneten 5 értékünk, a másodikon pedig 3, akkor egy 5x3-as hálót kapunk.

A Quad-háló létrehozása után a Points2 Vector segítségével tudod meghatározni, hogy az egér melyik négyzethez van a legközelebb. Ehhez a Points2 Vector-modul első két bemenetére kösd be a Cross modul x és y értékét, a második kettőre pedig Mouse modul x és y értékét! Így a harmadik, Length nevű kimeneten kapsz egy számsort, ami megadja a távolságot. Ha ezt bekötöd egy Map-be és a Destination Minimum funkcióhoz 0.1900-et írsz, a Maximum-hoz 0.1200-t, a kimenetet pedig rákötöd a Transform modulod Scale X és Y értékeire, akkor már követi is az egeredet.

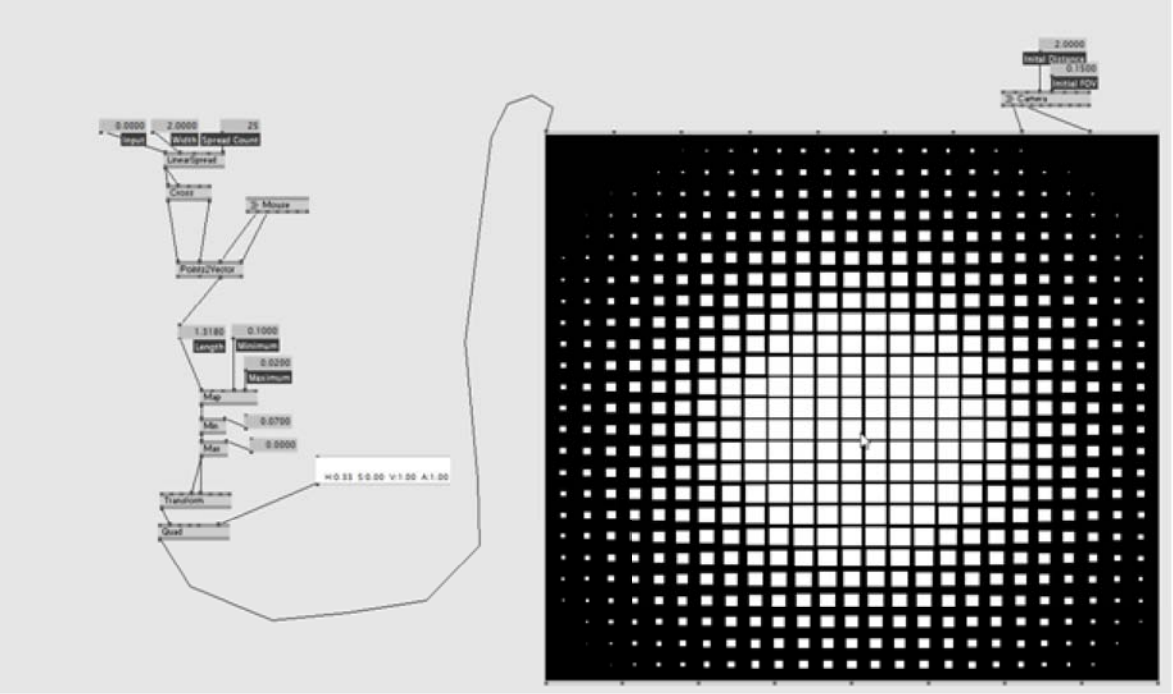


6.9. ábra: A Cross 2D

Ezek után egy nagyon látványos torzításfajta következik: az Attractor 2d. A Points2Vector-hoz hasonlóan ez is az első két bemenetén várja a négyzetek pozícióit, a második kettőn pedig az egérét.

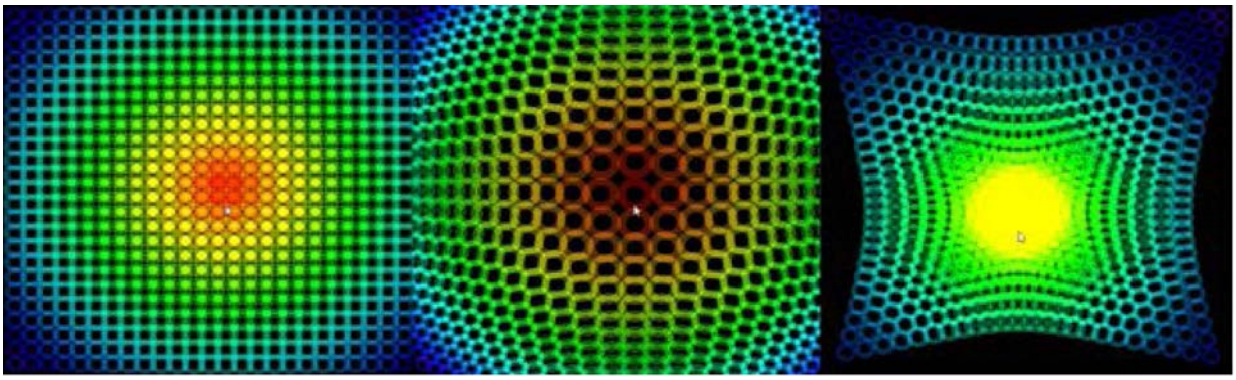
A maradék bemenetek között található a Strength, a Power és a Radius. Ezzel az utóbbival azt adhatod meg, hogy mekkora legyen a torzítás átmérője (ezt állítsd be mondjuk 1.4200-ra), míg az első kettővel azt, hogy milyen erős legyen. Hozz létre egy Or-t és kösd össze a jobb és a bal klikkel!

A jobb klikkre köss egy Map-et, a kimeneti minimumra, vagyis a 0-ra írj mínusz 2-t, a maximumra pedig 2-t! Ennek a kimenetét szorozd meg az Or kimenetével és ezt kösd bele a Strength-be, a Power-be pedig írj 1.290-et!



6.10. ábra: Points2Vector vezérelte Quad-ok

Ezzel létre is jön az effekt, aminek a paramétereit később nyugodtan megváltoztathatod, ha már jobban átlátod az Attractor 2d viselkedését.



6.11. ábra: A végeredmény: Points2Vector és a két Attractor 2D -vel torzította quadok

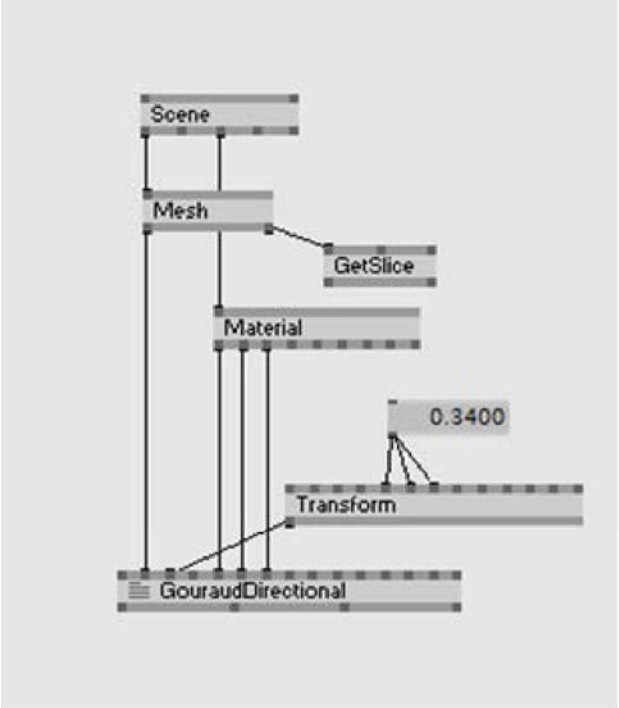
7. Háromdimenziós megjelenítők és azok egyszerűbb editálása

Ebben a leckében a háromdimenziós megjelenítőkkel és azok egyszerűbb editálásával fogunk foglalkozni. Elsőként nézzük végig a leggyakrabban használt 3D-s fájlbetöltőket és a legfontosabb megjelenítőket!

A Scene névre hallgató Assimp fájlbetöltőt más 3D-s szoftverekből is ismerheted. Ez egy nyílt forráskódú 3D-s objektumbetöltő, ami sok különböző fajta 3D-s formátumot kezel, mint amilyenek a .dae, a .blend, a .3ds, a .ase, a .obj, a .stl és még sok más. Ebbe remélhetőleg a közeljövőben integrálják a szokásos 3D-funkciókat, mint a csontozás és az animáció.

A betöltés után mindenképpen szükséged lesz ehhez egy Mesh Assimp modulra, ami a Scene kimenetét Mesh-re alakítja át. Ezenkívül még számos modul van a Scene-hez: ilyen például a Camera, ami az objektumba mentett kamerapozíciót olvassa ki, a Material, ami a színezésért és a textúra adataiért felelős, és a Node, ami pedig az objektum transzformációit adja meg.

A Scene betöltővel kapcsolatban fontos tudni, hogy a vertexeket optimalizálja, és ezt a funkciót sajnos nem lehet kikapcsolni. Ez akkor okozhat gondot, ha például sok, ugyanannyi vertexszel rendelkező formát mentettél ki, és ezek között szeretnél morfolni. Ez ebben az esetben nem fog működni, mert a betöltés után a vertexszámok nem egyeznek majd.

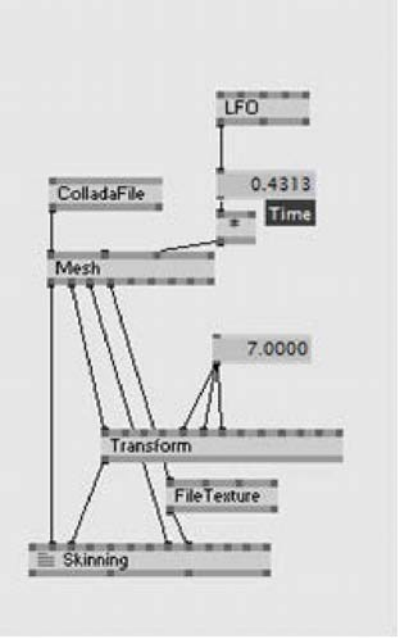


7.1. ábra: A Scene betöltő modul

A másik fájlbetöltő, a ColladaFile csak a .dae formátumot tudja olvasni, ezen belül viszont nagyon sok funkció használatát teszi lehetővé. Ebbe a betöltőmodulba már integrálták az összes olyan dolgot, ami az előzőből hiányzik, így ennek segítségével külső programban lementett mozgást vagy csontozást is tudunk használni, ahogy ez a példafájlban is látható. A csontozatot akár Kinect-tel vagy bármilyen más módszerrel is irányíthatjuk, ez viszont soha nem egyszerű feladat, mert mindegyik metódus más és más transzformációkat igényel, ezeket pedig nem könnyű kiismerni.

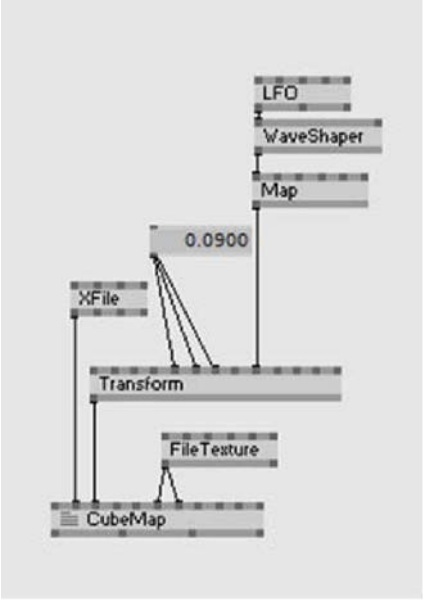
A ColladaFile modul mellé ugyanúgy szükséges Mesh-t használni az átalakításhoz mint az Assimp-nál, ebbe viszont már integrálták a sima Transform-ot, a Skinning transform-ot, a textúra elérési útját és a színezést is, valamint a bemenetei között megtalálható a Time bemenet is, ami az animáció lejátszásához szükséges.

A Mesh modulra aztán elég egy egyszerű háromdimenziós megjelenítőt hozzákötni, mondjuk PhongDirection-t – ez viszont önmagában még nem képes kezelni a csontozás funkciót. Ez utóbbihoz a Skinning modult kell használnod, aminek van Skinning transform bemenete. Ahhoz, hogy a betöltött objektum eredeti pozícióját megkapd, mindenképpen kösd össze a sima Transform kimenetét a megjelenítőd Transform bemenetével! Sőt, a kettő közé ajánlatos még egy 3d Transform-ot is bekötni, hogy aztán az eredeti transzformációt könnyen meg tudd változtatni.



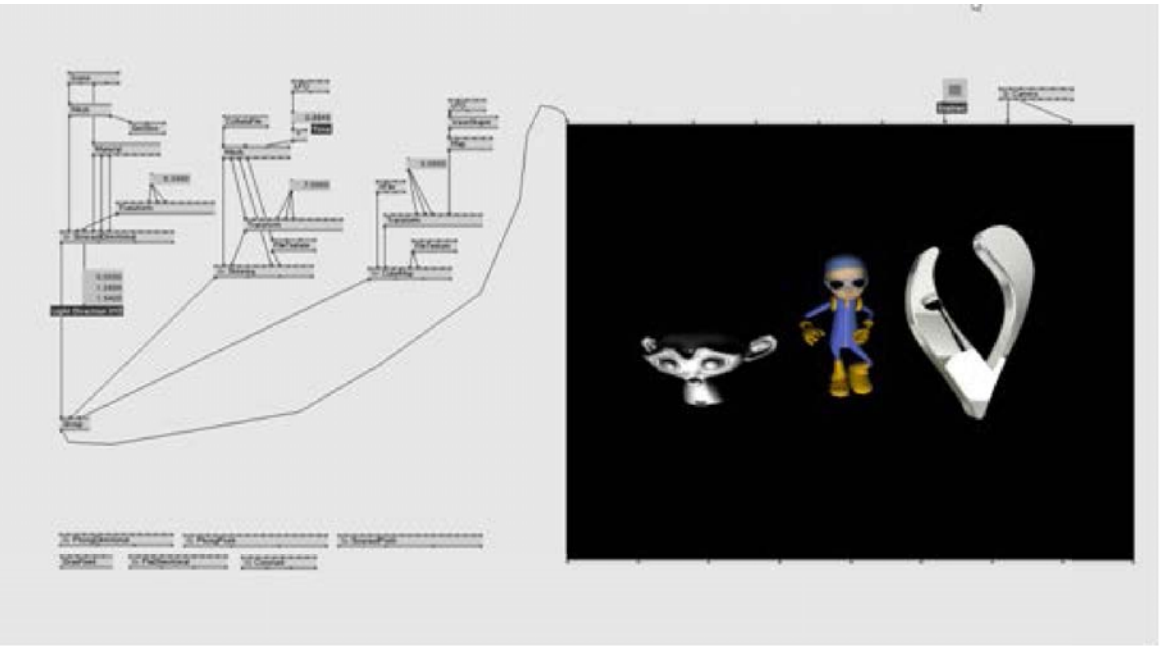
7.2. ábra: A Collada betöltő

A harmadik fajta és egyben legegyszerűbb 3D-s betöltő az XFile, ami .x formátumú fájlokat használ (ez a DirectX 3D-s fileformátuma). Ennek a fájltípusnak se plusz modulja, se Transform kimenete nincsen, viszont nagyon könnyű editálni és újakat létrehozni belőle, mivel a .x fájlok írásához elérhető külön erre a célra készített Writer node.



7.3. ábra: Az Xfile betöltő

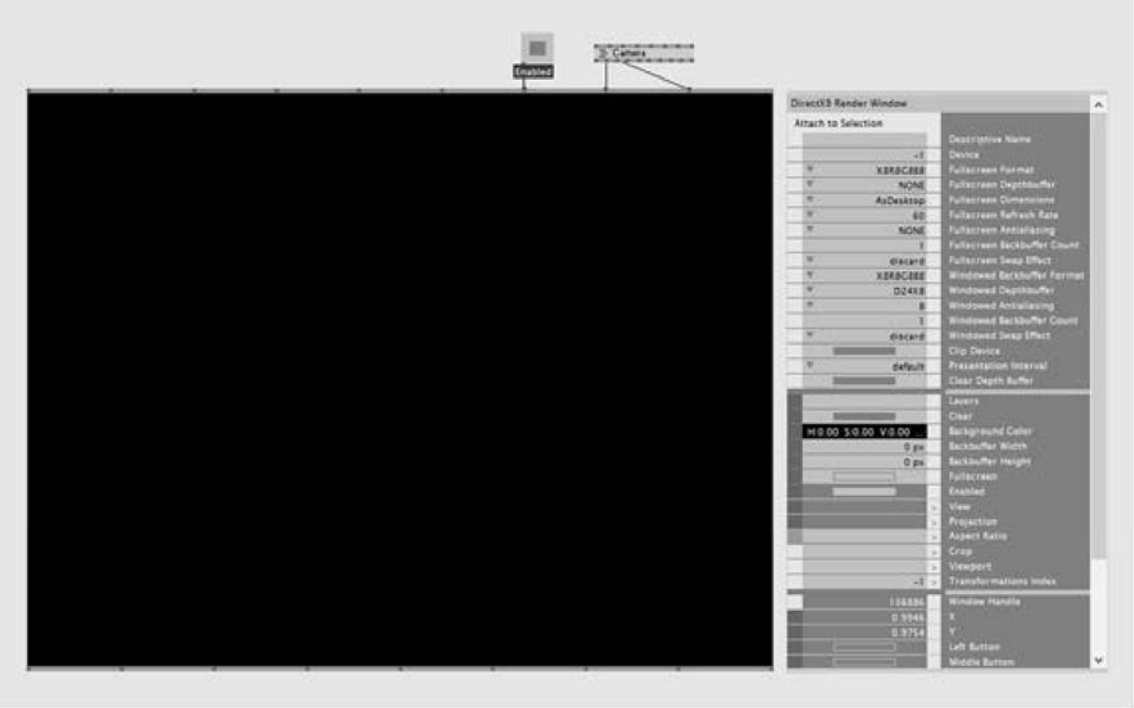
A példafájlban láthatod, hogy ehhez a modulhoz CubeMap texture-t kötöttem: ezzel a modullal tükröződő tárgyakat tudsz létrehozni, csak megfelelő textúra kell hozzá.



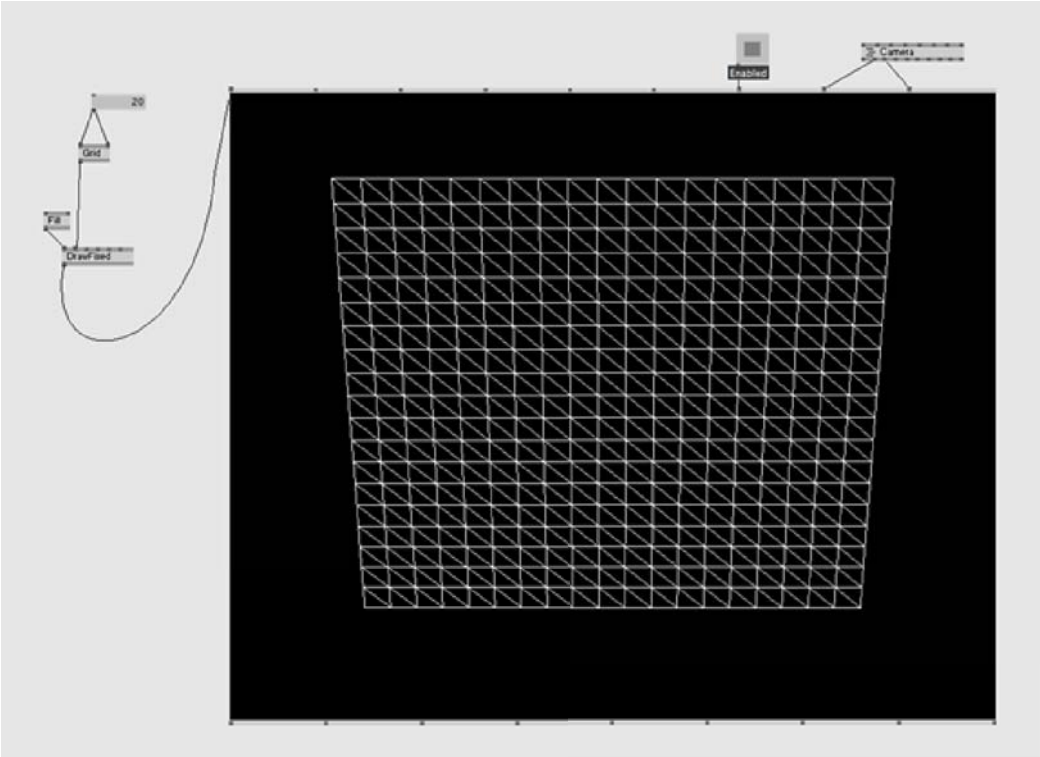
7.4. ábra: A 3d–s megjelenítők

A következő példafájlban egy lap hullámoztatását nézzük meg közelebbről. Ehhez először hozz létre egy DrawFixed modult, egy Renderer-t és egy Camera Softimage modult, és kösd össze őket! A Camera modul Initial Distance bemenetét írd át 1-re. (Itt se felejtsd el beállítani az Inspektorban a DepthBuffer-t D24X8-ra, máskülönben érdekes látványban lesz részed.)

A Mesh-háló létrehozásához válaszd ki a Grid EX9.Geometry modult és kösd össze a DrawFixed-del. A Grid-nek két Resolution bemenete van (X és Y), ezekből mindkettőt változtasd át mondjuk 20-ra. Ha létrehozol egy Fill modult, rákötöd a DrawFixed-re és kiválasztod a jobb bemenetén a WireFrame funkciót, láthatóvá válik a felbontása.



7.5. ábra: A DepthBuffer beállítása

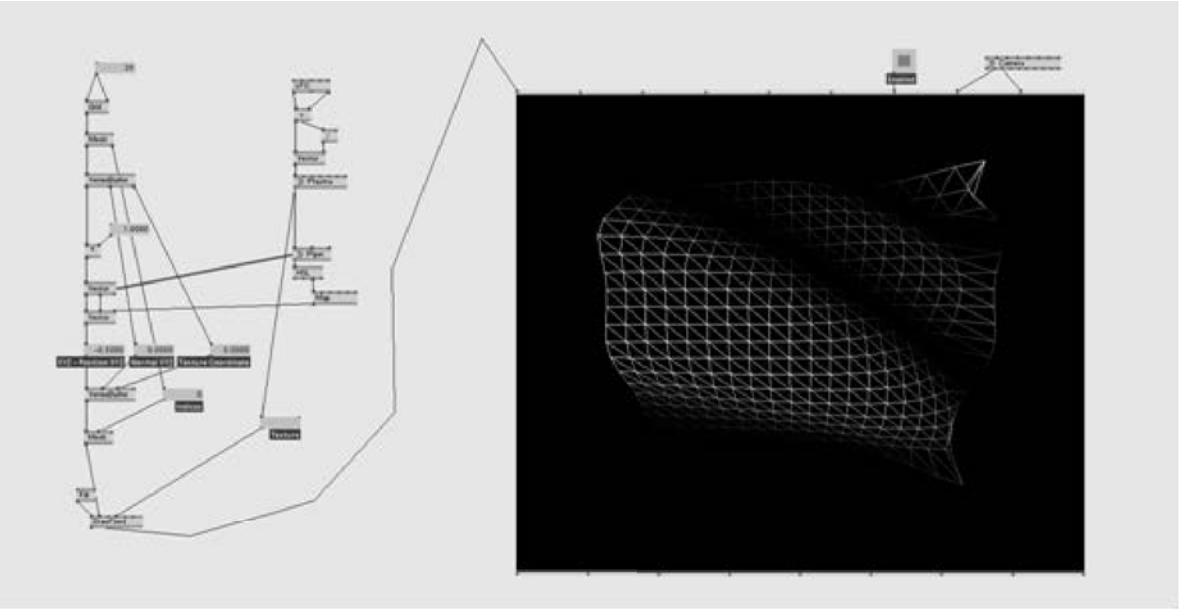


7.6. ábra: A Grid

Minden Mesh-t szét lehet bontani 3D-s pozíciókra: ehhez először hozz létre egy Mesh Geometry Split modult a VertexBuffer Geometry Split modul mellett, és kösd össze mind a hármat! A Mesh modulnak két kimenete van: egy VertexBuffer és egy Indices, a VertexBuffernek pedig alapesetben a Position XYZ és a Normal XYZ. Itt Inspektor segítségével be kell állítanod a Texture Cordinate 0-t 2d-re, mivel a feladathoz a későbbiekben textúra koordináták is fognak kelleni.

Ezek után hozd létre ennek az ellenkezőjét is: a Mesh Joint-ot és a VertexBuffer Joint-ot! Ebből leginkább a Position XYZ fog kelleni a későbbiekben, mivel azt kell majd módosítani, így a többit (a Normal XYZ-t és a textúrakoordinátákat) össze is kötheted a VertexBuffer-ek között, ahogy a Mesh-ek indexeit is. A pozíciókat Vector 3d Split-tel és Joint-tal tudod szétszedni és összerakni; az X és Y értékeket pedig itt is összekötheted, így a Mesh Joint máris rákötheto a DrawFixed-re.

Most az a célunk, hogy a Z pozíciókat változtassuk: ezt akár Random spread-del is megtehetnénk, de ez alkalommal inkább egy képet használunk erre a célra, amiből a színinformációkat Pipet Texture Simple modullal nyerhetjük ki. Ennek az eszköznek három bemenete van: egy textúra, ahová a feldolgozásra szánt képet kell betölteni, és egy X és Y koordináta. Textúraként most használd a Plasma modult, ami egy olyan plazmát generál, aminek a színtelítettségét, méretét és sűrűségét is be tudod állítani (a szaturációt közben húzd le 0-ra). Ezt kösd össze a Pipet-tel, a másik két bemenetre pedig kösd rá Vector Split modul X és Y értékeit, ami így a textúrából lekérdezi majd a Grid modul X és Y pozícióinak megfeleltetett színeket.



7.7. ábra: Pipet torzította Grid

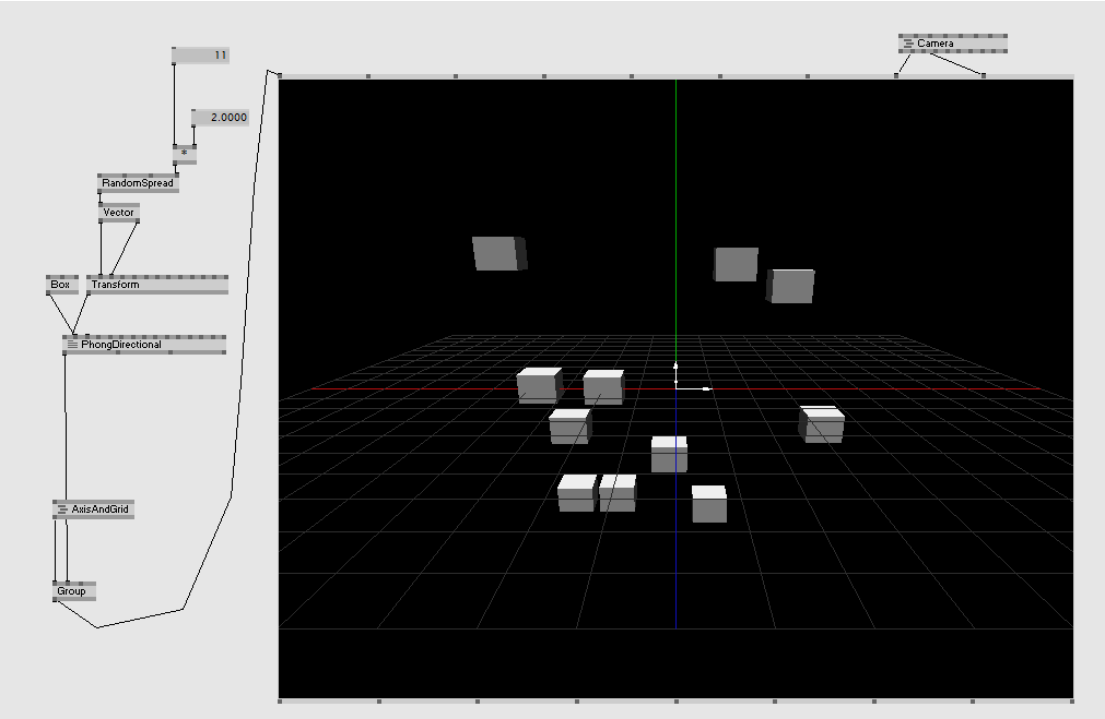
Ha mindez megvan, a Pipet kimenetén kapsz 400 színt, vagyis 20x20-at, amit egy HSL Color Splitbe bekötve megkapod a fényerősséget – ezzel már ki tudod emelni belőle a Z koordinátákat. (A kapott adatokat ajánlott Map-el átméretezni, mert nagyon nagy lesz az eltérés 0 és 1 között, ezért a kimeneti minimumra írj be mondjuk 0.2500-t, a kimeneti maximumra pedig 0-át. A kimenetet azért kell inverzbe tenni, mert különben a képen a sötét értékek fognak kiemelkedni, a fehérebbek pedig besüllyedni, ami ellentétes a hullámzás látványával.)

A kapott értéket már csak össze kell kötni a Vector Joint modul Z bemenetével, és már látható is a renderablakban, ahogy emelkedik.

A plazmát úgy tudod a legegyszerűbben megmozgatni, ha egy LFO első és utolsó kimenetét összeadod, míg az időzítését átírod 10 másodpercre. Ezt egy 2d Vector-ba kötve és továbbkötve a Plasma modul Offset bemenetébe máris megindul a mozgás.

8. Transzformációk és csoportok kezelése

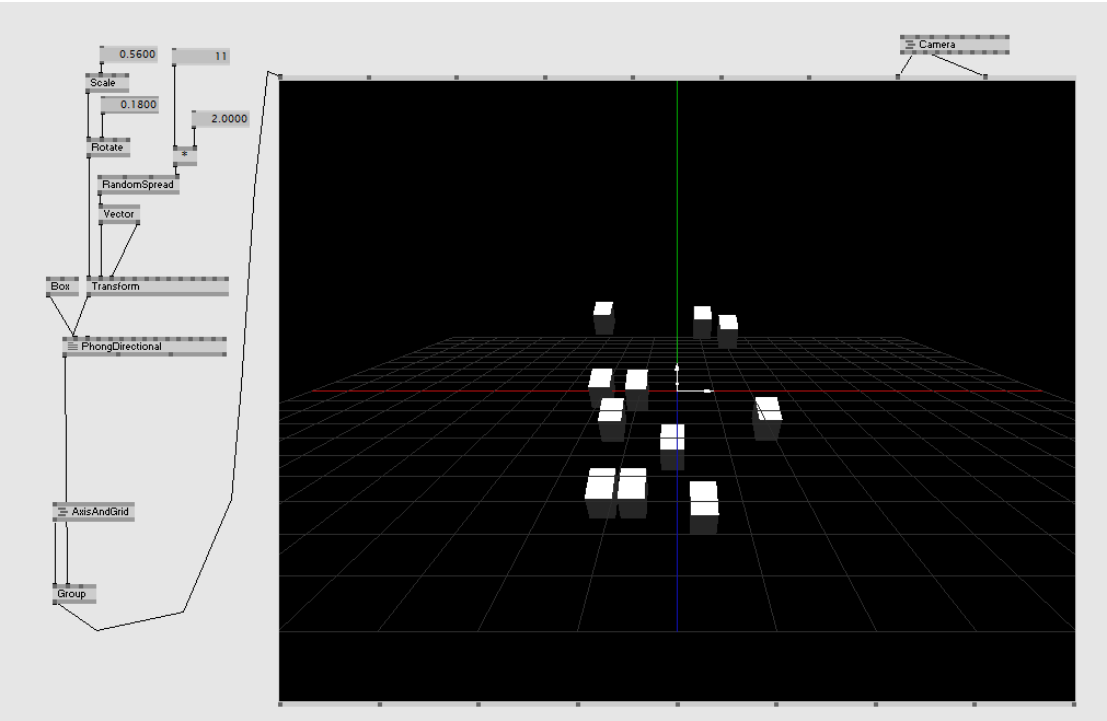
Eddig tapasztalhattad, hogy mindegyik transform-ot egymásba kötheted, így az első réteg transform-mal meg tudod határozni az adott objektum pozícióját, a fölé kötött transform-mal, pedig magát az egészet tudod módosítani, vagyis alárendeltje lesz az alatta lévő objektum. Számos esetben ez megoldás lehet, viszont vannak olyan helyzetek, ahol sokszorozni kell az adott objektumainkat, vagy csak egy adott részüket kell módosítani, de lehet, hogy ez a megoldás sem segíthet.



8.1. ábra: Normál Transform

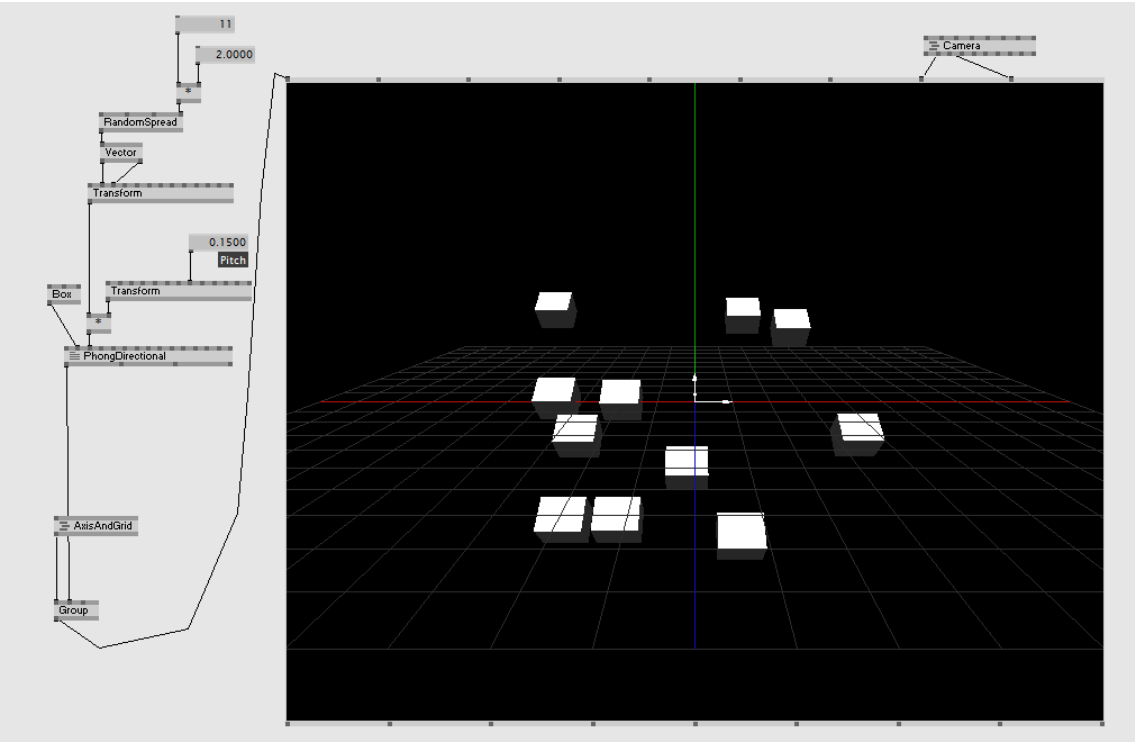
Először is hozz létre egy Rendert, a Grouppal az első bemenetére a groupnak köss egy AxisAndGridet, míg a másikba pedig egy PhongDirectionalt. Ennek második bemenetébe köss egy Box EX9.geometry-t, és a transformba pedig egy 3d

Transformot. Hozz létre egy Camera (Transform Softimage)-t és kösd a Rendernek a View és Projection bemeneteibe. A RandomSpread segítségével hozz létre az x és az y koordinátán 10 pozíciót, így már láthatók a térben is a dobozaid, ha szükséges közelíts rá, vagy változtass a Random szélességén, hogy jól elkülöníthető legyen egymástól mindegyik elem. Továbbiakban, ha rákötsz egy Transform 3d-t és elkezded forgatni az elemeidet, láthatod, hogy egyszerre mozgatja az elemeket. Maga a Transform helyett használhatsz még Rotate, Scale, UniformScale, Translate modulokat is, de így csak a számodra szükséges transzformációt tudod használni.

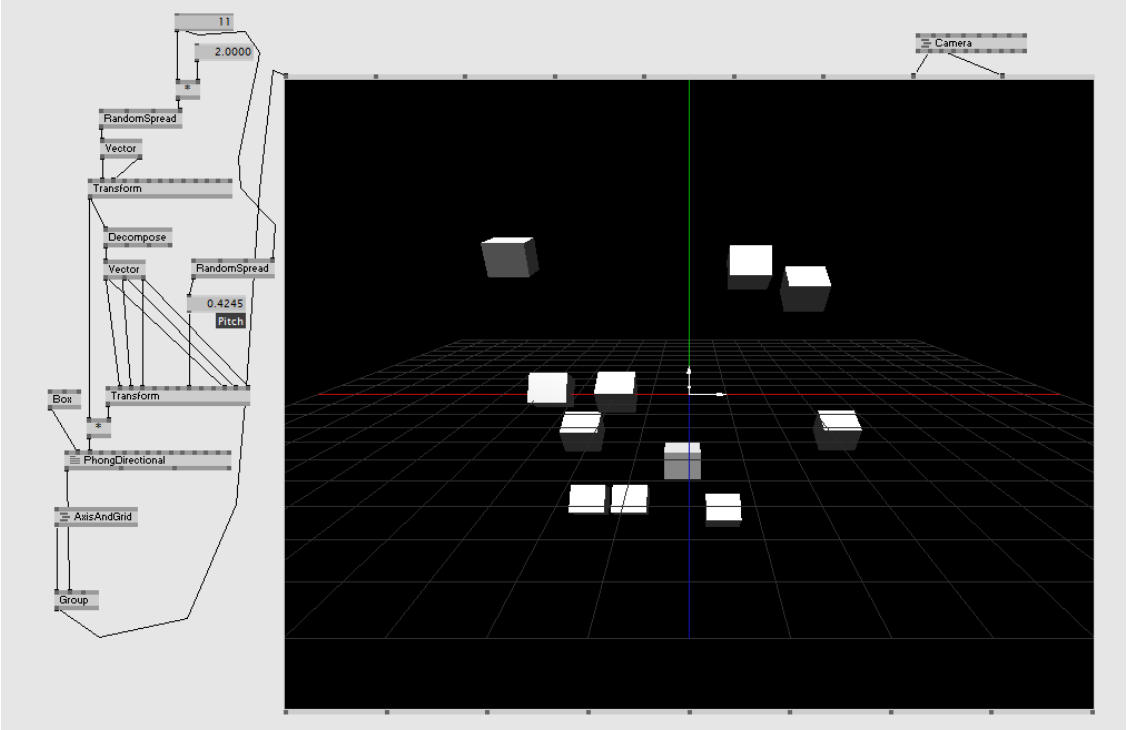


8.2. ábra: Többrétegű Transform

Van egy másik megoldás is, hogy az eredeti Transform-ot tudd egyszerre mozgatni. Ez a következő: ha az eredeti Transform és a Phong közé kötsz egy „* (Transform)”-ot és annak a második bemenetét szorzod egy újabb Transform-mal. A szorzásnál van arra is lehetőséged, hogy egyesével forgasd és mozgasd az objektumokat. Az eredeti Transform-ból Decompose Vector modul segítségével az összes transzformációt ki tudod nyerni. Például mindegyik objektumodat egyesével saját tengelye körül szeretnénk elforgatni, akkor egyszerűen az X Y Z koordinátát be kell kötnünk a transformunk X Y Z bemenetére és a Center X Y Z-re is. Így az összes objektum megkapta az eredeti origóját. Mindezek után lehetőséged van arra, hogy vagy SetSlice-al, vagy akár RandomSpread-el változtasd az elforgatásokat.

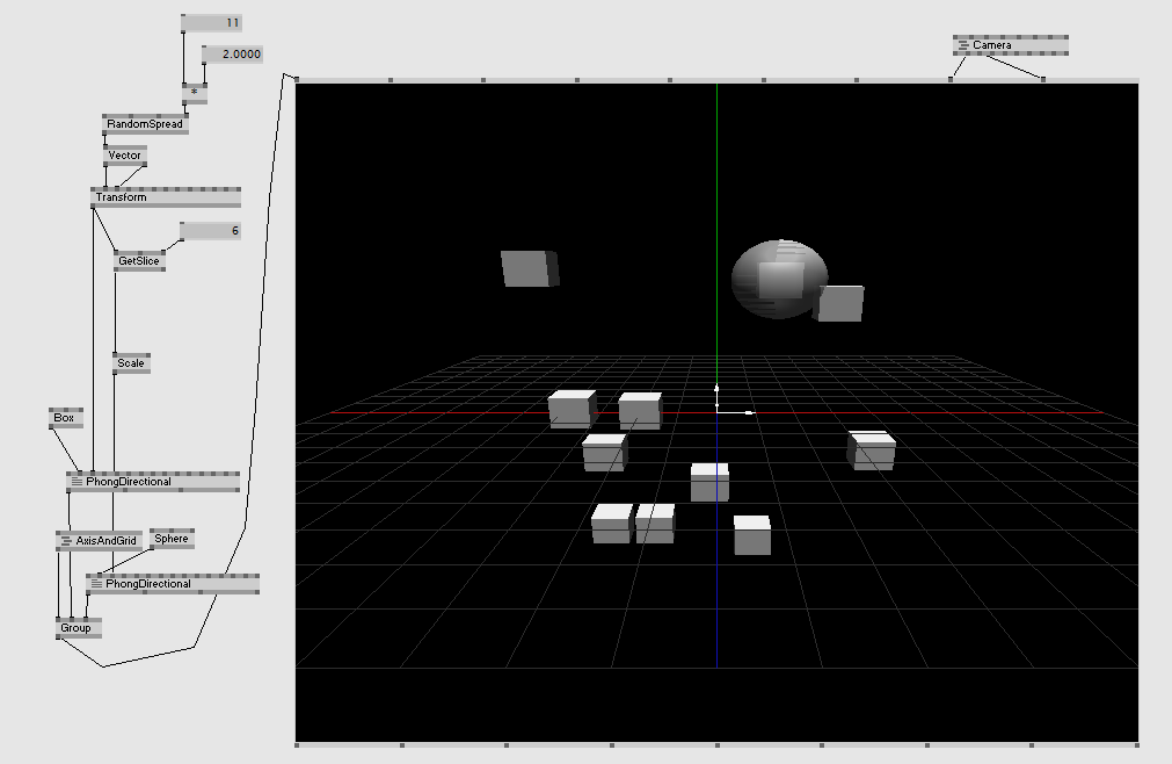


8.3. ábra: Transform szorzása



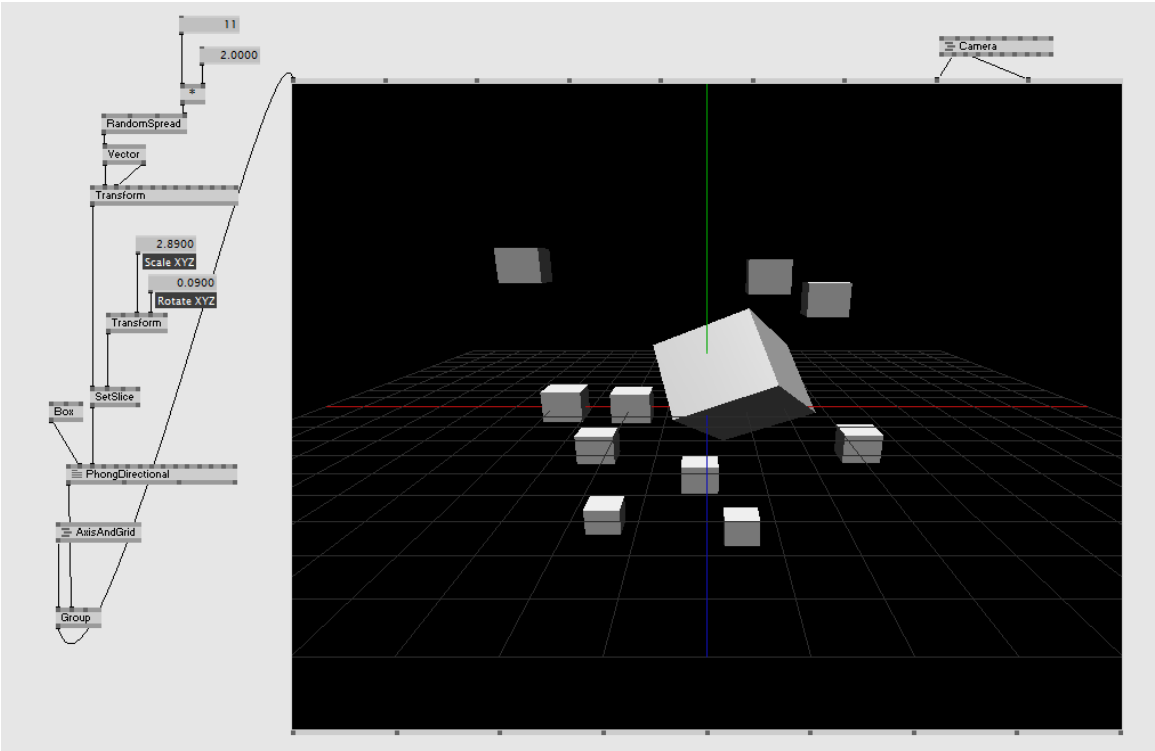
8.4. ábra: Decompose vector

A GetSlice (Node) modullal kiválaszthatjuk egy több Transform-ból álló lista bizonyos elemeit. Három bemenettel rendelkezik az első a Transform bemenet, a második a Bin Size, amiben beállíthatjuk, hogy egy kijelölés hány Transform-ra legyen értelmezve, és végül az Index, ami a kijelölésre szánt elemeket jelenti. Ezt, ha egy Decompose modulba kötjük, akkor már meg is kaptuk a pontos adatait a kiválasztott transformok-nak.

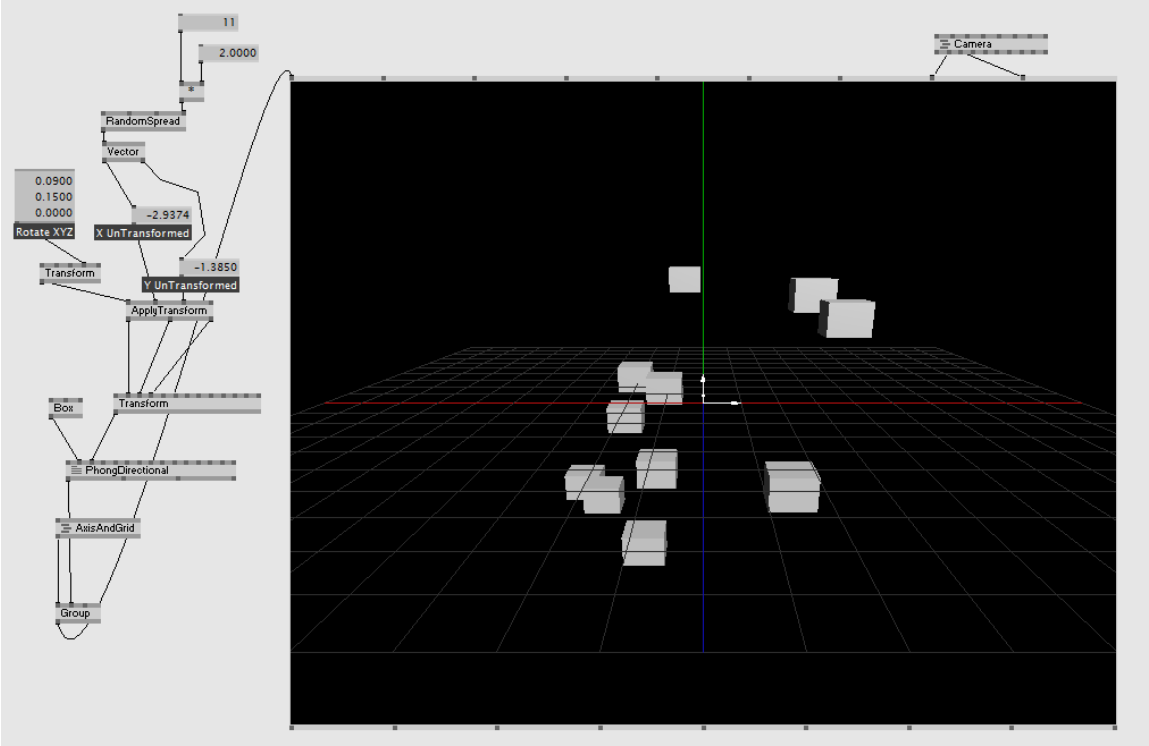


8.5. ábra: GetSlice Transform

Lehetőség van arra, hogy SetSlice (Transform)-ot az eredeti Transform és a Phong közé kötünk, akkor teljesen új pozíciót adhatunk meg egyesével az objektumainknak, mivel ennek a modulnak két Transform bemenete van, az első a forrás a második az új pozíció, ezen kívül még van egy Bin Size és egy Index bemenete is, ahol be lehet állítani, hogy mely elemekre legyen alkalmazva az új transzformáció.

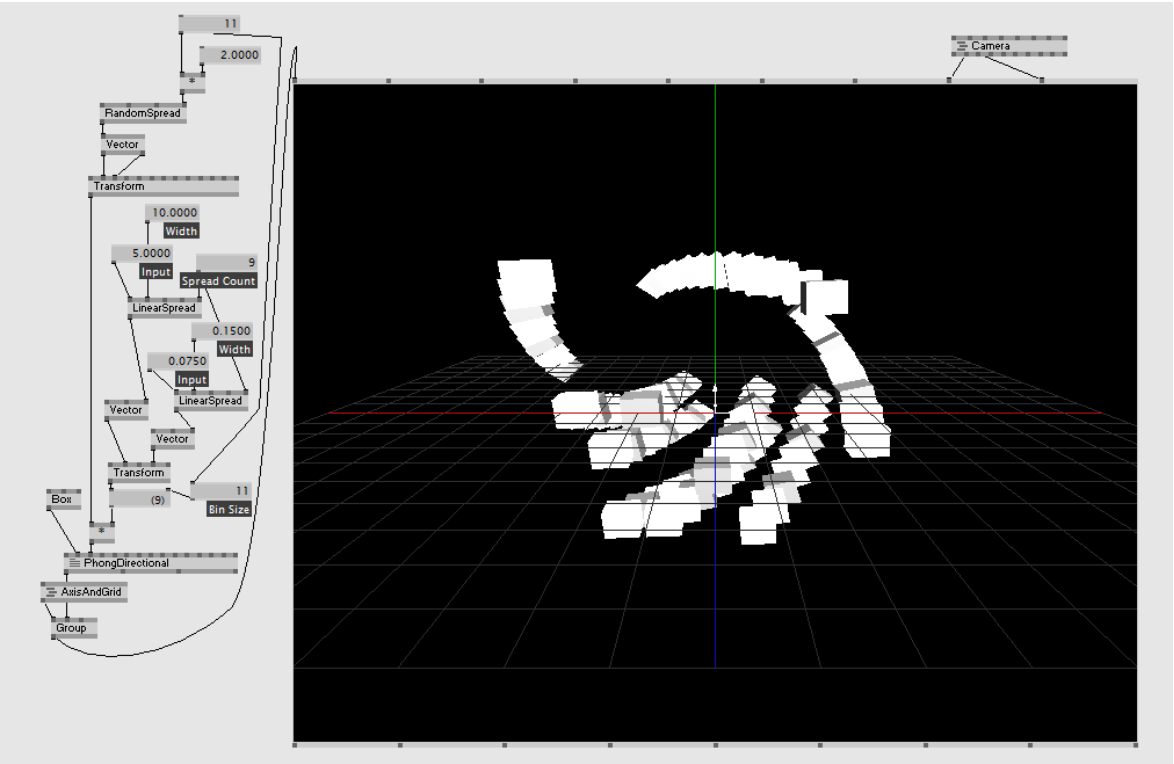


8.6. ábra: SetSlice Transform



8.7. ábra: Apply Transform

Egy másik módja az elemek pozicionálásának az ApplyTransform. A modulba közvetlen kell bekötni az X Y és Z koordinátákat, és az első bemenetébe pedig egy Transform-ot, így tudjuk elmozdítani a objektumunkat. Végül az X Y és Z kimeneten megkapjuk, amit egy alap Transform 3d-be kell kötni. Ebben az esetben a modul kiszámolja az elforgatást, és el is mozdítja az objektumot a térben, de láthatjuk viszont, hogy nem forgatja el úgy, mint az eddigi esetekben. Ezt a modult például arra lehet használni, hogy a Vertex pozíciókat elforgassuk, vagy átpozicionáljuk.



8.8. ábra: Transform többszörözése

Van olyan helyzet, amikor több elemből álló Transform-ot kell sokszorozni (ez az erre kötött transform-mal sajnos nem működik), ehhez létre kell hozni egy Node üzenet típust. A bal oldali bemenetébe kell kötni a sokszorozáshoz használt Transform-ot, és a jobb bemenetébe pedig a Bin Size-ba az eredeti SpreadCount-ját, vagyis azt a számot, amennyi elemből áll. Végül nincs más teendőnk, az előző esetekben használt szorzásba kell bekötni, így már többszörözni fogja az objektumokat.

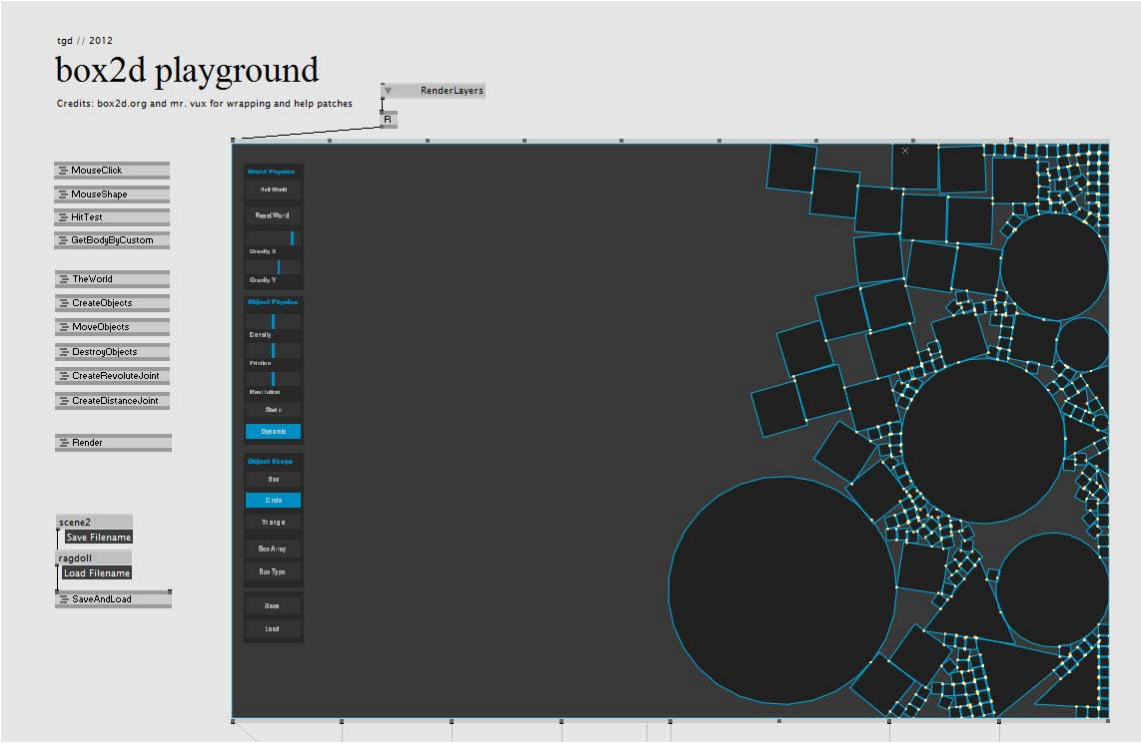
Most pedig beszéljünk a fizikáról röviden. Háromféle fizikai engine van beépítve a VVVV-be:

- a Box2d http://en.wikipedia.org/wiki/Box2D,

• az ODE http://ode-wiki.org/wiki/index.php?title=Main_Page,
• és a Bullete http://en.wikipedia.org/wiki/Bullet_(software),

illetve a felsoroltakon kívül van még Particle alapú fizika is, mint például a Particles GPU Shader Library (http://vvvv.org/contribution/particlesgpu-shader-library) és a CiantParticles - 16000000p GPU Particle System (http://vvvv.org/contribution/ciantparticles-16000000p-gpu-particle-system-%20-rotations), melyek GPU alapú Particle system-ek, vagyis a pozíciókat már nem a proceszorok számítják ki, így akár százezernél is több pontot tudsz egyszerre generálni, de most csak a Box2d-re és a Bullet-re térnék ki.

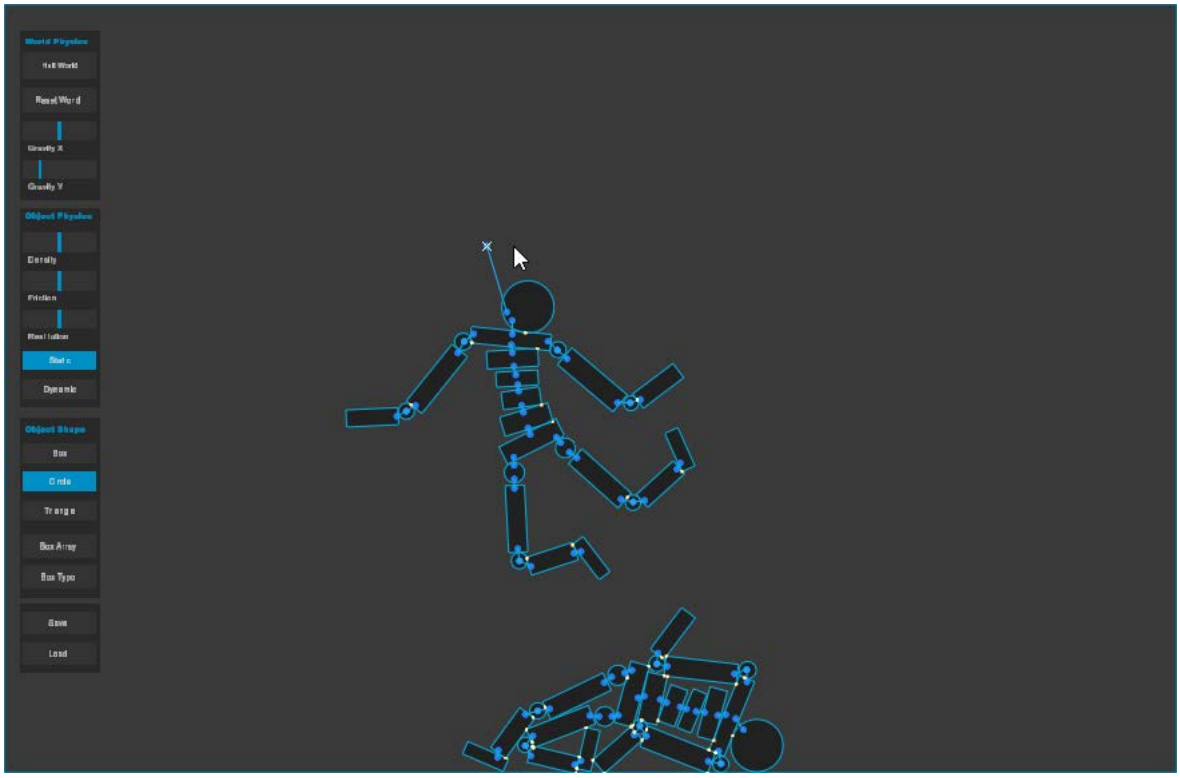
Először vegyük a Box2d-t, ezzel az Engine-vel már találkoztál különféle PC-s, illetve telefonos játékokban (többnyire motoros és tankos játékokban használták). A VVVV- ben is integrálva lett, és ebből egy nagyon jól használható kezelőfelülettel külön letölthető kiegészítőt készített TGD, mégpedig a box2d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



8.9. ábra: BOX2D kezelőfelülete

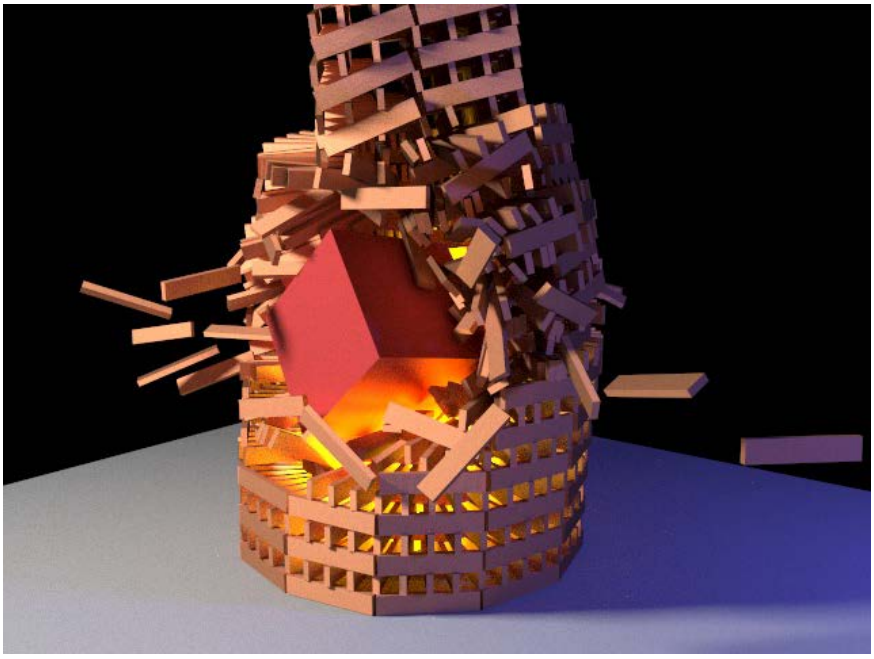
Ha elindítod a patchet, akkor láthatod, hogy a kezelőfelület DirectX renderben jelenik meg. Létre tudsz hozni statikus és dinamikus tárgyakat, ilyenek például a négyszög, háromszög. Előre elkészített elemeket is létrehozhatsz, mint például a négyzethálót és

a szöveg mintára felsorakoztatott dobozokat. Ezeket a beállításokat tudod változtatni, csak be kell menned a CreateObjects node-ba, és azon bellül is a CreateBoxTypo-ba, vagy CreateBoxArray-be. Lehetőséged van mentésre és betöltésre is, ha indítás után ráklikkelsz a Load gombra, akkor alapesetben egy „csontváz” fog betöltődni. E példán látható, hogy kapcsolódási pontok vannak bizonyos elemek között. Ezt úgy tudod leggyorsabban létrehozni, hogy két elemet egymásra húzol Halt World módban, és ráklikkelsz jobbal arra a pontra, ahova a csomópontot szeretnéd elhelyezni.

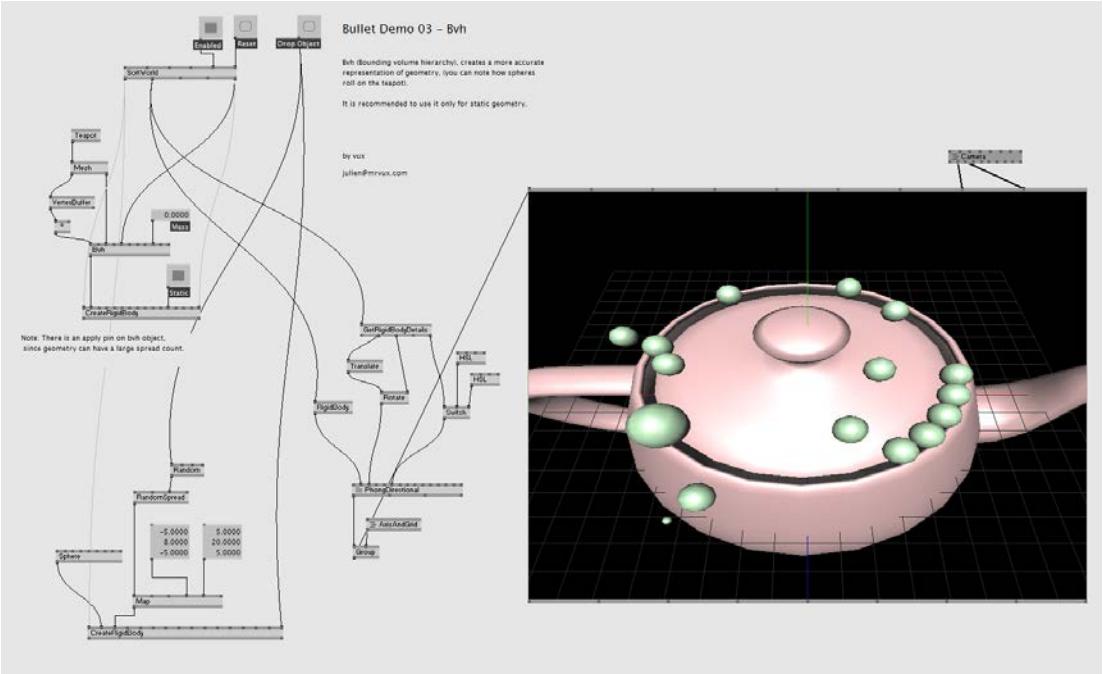


8.10. ábra: BOX2D mentés betöltése

A Bullet fizika az addonpackkal érkezik, a bemásolás után az \addonpack\girlpower\BulletPhysics\ könyvtárban találhatod a példaállományokat. A bullet már egy sokkal komplexebb 3d-s fizikai Engine, amit számos neves 3d-s cég alkalmaz is fizikai szimulációra. Láthatod, ha megnyitod a példaállományokat, hogy működésük kicsit más, mivel úgy kommunikálnak egymással a modulok, hogy nincsenek összekötve, de még a Send és Receive modul sem. CreateRigidBody és CreateSoftBody-val van lehetőséged arra, hogy létrehozz tárgyakat a térben, melyek statikusak vagy dinamikusak lehetnek. Továbbá ezek a tárgyak lehetnek egyszerű geometrikus formák, melyekhez már kész Bullet-modul is van, de akár saját tárgyat is betölthetsz, mert a BVH nevű modul képes átalakítani a Mesh fájlt a számára megfelelőre.



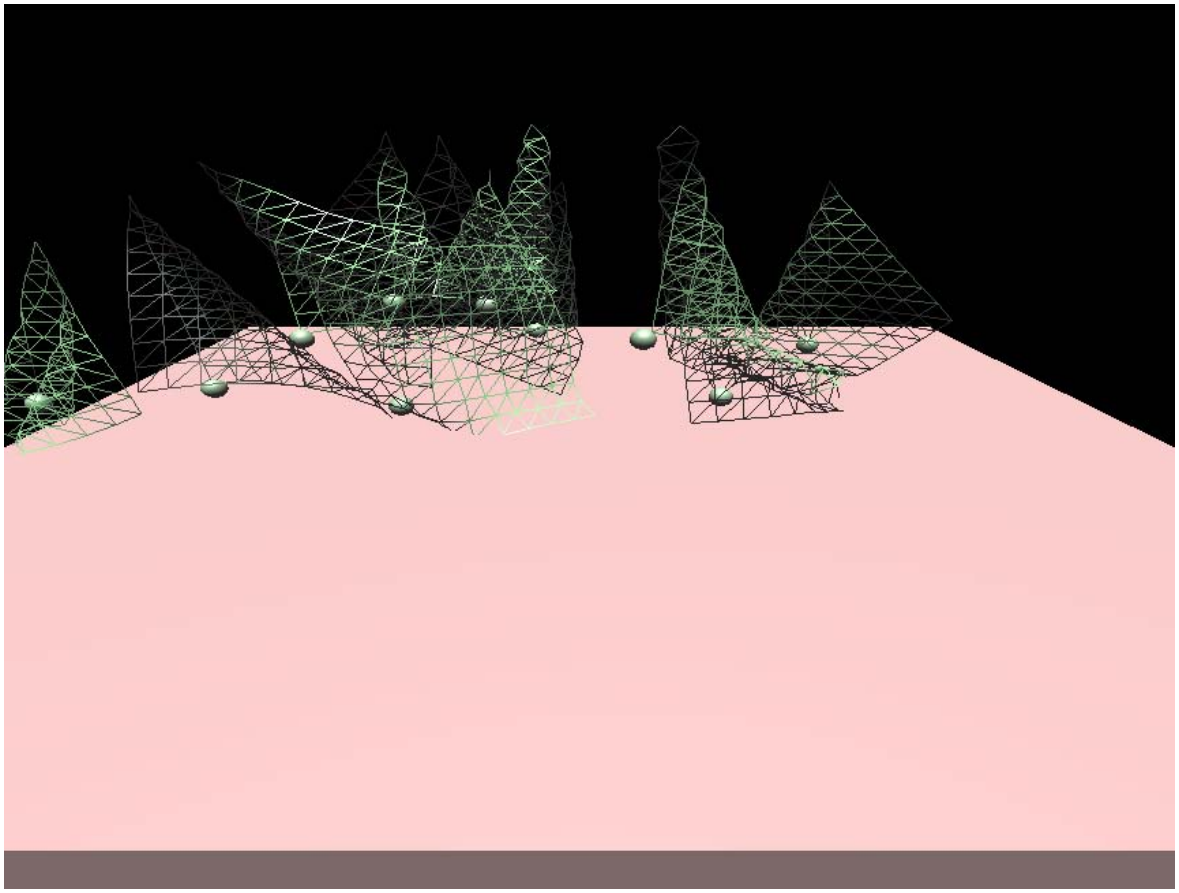
8.11. ábra: Bullet az OpenCL-ben



8.12. ábra: Bullet BVH Mesh betöltése

A SoftWorld-del hozod létre ezt a fizikát, ebből a modulból tudod kinyerni azokat az információkat, amivel a végső Renderben már vezérelni tudod a modulokat. A RigidBody-val kapod meg a SoftWorld-ből a mesh formákat, míg a

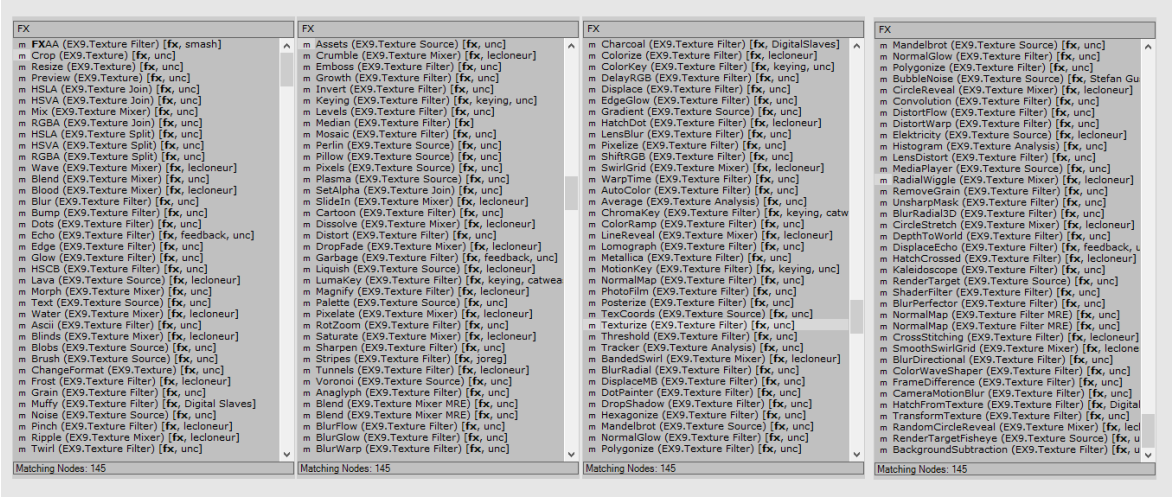
GetRigidBodyDetails-szel a transzformációkat, amit már be tudsz kötni egy PhongDirectional-ba.



8.13. ábra: Soft rigid

9. A filterek és a beépülő modulok

Már korábban telepítetted az Add-on Pack-ot, a VVVV-d az alapfunkciókon túl már rendelkezik számos új, hasznos pluginnal és effekttel, amiket a következőkben fogsz megismerni.

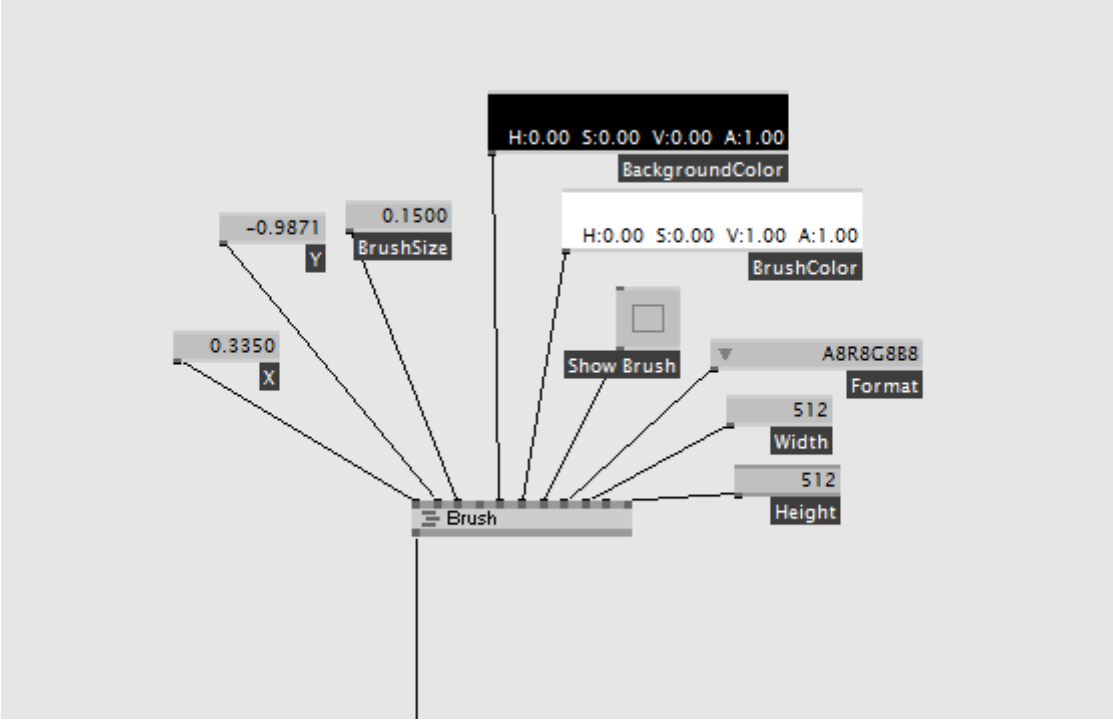


9.1. ábra: Effect lista

Beírod a node keresőbe, hogy „FX”, láthatóvá válik az elérhető effektek listája. Ezeknek a nagy részét az UNC készítette, és a legtöbbet – például a különböző lyukasztásokat, elmosásokat, torzításokat, színkorrekciókat és hasonlókat – már ismerheted különböző kép- és videószerkesztő szoftverekből.

Első lépésként egy jórészt kiegészítőkből álló rajzoló patch elkészítését vesszük végig, amit a későbbiekben maszkoláshoz és rajzoláshoz egyaránt használhatsz. Maga a patch egy az ecset paramétereiért felelős Brush eszközből, egy a képek összeadásáért felelős Blend-ből, és egy ideiglenes tárolóból, vagyis Buffer-ből fog állni.

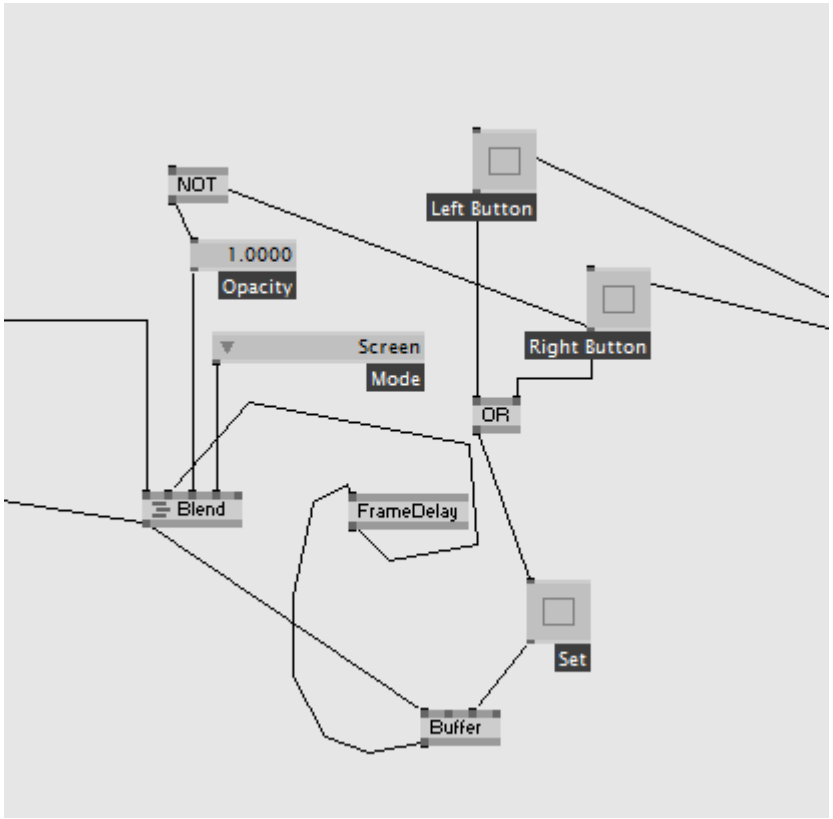
Először hozz létre egy Render-t és köss bele egy Group-ot. A Group első bemenetére köss egy FullScreenQuad-ot, míg a másikra egy Cursor-t. Majd hozz létre egy MouseState Joint-ot és kösd a Cursor-ba. A MouseState X és Y bemenetét a bal egérgombbal együtt kösd bele a Render azonos kimenetébe, hogy a Cursor az egered mozgását kövesse. Ezt egyelőre azért nem látod, mert a FullscreenQuad textúrabemenetére még nincs rákötve semmi, így az az egész képernyőt fehér színnel tölti ki.



9.2. ábra: Brush plugin

Ennek áthidalásához ideiglenesen köss bele egy újonnan létrehozott Brush-t. Most ha megnézed ennek a bemeneteit, látható, hogy rendelkezik X és Y pozíció, ecsetméret és ecseterősség bemenetekkel is. Kösd a Render modul X Y pozícióját az annak megfelelő Brush-bemenetbe, a méretét pedig állítsd át, mondjuk 0.1500-re. Most hozz létre egy Blend EX9.Texture Mixer-t, és az első bemenetére kösd rá a Brush modult. Ahhoz, hogy a képet a program összeadja, a második bemenetbe saját magát kell visszakötni, ami csak egy FrameDelay EX9 Texture-n keresztül lehetséges. Ha ezt most belekötöd a FullscreenQuad-ba, valószínűleg semmit nem fogsz látni, ezért állítsd át a Blend Mode-ot Screen-re, az Opacity-t pedig húzd le 0-ra, majd vissza 1-re. Így már folyamatosan tudsz rajzolni.

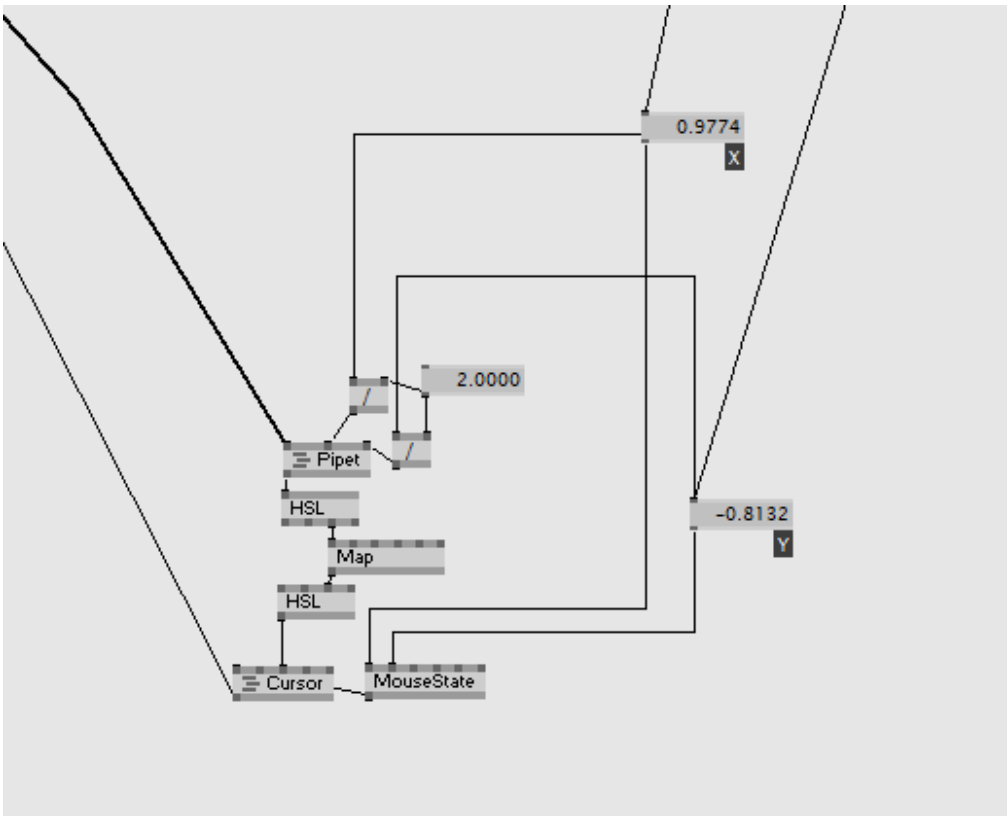
A kép elsőre furcsának tűnhet, ez a modul ugyanis alapesetben felhőszerűen rajzol, vagyis minden ecsetvonáshoz hozzáad egy halvány fekete kontúrt is. Ezt a Brush modulban lehet korrigálni, mégpedig úgy, hogy a Background Color bemenet alfáját felhúzod 1-re.



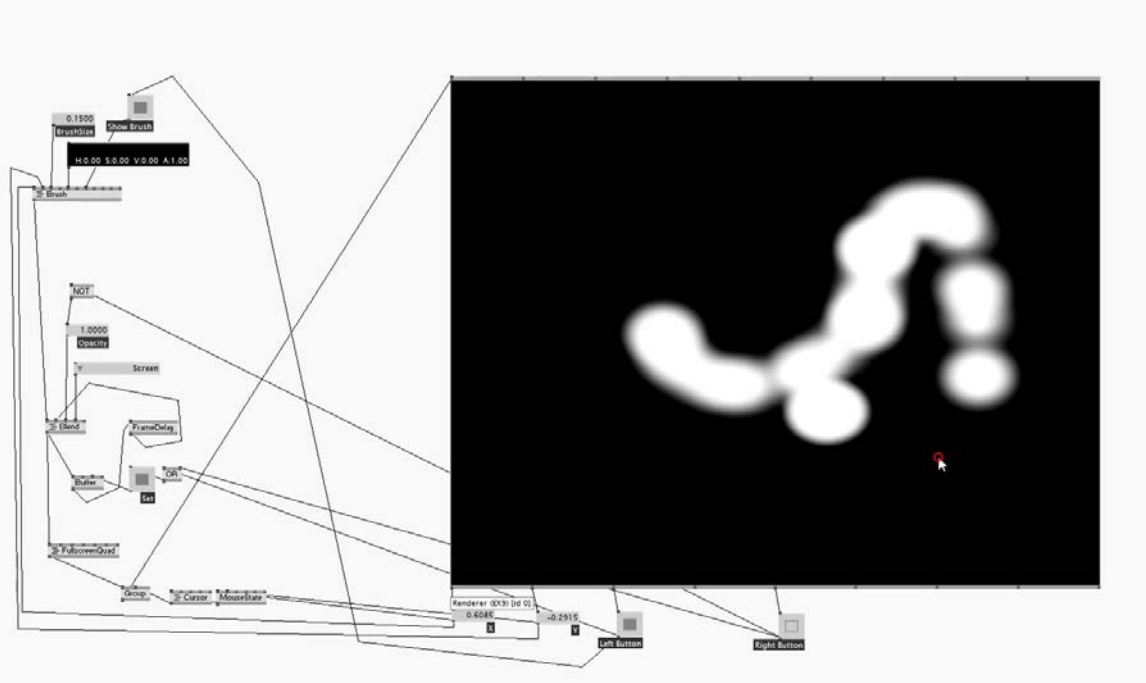
9.3. ábra: Blend és a Buffer bekötése

Ezek után köss egy Buffer EX9.Texture modult a Blend és a FrameDelay közé, a Buffer modul Set bemenetére pedig kösd Or-on keresztül a Render balklikkes egérkimenetét. Így ha a Blend Opacity bemenetére egy NOT kapun keresztül bekötöd a jobb egérgombot, a program a jobb gombra reagálva törölni fogja a képedet, bal gombos klikkel pedig rajzolni és tárolni fogsz tudni.

Fontos, hogy törléskor teljesen fekete képet kapj, ezért ilyenkor mindenképpen el kell tüntetned a Brush-t. Ehhez a Show Brush bemenetébe kell bekötnöd a bal egérgombot: ezzel azt éred el, hogy a program csak akkor mutassa az ecsetet, amikor nyomva tartod az egér gombját. Ahhoz, hogy az egered mindig inverz színű legyen a mögötte lévő képhez képest, egy PipetSimple-t kell létrehoznod, a Blend kimenetét pedig hozzákötni a Pipet-hez. Ez a modul a másik két bemenetén szintén az egér pozícióit várja, csak nem -1 től +1-ig, hanem -0.5-től 0.5-ig, ezért itt a Render modul X és Y kimenetét el kell osztani kettővel, majd azt belekötni a Pipet-be. Ez a patch továbbfejleszthető még egy Tablet modullal is, ami Wacom rajzolótábla használata esetén nyomásérzékennyé tudja tenni az ecseted átmérőjét.

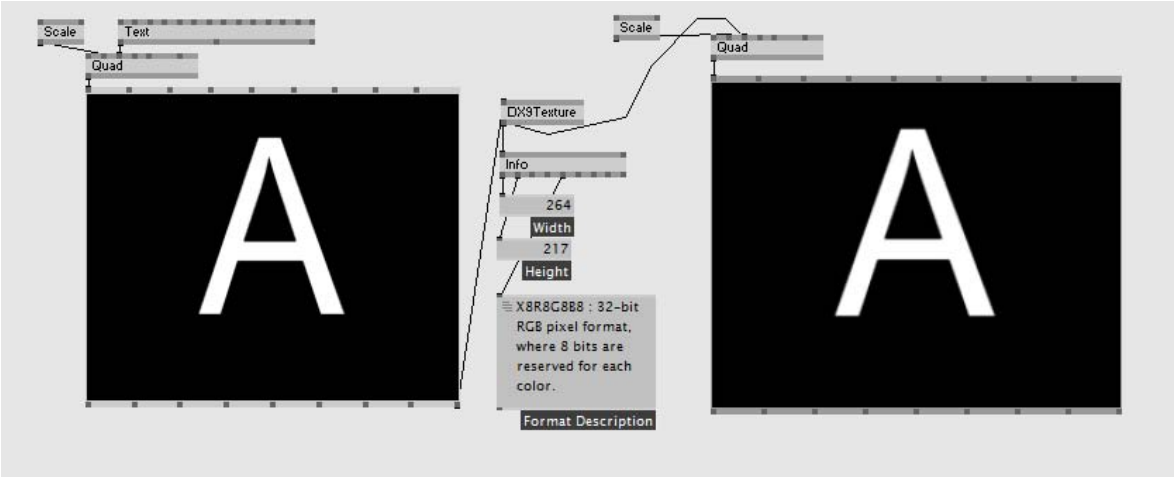


9.4. ábra: Cursor és a Pipet bekötése



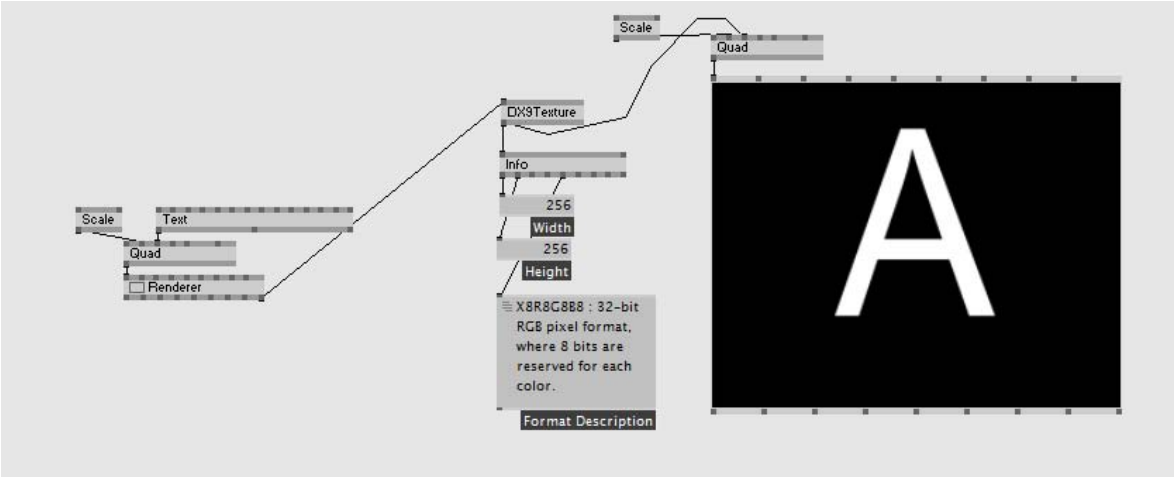
9.5. ábra: Brush patch

A következőben azt nézzük meg, hogy egy kész Render patch hogyan effektezhető.



9.6. ábra: DX9 Texture

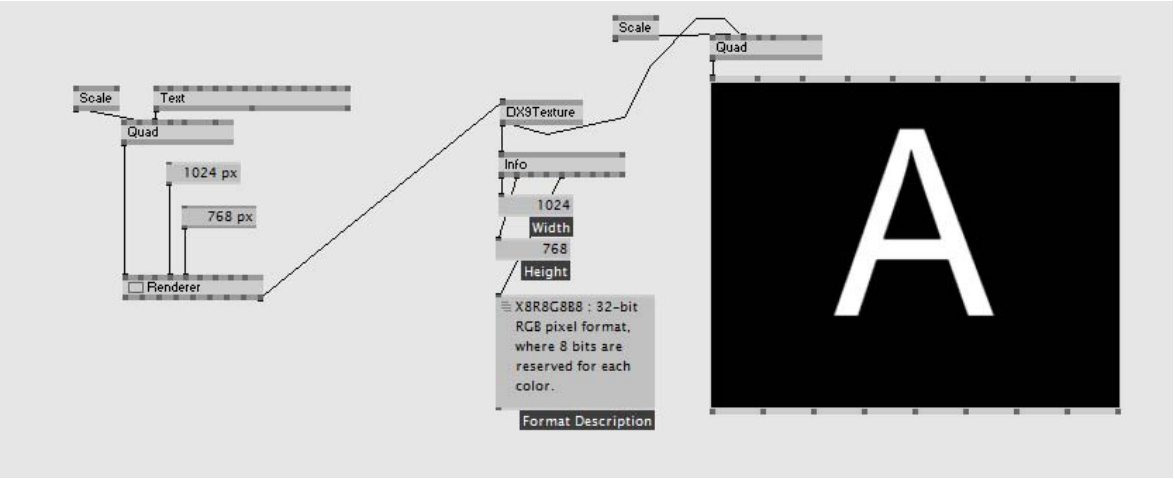
Mindegyik Render modulnak van textúra kimenete, amit újabb Quad-ba, vagy esetleg Render-be tudsz bekötni. Alapesetben maguk az effektek is így épülnek fel: ha jobbklikk-kel belemész egy effektmodulba, láthatóvá válik, hogy ez is hasonlóképpen épül föl, mint az eddig látott patchek, csak itt a Quad vagy más megjelenítők helyett Shader-eket használnak, a végeredményt pedig egy DX9 Texture modul nyeri ki.



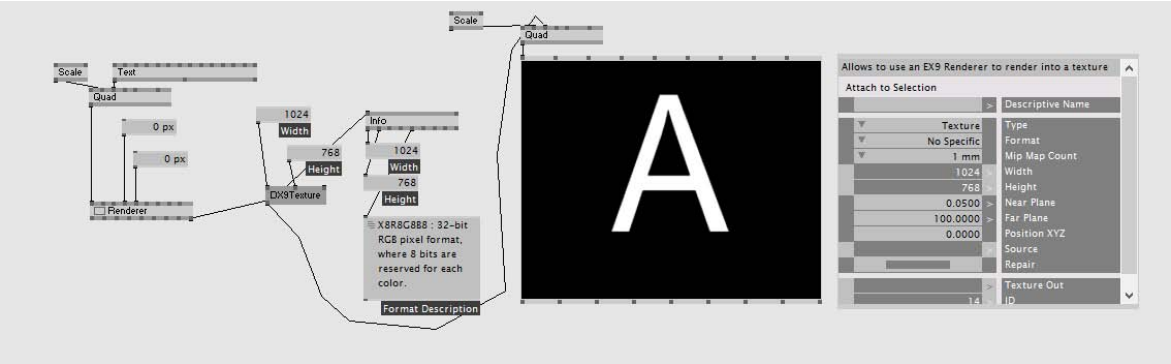
9.7. ábra: Felbontás beállítása nélkül eltüntetett Render

Ha a későbbiekben DX9 Texture-t használsz, mindig figyelj arra, hogy az előtte lévő Render-ed mekkora felbontásban hajtja végre a renderelést, mivel ha nem állítod be előre a méretét, akkor a program az aktuális ablakméretet veszi alapul a renderelés felbontásához. Ha viszont beállítod, onnantól kezdve bármekkora is az ablakod, sőt akkor is, ha ALT+3-mal egészen eltünteted, a program mindig akkora felbontásban

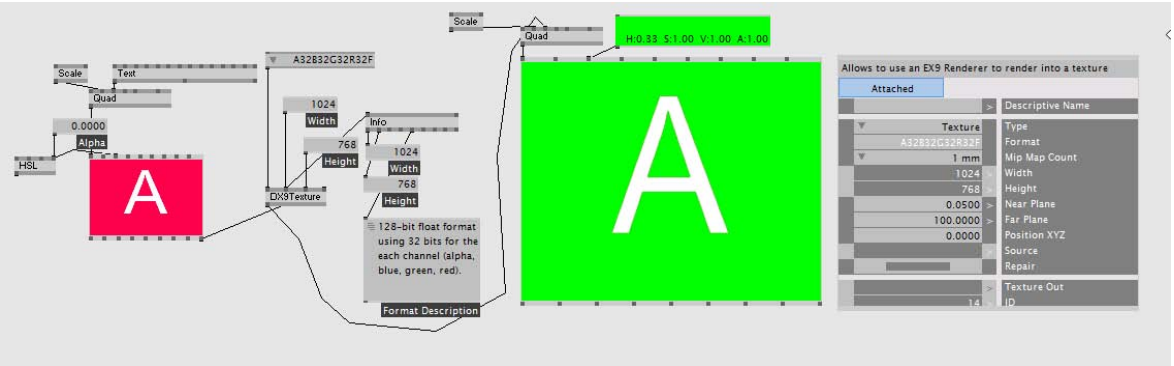
renderel, amekkorát megadtál. Ezt a paramétert külön is be tudod állítani a DX9-ben Inspektor segítségével.



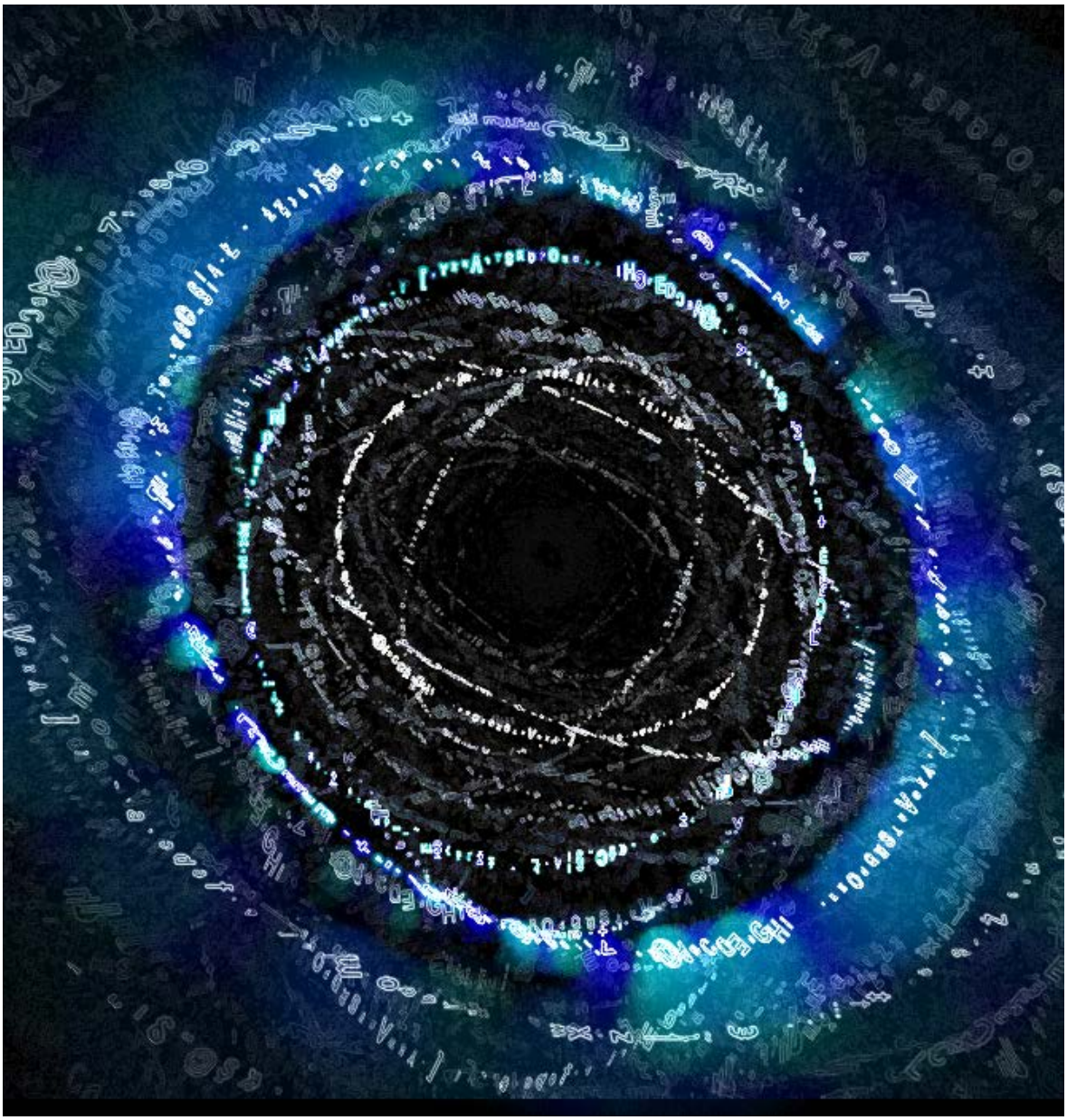
9.8. ábra: Felbontásbeállítással eltüntetett Render



9.9. ábra: Felbontás beállítása DX9Texture-ben

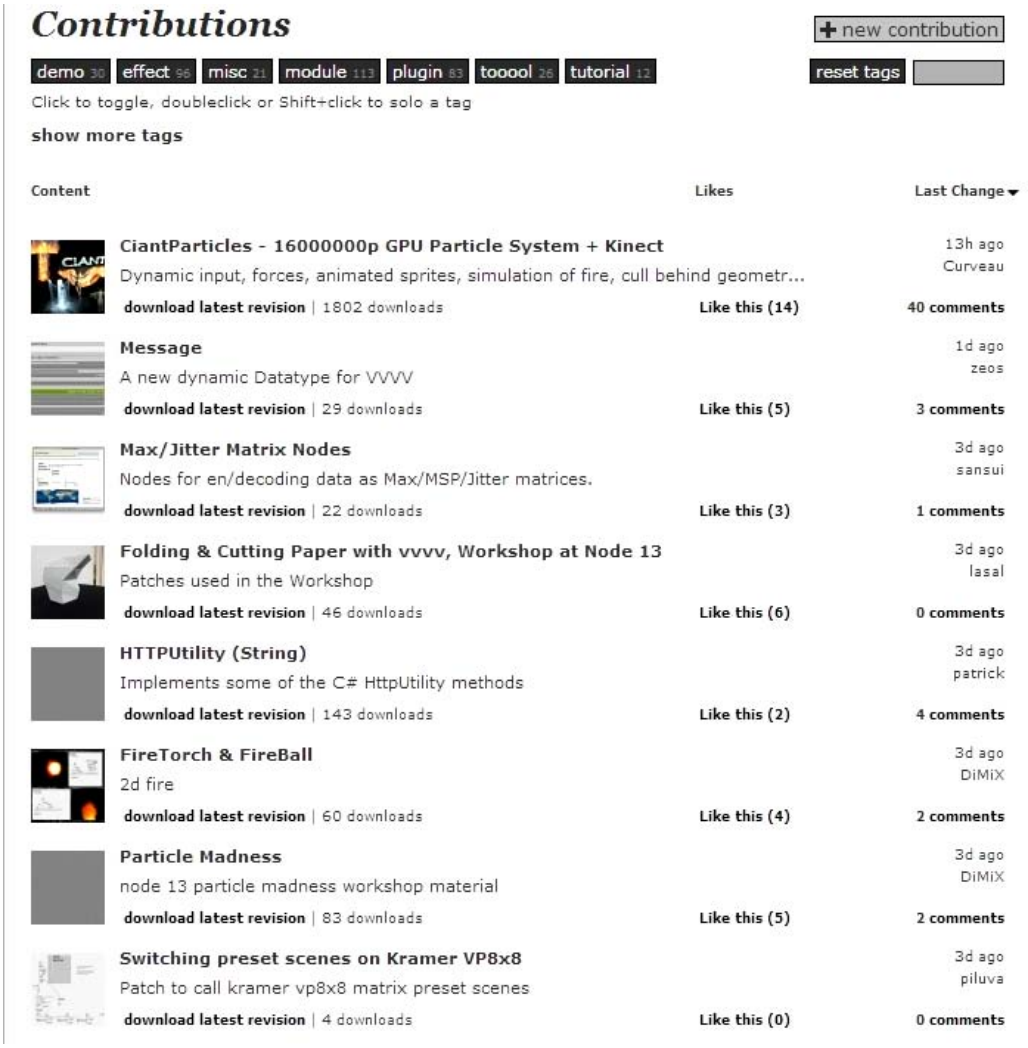


9.10. ábra: Alpha texture kimenet



9.11. ábra: Példa effekt

A VVVV fejlesztőinek és a felhasználóknak köszönhetően ma már számos plugin és kiegészítő tölthető le a VVVV oldaláról, a http://vvvv.org/contributions webcímről. Itt elsősorban még nem beépített, alfa vagy béta állapotú plugineket, projekteket, hardvereszközök kezelőit és különféle Shadereket találsz.

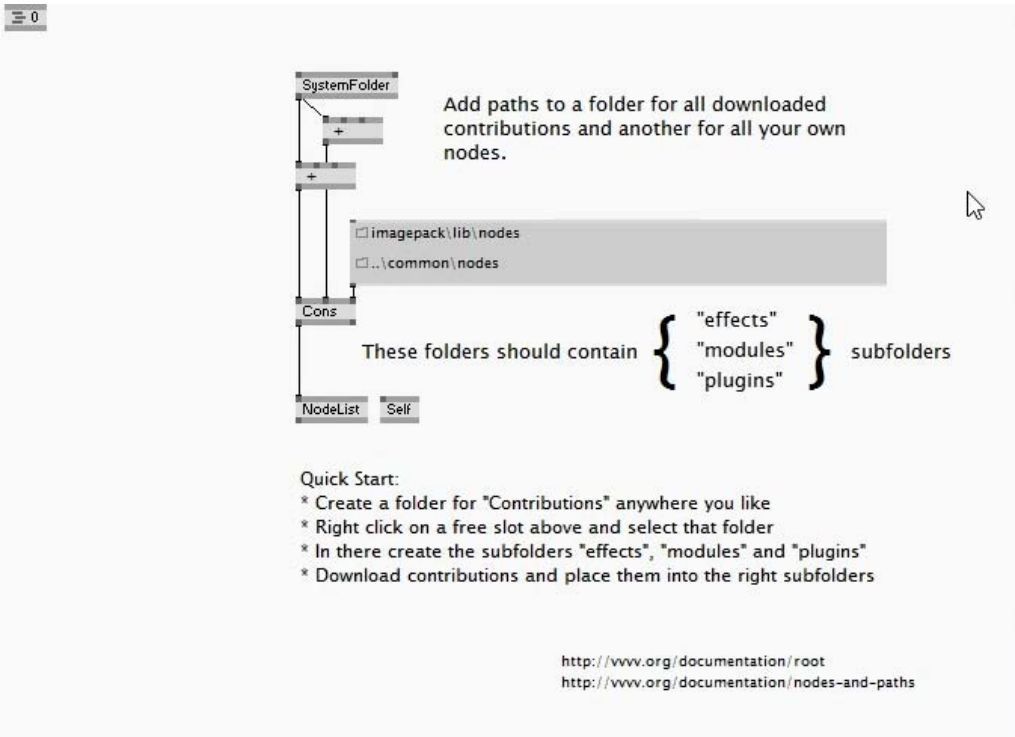


9.12. ábra: Contribution oldala

Az oldalt érdemes gyakran látogatni, mivel sokan töltenek fel hasznos kiegészítőket, és ezek számos probléma megoldását leegyszerűsítik. Mindegyik kiegészítőhöz viszonylag jó dokumentációt találsz a rá vonatkozó oldalon, valamint magában a patch-ben is, de ha problémád merül fel vagy hibát találsz a programban, azt érdemes jelezni a patchre vonatkozó oldalon hozzászólásként. Sokszor fordul elő, hogy egy probléma megoldását épp a fórumokon lehet megtalálni, ezért specifikus kérdések esetén érdemes a keresőt is használni.

Ha az idők során már egész plugingyűjteményt sikerült összeraknod olyan kiegészítőkből, amiket sokszor használsz, és meguntad, hogy ezeket mindig bemásold a patched alapkönyvtárába, akkor a következőt érdemes tenned.

Indítsd el a VVVV-t, és közvetlenül a program betöltődése után nyomd le az ALT+R kombinációt, vagy középső egérgombbal hozd elő a menüt és válaszd ki a Show Root funkciót. Ekkor felugrik a Root.v4p nevű ablak. Fontos, hogy ilyenkor az addig nyitva lévő ablakok mind bemásolódnak a gyökérpatch-be. Jelen esetben ezt a 0.v4p-nek hívják, amit ki kell törölni, máskülönben ez a patch, vagy az, ami akkor nyitva volt, minden indításnál el fog indulni. Persze lehet éppen az is a célod, hogy a program indításakor mindig egy adott patch induljon el. Ez esetben nem kell mást tenned, mint hogy az elindítandó programot behúzod a Root.v4p-be.

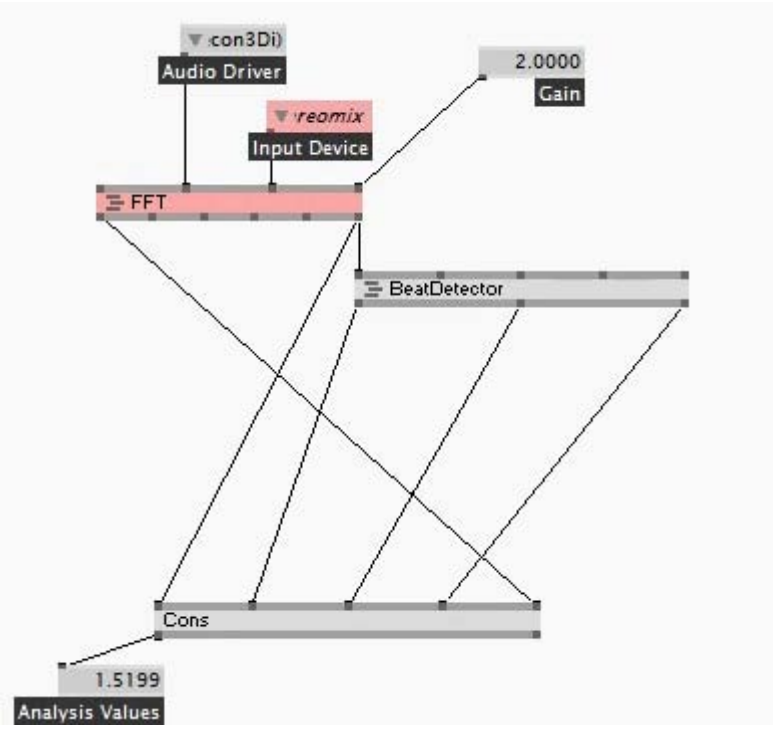


9.13. ábra: Root patch

A pluginek automatikus betöltéséhez a Root ablakban meg kell keresned a String nevű listát, ami a Cons jobb bemenetére van kötve: itt tudod kiválasztani jobbklikkkel, hogy indításkor melyik könyvtárt töltse be automatikusan a program. Ha ezeket kijelölted, akkor csak mentened kell és ki is léphetsz. Fontos, hogy ennél a módszernél nem érdemes túl sok alkönyvtárat használni, így nagyobb mélységű listák esetén inkább add meg őket külön-külön. Ebben az esetben, ha a betöltött könyvtárba új plugint másolsz, az a program futása közben is automatikusan betöltődik, vagyis nem szükséges hozzá újraindítani.

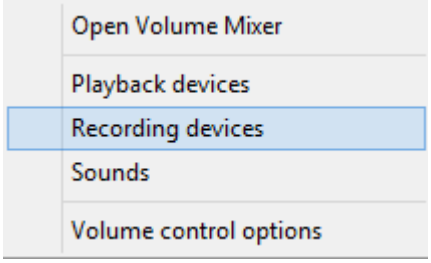
10. Hangvezérlés

Hangot a legegyszerűbben az AudioAnalysis modullal lehet vezérelni.

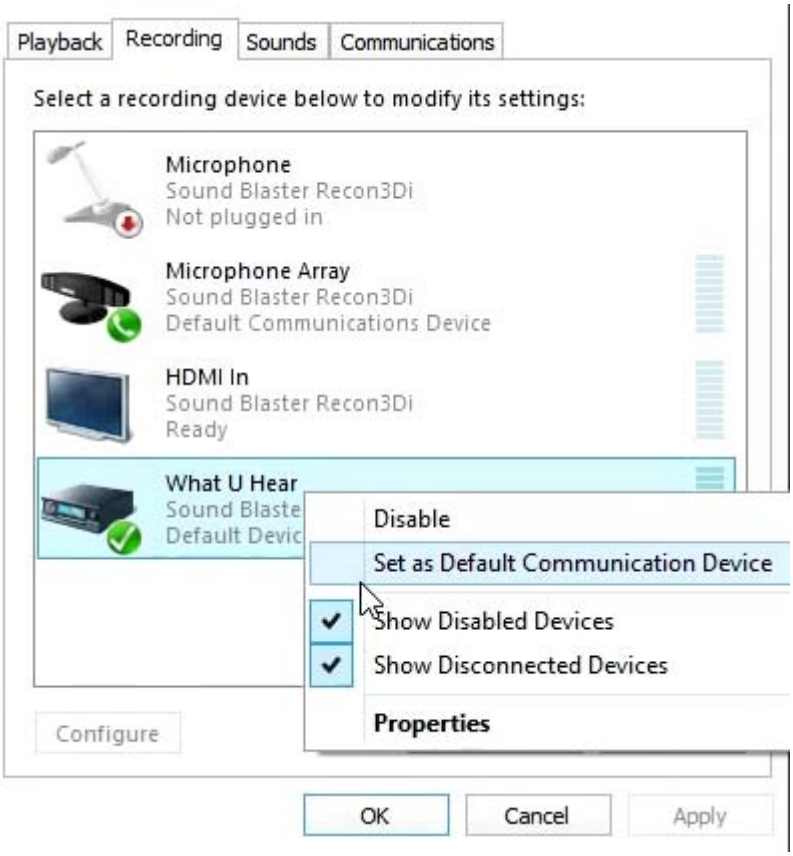


10.1. ábra: AudioAnalysis

Ez egy készre gyártott, több modulból álló detektáló eszköz, aminek az alapja az FFT nevű hanganalizáló modul. Ez a hang alapján 256 darab számsort állít elő, amelyek a hang egy-egy tulajdonságát adják meg. Az AudioAnalysis-nek egy kimenete van, amin egyszerre küldi ki ezeket az információkat, mégpedig a következő sorrendben: 0-3-ig a mély, mélyközép, magasközép és magas csatornákét, 4-7-ig a BeatDetect leütéseit, majd 8-11-ig ennek a Decay változtatásait, 12-17-ig az EnergyOut-ot, végül pedig 18-274-ig a teljes FFT-t.



10.2. ábra: Felvételi eszköz kiválasztása



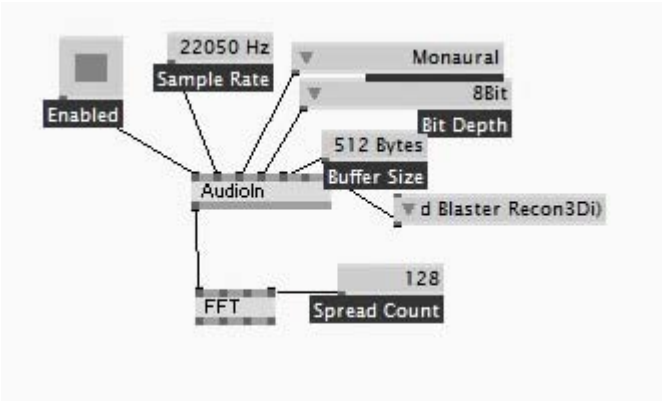
10.3. ábra: Felvételi eszköz beállítása Windows-ban

Ebben a modulban ki tudod választani, hogy milyen forrást nézzen mekkora gain-nel és hogy sztereó vagy monó legyen-e. Itt bemenetnek nem csak fizikai audióbemenetet lehet használni; Windows 7-ben a hangbeállításoknál be lehet állítani a felvételi eszközöknél azt, hogy belső hang is legyen. Ez alapesetben nem látható a Windows-ban, de ha ráklikkelsz a háttérre, és kiválasztod, hogy a mutassa a kikapcsolt bemeneteket is, akkor ez is láthatóvá válik. Így ha egy másik programban elindítasz egy hanglejátszást, akkor ezen keresztül azt már a VVVV is megkapja. Ha az

AudioAnalysis kimenetének csak egy bizonyos részét szeretnéd használni, akkor két választásod van.

Az első, ha a SplitAudioAnalysis modult használod, ami az AudioAnalysis modult szétszedi csoportokra. A másik mód, ha a GetSlice vagy a GetSpread modulokat használod, ahol pontosan meg tudod adni, hogy a kapott adatfolyamból mettől meddig szeretnél használni.

Most pedig készítsünk el egy FFT alapú példát!



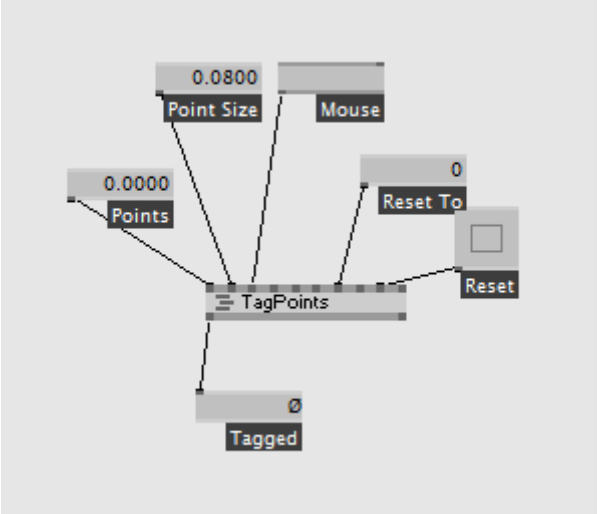
10.4. ábra: AudioIn és az FFT

Hozz létre egy AudioIn (Dshow9) modult, kapcsold be és állítsd be a mintavételezést 22050 Hz-re, 512 Byte-tal, monóra, 8 biten. Majd ezt követően hozz létre egy FFT (Dshow9) modult és kösd össze az audió bemenettel. Az FFT-nek érdemes beállítani a Spread Count-ját, méghozzá 64, 128, 256-ra. Ha közben elindítasz valami zenét vagy hangot, most már láthatod, ahogy az első 2 kimeneten hogyan változnak az értékek 0 és 1 között. Ezek után hozz létre egy Render-t Group-pal és köss bele egy Quad-ot, abba pedig egy Transform 2d-t. Majd hozz létre egy Linear Spread-et, állítsd át a szélességét 2-re. A Spread Count-ot érdemes azonos számbox-ba kötni az FFT-vel, például 128-ra. Ezzel a Linear-ral határozzuk meg a létrehozandó Quad modulok X pozícióját, amihez a Quad-unk szélességét le kell csökkenteni, mégpedig úgy, hogy a Spread Count értékével el kell osztani 1.5-öt, aminek az eredménye 0.0117. Ezt az eredményt kösd a Transform Scale modul X értékébe, a Linear Spread-et pedig az X pozícióba. Ha a későbbiekben szeretnéd megváltoztatni a mintavételezés felbontását, a négyszögek szélessége dinamikusan fog változni. A következő lépés a magasság beállítása. Ahhoz, hogy a Quad magassága -1-től induljon és 1-ig érjen, az FFT-nek el kell tolni az Y centerét és ezzel együtt az Y pozíciót is.

Tehát állítsd át az Y pozíciót -1-re és a Center Y-t -0.5-re. Ezek után, ha bekötöd az FFT-t a Scale modul Y bemenetére, akkor már látható lesz. Érdemes erre a bemenetre

egy Min modult is kötni, amivel be tudod állítani a maximum magasságot, mondjuk 2-re. A bejövő jeled simítását InputMorph-fal tudod megoldani: kösd az Input 1-be az FFT-t, míg a kettes bemenetre az InputMorph kimenetét egy FrameDelay-vel – így 0-1-ig tudod állítani a simítás mértékét. Ezt az értéket, ha elosztod 2-vel, egy újabb InputMorp Color-ba kötöd és beállítod azt, hogy milyen két szín között képezzen átmenetet, akkor már a színeid is változni fognak.

A bejövő hangot vonal alapon is ki lehet rajzolni; ehhez hozz létre egy Line EX9-et és kösd bele a Group-ba. Alapesetben ez a modul két pozíció között vár adatokat, viszont ha kijelölöd Single Input-ot, már eggyel is megelégszik, és a Vertices XYZ-n várja csak a paramétereket. Ebbe a bemenetbe köss egy Vector 3d-t, annak az X bemenetére kösd rá a fentebb használt Linear Spread-et, az Y pozícióra pedig egy Map-et, amivel átméretezed a lesimított FFT-t -1-től 1-ig. Ha esetleg a zenelejátszókból ismert, lassított visszatérési idejű vonalat szeretnéd használni, akkor a Vector és a Map közzé köss egy Decay-t és annak a lefutási idejét állítsd át a neked megfelelőre.



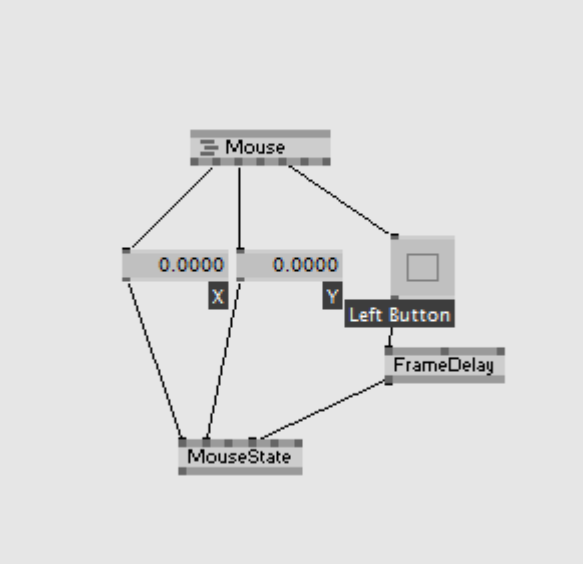
10.5. ábra: TagPoint plugin

A következőkben azt nézzük meg, hogyan lehet egy adott szakaszt kijelölni, és annak az értékét átlagolni, hogy az FFT-nek csak egy adott részéből nyerjünk ki információkat.

Először az egér pozíciójától függő kijelölést kell megoldani: erre van egy előre elkészített modul, mégpedig a TagPoint. Az első bemeneten kell bekötni a pontok pozícióját, vagyis a LinearSpread-et egy Vector 3d-n keresztül, úgy, hogy a második és a harmadik bemenetét 0-n hagyjuk. A második bemenete a Quad-ok mérete, vagyis egy 3d Vector, aminek az X értéke a fentebb Spread Count-tal elosztott 1.5, az Y értéke pedig 2 – mert maximálisan akkora méretű a Quad –, a Z pedig 0. A modul a harmadik

bemenetén várja az egérpozíciót. Sajnos ebben az esetben nem elég csak simán belekötni az egeret, mert később a kijelölés törlésénél késlelteni kell a klikkelést 1 frame-mel, és a jobb klikket sem használhatjuk, mert arra a törléshez lesz majd szükségünk.

Ehhez hozz létre egy Mouse System Window-t, és egy MouseState Joint-ot, összekötve a bemeneteiket: X koordinátát az X-be, az Y-t az Y-ba, a bal klikket pedig egy FrameDelay-en keresztül a bal klikkbe. Így már belekötheted az egészet TagPoint-ba. Most kösd össze az egér jobb és bal egérgombját egy Or-ba, és azt kösd bele a Reset Pin-be. Végül köss egy NIL modult a Reset To bemenetbe: ez azért kell, hogy amikor törlöd a kijelölést, a program ne jelöljön ki semmit, a NIL modul ugyanis konkrétan semmit küld információként. Ennek van egy fordított modulja is, az AvoidNIL, ami ha nem kap jelet, egy általunk megadott számot küld ki folyamatosan. Így ha a kimenetre kötsz egy Sample And Hold-ot, vagyis S+H-t, és az átengedést csak akkor engedélyezed, amikor klikkelés van, vagyis az Or-t kötöd be, már készen is vagy a kijelölési résszel.



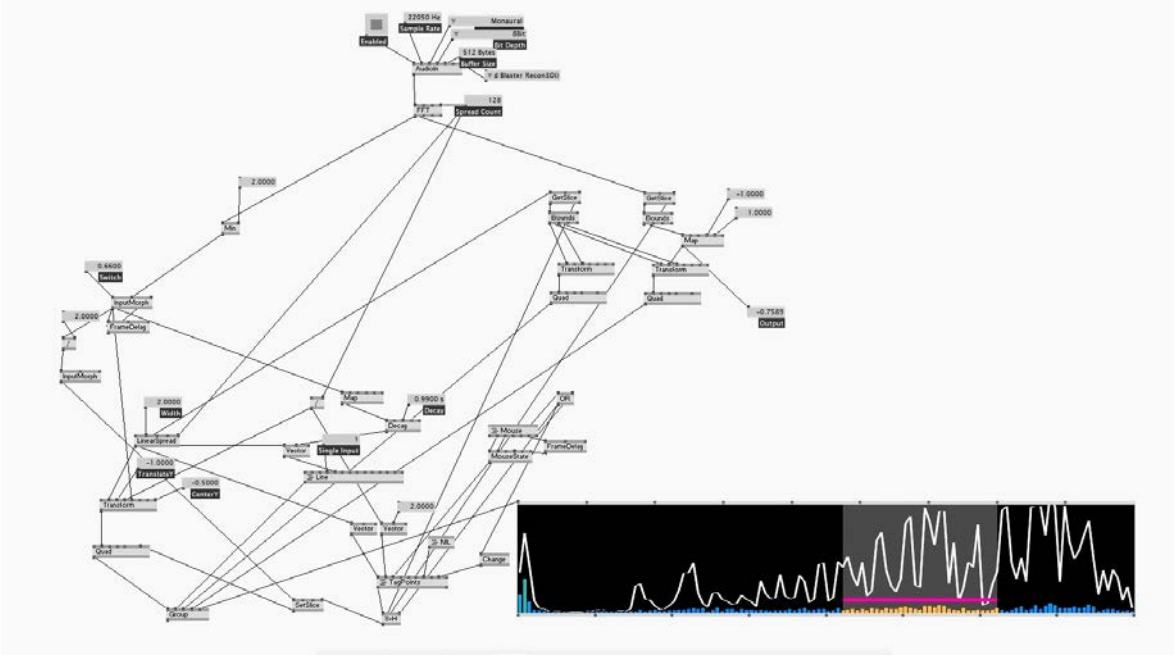
10.6. ábra: Mouse és MouseState TagPointhoz

Most hozz létre egy SetSlice Colort és kösd a fenti Quad és az InputMorph közé, majd adj meg egy teljesen ellentétes színt, és az utolsó pinjére, vagyis az Index-re kösd rá az S+H kimenetét. Ekkor már, ha egérrel kijelölsz valamit, az oszlopaid átszíneződnek. Most a kijelölés szürkítése következik, hogy jobban látszódjon: ehhez kell egy Quad és Transform 2d modul, amiket bele kell kötni a Groupba. Köss egy GetSlice indexbemenetet az S+H-ra, az inputját pedig a LinearSpreadbe, hogy megkapd a kijelölt Quad-ok pozícióit. Ezen értékek átlagolásához Bounds modulra lesz szükséged: ha ezt belekötöd, akkor megkapod a bejövő érték pontos közepét,

szélességét, minimumát és a maximumát. Kösd a közepét az X-be, a szélességét a Scale X-be és állítsd át a Scale Y-t 2-re.

A kijelölt értékek mozgásának átlagolására is a Bounds kell: ehhez másold le az előzőleg megcsinált Quad Bounds részt, és kösd bele a Group-ba. Kösd az előző Bounds középső- és szélességértékét az új Transform megfelelő helyére, és a másik Bound-ot, ami szabadon maradt, kösd az Y-ba egy Map-en keresztül, amit átméreteztél -1 és 1 közé. Végül kösd az ehhez tartozó GetSlice Input bemenetére az FFT-t, és méretezd át a Scale Y-t 0.1-re.

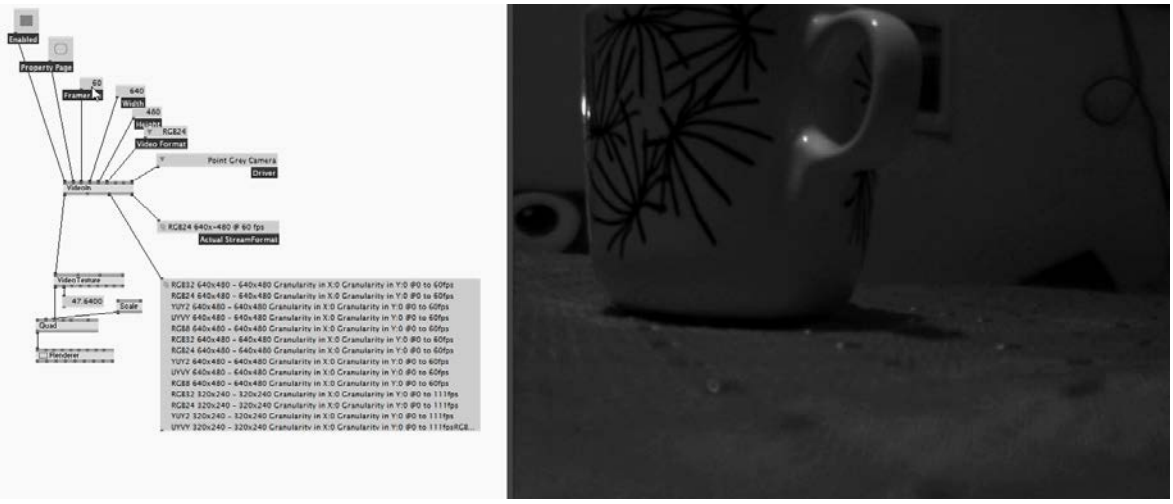
Az elkészült modulon ezzel a Bound-szal tudsz végül mást is vezérelni, illetve az előzőtől függetlenül is tompíthatod InputMorph modul használatával.



10.7. ábra: FFT patch

11. A kamera használata – kinyerhető adatok felhasználási lehetőségei

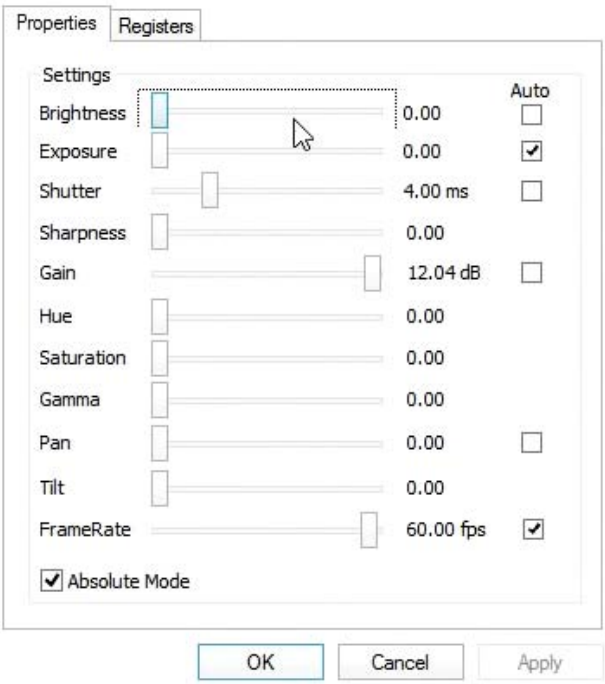
Kamerabemenet használatára a VVVV-ben több lehetőség is van, ezek közül a legegyszerűbb a programba már integrált VideoIn. Ez egy Directshow alapú modul, vagyis minden, ami a Directshow-val kompatibilis, az működni fog itt is. Sajnos számos videokamera nem feltétlenül működik Windows rendszer alatt, így vásárlás előtt érdemes leellenőrizni, hogy az adott kamera WDM kompatibilis-e.



11.1. ábra: VideoIn DirectShow

A VideoIn modulban be lehet állítani, hogy milyen Fps-sel jelenjen meg a kép, valamint a kép felbontását és formátumát (például MJPG, YUY2, UYVY, RGB8, RGB24, RGB32). Inspektor segítségével még számos paraméter állítható benne, mint a Brightness, Contrast, Exposure, Gain stb. – ez utóbbiakhoz viszont szükséges, hogy maga a kamera is engedélyezze ezeket a módosításokat. Ha a Property Page fülre klikkelsz, a felugró ablakban megjelenik a kameradriverhez tartozó Windows-os menü. A modul utolsó két kimenetén pedig látható, hogy milyen módokat és felbontást támogat, illetve hogy aktuálisan éppen milyen módot használ.

Hasonlóan a Filestream-hez, ehhez a modulhoz is szükséges egy VideoTexture, mert csak ezen keresztül tudod belekötni a Quad-ba, vagy bármilyen másfajta TextureFX-be.

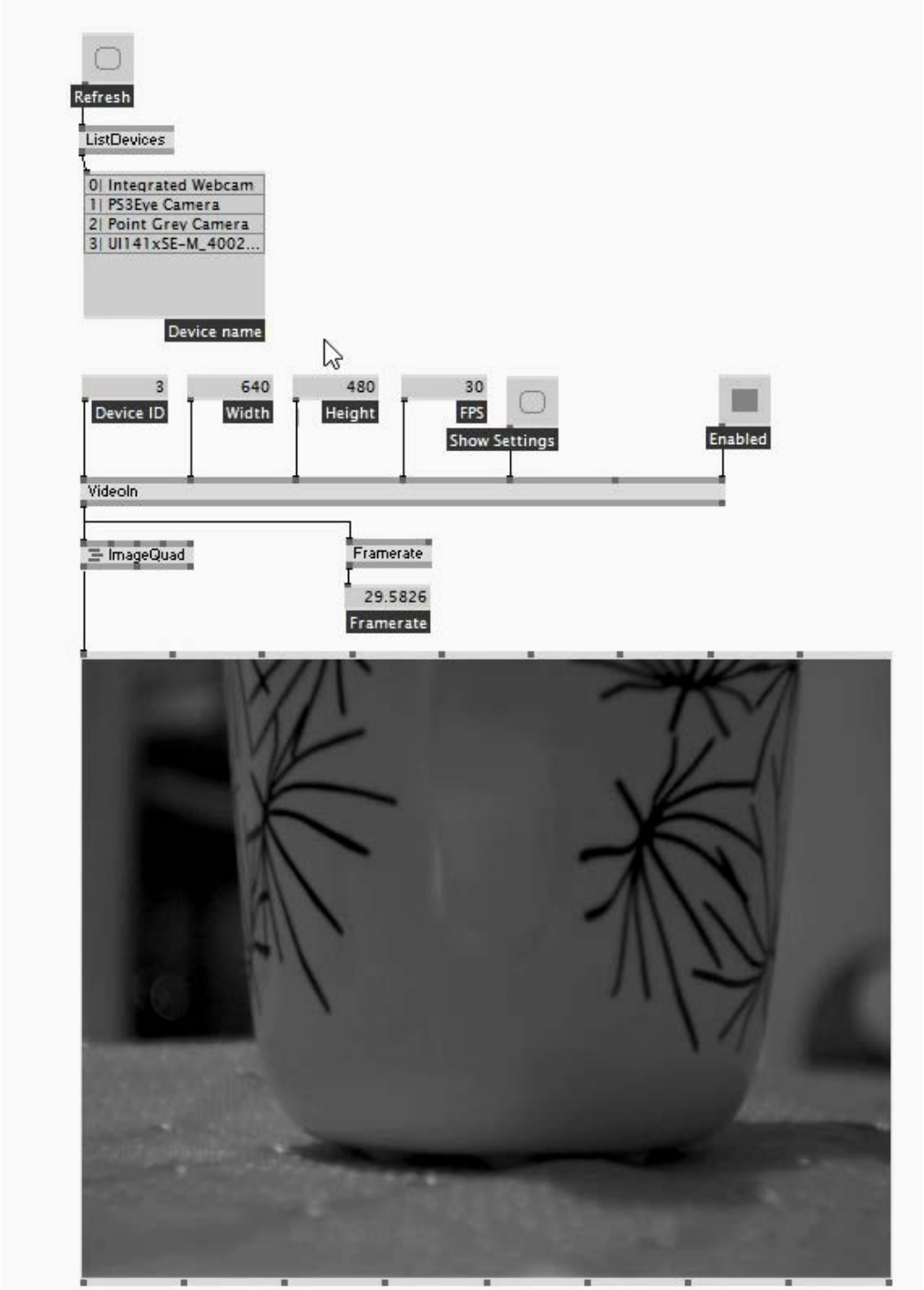


11.2. ábra: VideoIn beállításpanel

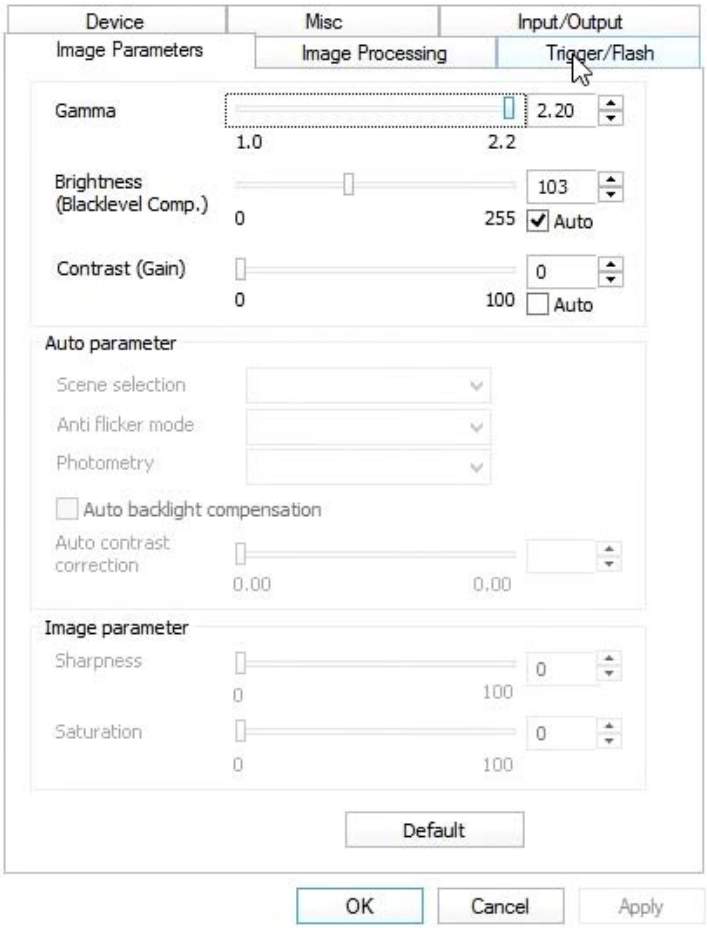
A másik, ritkábban előforduló kameramodul az uEye. Ezt a kamerát az IDS gyártja, és elsősorban ipari célokra gyártják, gyártósorok scannelésére és hibakeresésre. Ezen rengeteg paramétert lehet beállítani és kiemelkedően tiszta képet ad, ennek következtében viszont elég drága is. Hasonlóan jó minőségű kamerákat gyárt még a Point Grey, a Basler és az Imaging Source is, ezek közül a Point Grey a legelérhetőbb árú. Ha megteheted, akkor érdemes beruházni egy ilyenbe, mivel a pontos paraméter elérhetőség és a zajmentes kép nagyon sokat segít a detektálásban.

11.3. ábra: VideoIn Ueye

Az OpenCv VideoIn kamerabemenet alapesetben nincs integrálva a programba, ezért ezt a Contribution oldalról kell letölteni, a következő webcímről: http://vvvv.org/contribution/vvvv.packs.image.



11.4. ábra: VideoIn OpenCV DirectShow



11.5. ábra: VideoIn OpenCV Ueye beállításpanel

A letöltés után rákeresel a VideoIn modulra, már látszik a három új modulverzió, a CLEye, az OpenCv és az OpenCv DirectShow. A CLEye PSP Eye kamerákhoz használható, de nekünk ezek közül a DirectShow verzió fog kelleni. Hasonlóan az eredeti DirectShow VideoIn-hez, itt is a felugró ablakban tudod beállítani a paramétereket, illetve ha a CaptureProperty (DirectShow) modult használod, akkor azon kívül is. A textúrát ebben az esetben az AsTexture modullal tudod kinyerni. Érdemes végigmenni a Help fájlokon, mert számos új funkciót mutatnak be, mint például a projektorkalibráció vagy a StructuredLight.

11.6. ábra: IDS Ueye

11.7. ábra: Imaging Source

11.8. ábra: Basler kamerák

11.9. ábra: Point Grey Flea

11.10. ábra: Point Grey 3d kamerák

A videóbemenetek listájában az utolsó a Kinect/Primesense. Kinect kamera kezelésére két lehetőség van, az első, ha a Microsoft Kinect SDK-t, az ahhoz tartozó modulokat használod, a másik pedig az OpenNi által készített plugin, amihez viszont

telepíteni kell az OpenNi-t, a Nite-ot és a Driver-t. A Driver-ből két variációt használhatsz a kamerád típusától függően: van egy Kinectre és egy másik Primesense kamerákra, amiket az Asus gyárt Xtion, illetve Xtion Pro néven (ezek közül csak a Pro verzió rendelkezik színes kameraképpel). A Xtion kamerák nagy előnye, hogy majdnem harmada a méretűk a többi kamerához képest, és nem kell hozzájuk plusz tápegység, vagyis egy egyszerű USB-kábellel már működtethetőek. Ugyanakkor nincsen bennük szervomotor, amivel a kamera szögét lehet állítani – erre azonban csak speciális esetekben van szükség, így ez nem okoz nagy problémát.

11.11. ábra: Asus Xtion, Xtion Pro

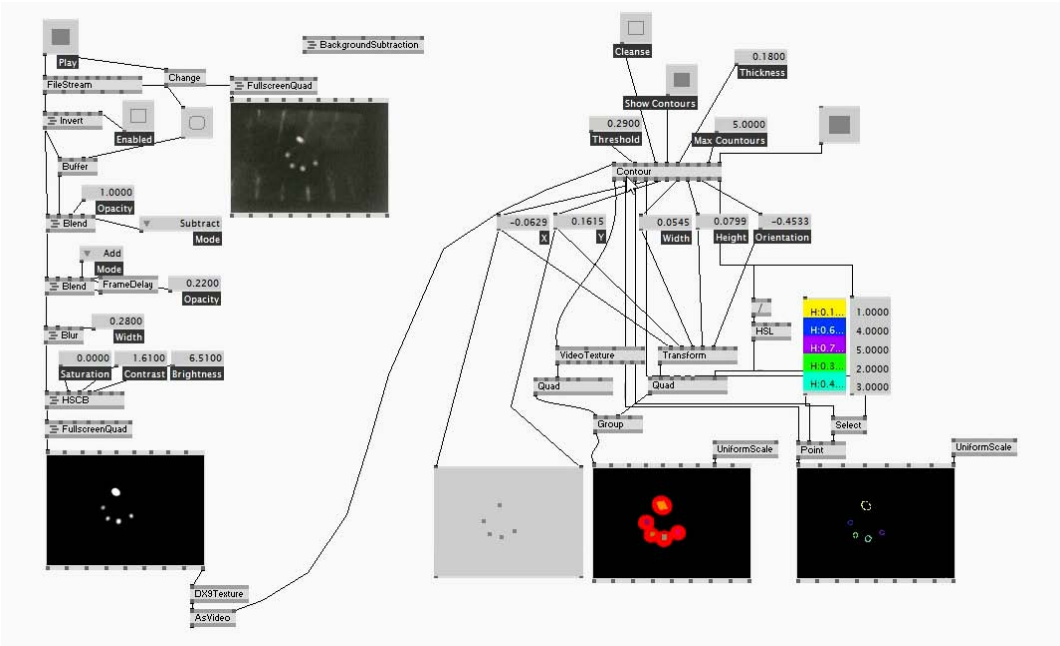
Minden Kinect és Primesence driver és telepítő csatolva megtalálható a könyvtárban. Fontos, hogy a Microsoft Driver-t és az OpenNi Driver-t egyszerre nem tudod használni, hanem mindig váltani kell közöttük az eszközkezelőben. Így ajánlatos előre eldöntened, hogy melyik plugint szeretnéd használni, mert a nem megfelelő telepítésük számos bonyodalmat okozhat később. Ahhoz, hogy az eszközt végül felismerje a VVVV, a fentieken kívül még az OpenNi-hez tartozó megfelelő drivert és a Nite-ot kell telepíteni.

KINECT™
for XBOX 360.

11.12. ábra: Microsoft Kinect

A kameránkat sikeresen összekötöttük a VVVV-vel, ideje végigvennünk a különböző trackelési lehetőségeket.

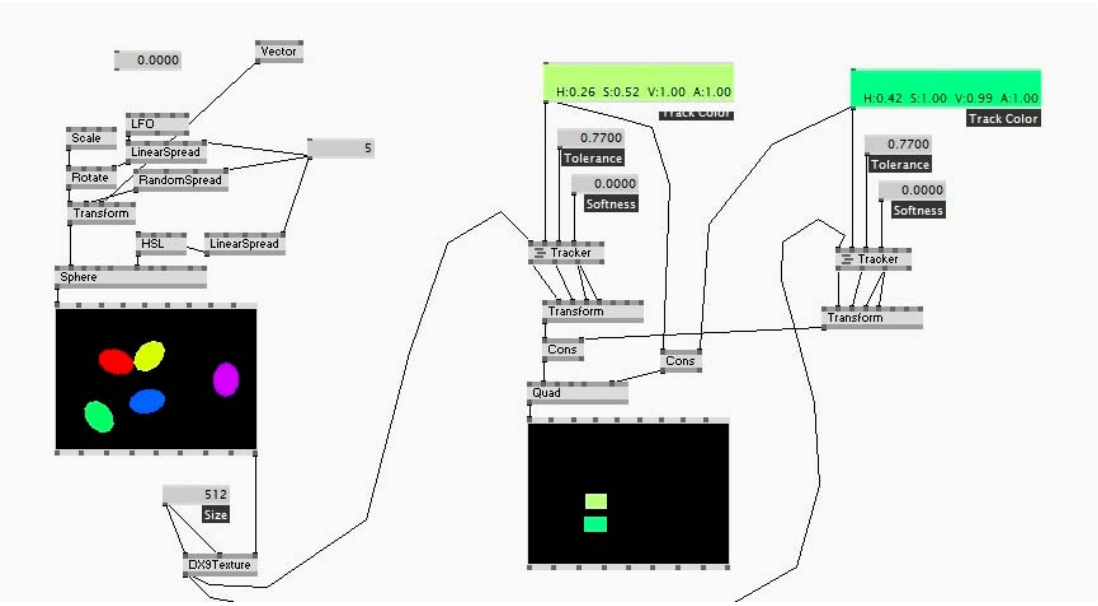
Ezek közül az első a Contour detektálás: ehhez nyisd meg a könyvtárban található Contour.v4p fájlt, és indítsd el a videólejátszást. A bal oldali sávban látható kép a bejövő képnek egy olyan letisztított verziója, amit már képes értelmezni a Contour modul. A legfelső videón látszik, hogy egyelőre nagyon zajos a kép, az infraledek miatt pedig még plusz fénysávok is vannak rajta. Ezt úgy lehet letisztázni, hogy egy kezdő képet, ahol még nem látszik semmi, ki kell vonni a bejövő videóból egy Buffer-rel. Ez végső soron nem más, mint egy alapszintű Background Substraction. Ha nem elég erős a kép, akkor Feedback és HSCB segítségével kicsit tudsz rajta erősíteni, a kamera zajosságát pedig Blur-rel tudod csökkenteni. Ezek után már csak a fényerőt és a kontrasztot kell beállítanod, és a szaturációt levenni nullára.



11.13. ábra: Contour Tracking

A Contour modul bemenete csak DirectShow típusú videót, vagyis egy eredetileg VideoIn modulból jövő jelet tud értelmezni, így a DX9Texture-t AsVideo-val át kell alakítanod ilyen formátumra. Ezek után már be tudod állítani a toleranciáját (hogy maximum hány pontot érzékeljen), és hogy végezzen-e tisztítást a képen. A kimeneteken pedig megkapod a talált pontok kontúrjának megrajzolásához szükséges pozíciókat, a detektált pontok középpontjait, szélességét, irányát és azokhoz tartozó címzést. Ezt az eljárást multitouch célokra vagy akár emberek kontúrjainak a megrajzolásához is tudod használni.

A második lehetőség az UNC által készített Tracker modul, amivel színeket lehet lekövetni, és a működését a Color.v4p fájlban láthatod. A Tracker modul viszonylag kevés beállítási lehetőséggel rendelkezik: először a trackelni kívánt színt tudod megadni, majd a hozzá tartozó toleranciát és simítást. Több színt nem tud egyszerre felismerni, és egy szín esetében is ajánlatos csak egy színfolttal használni, mert több folt esetén a közöttük lévő abszolút középpontba fogja helyezni a pontot.

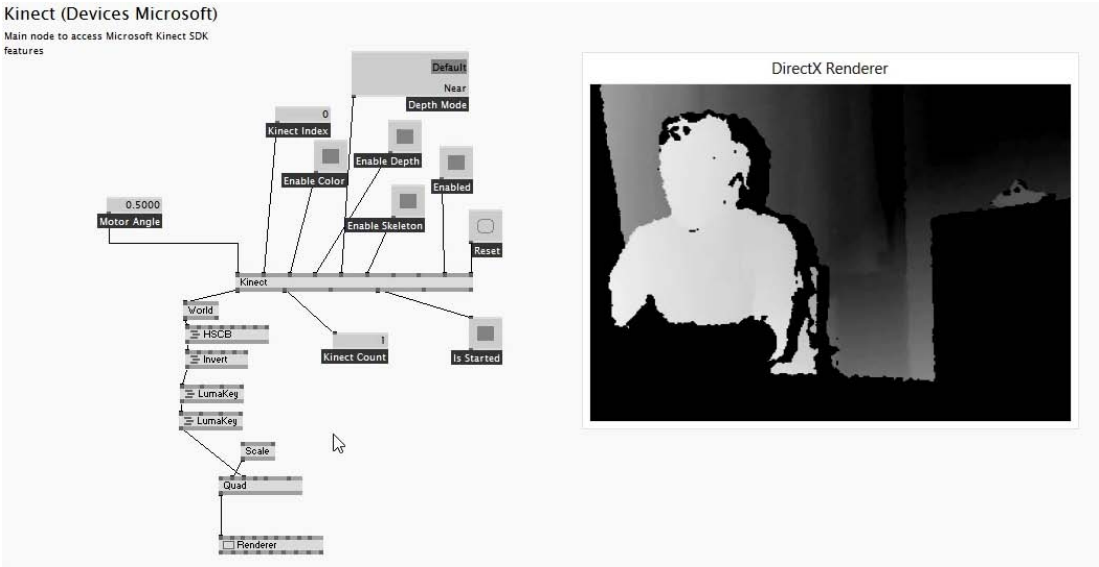


11.14. ábra: Color Tracking

A harmadik módszer a Primesense alapú trackelés: ehhez mindkét fejlesztőnél hasonló modulok találhatóak, amilyen például az RGB, a User/Player, a Depth és a Skeleton. A Hand és a Gesture viszont csak az OpenNi-ben van, míg a Face csak Microsoft moduljában. Mindkettőben közös viszont, hogy a futtatásukhoz Kinect modul szükséges, amire aztán rá tudod kötni az RGB-t, ha kamera képét akarod látni (a sima Xtion-nál ez nem működik, mivel nincsen kamera). A mélységhez a Depth modulra, csontvázhoz pedig a Skeleton-ra lesz szükséged. A Depth-nél egyszerűen egy Lumakey segítségével tudod beállítani, hogy mettől meddig érzékeljen, és ugyanez vonatkozik a User-re is, csak itt színre bontva kapod meg a felhasználókat, és az alapján tudsz szűrni közöttük.

Az OpenNi Skeleton-nál X Y Z pozíciókat kapsz, annak megfelelően, hogy melyik kulcspontot jelölted ki a listából. Ez lehet egyszerre az összes is, de ha csak a fej és a két kéz pontos pozíciója érdekel, ezt is ki tudod választani.

Az egyes modulokkal kapcsolatban mindegyikhez találsz példafájlt a mellékelt könyvtárban.



11.15. ábra: Kinect mélység

11.16. ábra: Kinect felhasználódetektálás

12. Külső kontrollerek használata

Ebben a leckében a külső vezérlőeszközökkel fogunk megismerkedni.

A *VVVV*-ben számos lehetőség van a program külső vezérlésére, akár Midi-alapú kontrollerek, Wacom Tablet, Wiimote, Joystick, Android vagy iPhone telefonok segítségével.

12.1. ábra: Behringer BCR Midi eszköz

Először lássuk a Midi-alapú kontrollereket! Ilyeneket rengeteg cég kínálatában találhatunk, mint amilyen az Akai, a Behringer, a Korg, az Alesis és a Doepfer – szerencsére a *Contributions* oldalon vagy a fórumon szinte mindegyikhez található előre elkészített *VVVV*-patch, amit használhatunk. Ezek közül a leggyakrabban az Akai, a Behringer BCR széria és a Korg Nano kontrollereit szokták használni.

Mi most a *Korg Nano Kontrol*-t fogjuk részletesebben megnézni. Ez egy nagyon olcsó, akár 13000 Ft-ért is beszerezhető vezérlő, ami nem a pontosságáról híres, viszont cserébe hordozható méretű, és nagyon sok csatornával rendelkezik. Az eszköz három különböző verzióban érhető el: a *Kontrol* nevet viselő alaptípuson 9 potméter, 9 slider, 9 gomb és kapcsoló található, míg a *Pad* verzión 12 pad és egy touch felület, a *Key* nevű verzión pedig 25 billenytű.

12.2. ábra: Korg Nano széria

A Midi elérése a *MidiController* modullal lehetséges, amin ki tudod választani, hogy melyik Midi eszközt szeretnéd használni, melyik csatornát, és melyik vezérlőt. Ha ez megtörtént, a *Controller* által használt kiosztást *GetSlice* segítségével tudod beállítani. A bonyolultabb vezérlőkben arra is van lehetőség, hogy mindegyik kontrollernek egyéni kiosztást adj, illetve ha rendelkezik *Encoder*típusú, vagyis végtelenített potméterekkel, akkor azt is kiválaszthatod, hogy azok lineáris vagy logaritmikus módon működjenek-e, valamint hogy mettől meddig tartson az értékük. A

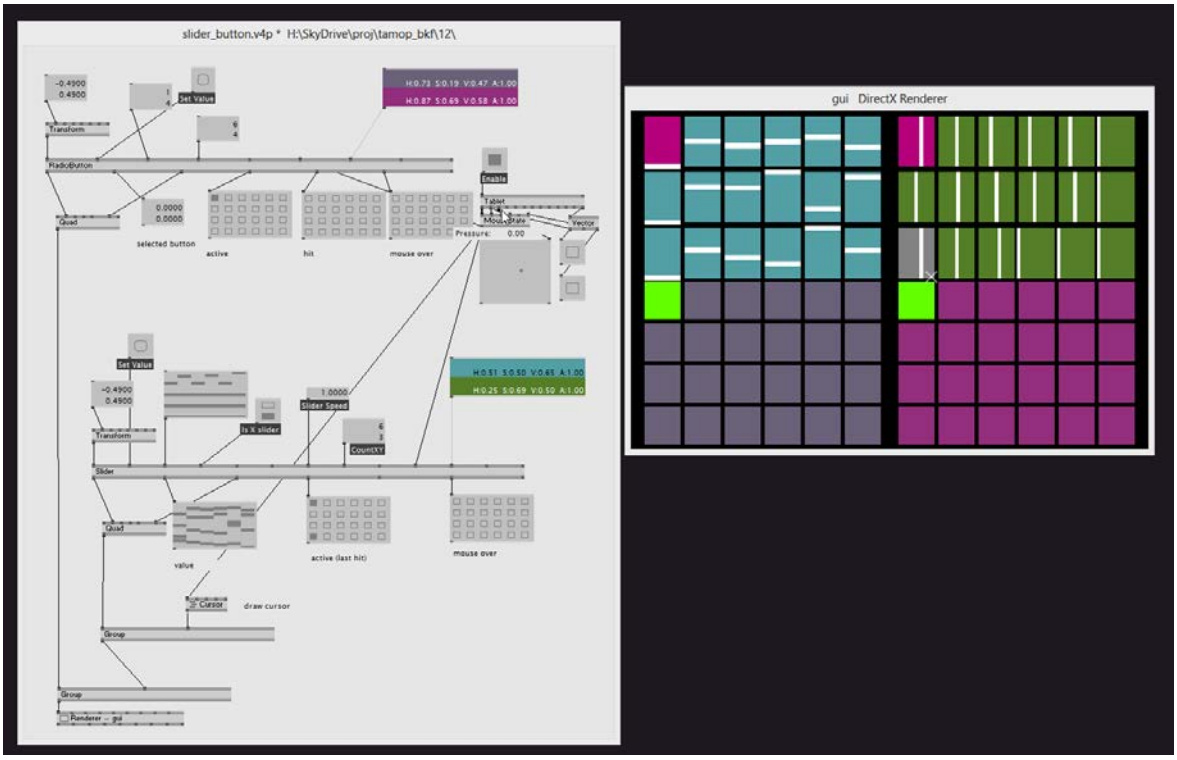
MidiController modullal részletesebben is megismerkedhetsz a mellékelt *KorgNanoKontrol.v4p* fájlon belül.

12.3. ábra: Wacom tablet

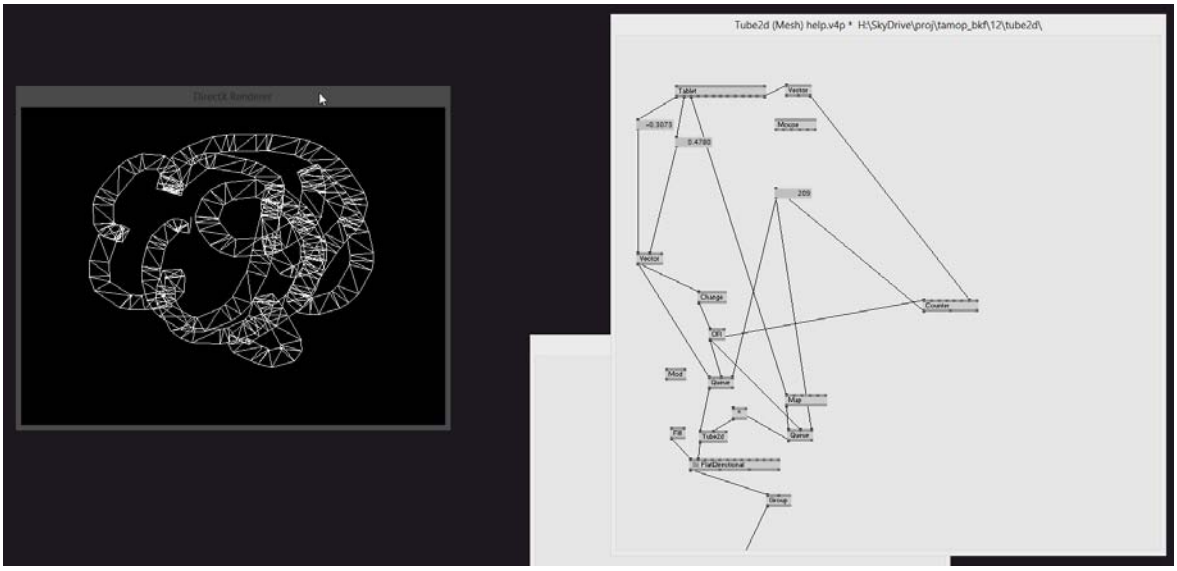
A következő típusú vezérlési lehetőség a Wacom Tablet, a modul neve pedig egyszerűen *Tablet*. Az interneten számos olyan projektet lehet találni, amik ezen az eljáráson alapulnak.

Ezek közül a legpraktikusabb az, amikor egy saját kezelőfelületet készítesz a tablethez. Ezt úgy tudod megcsinálni, hogy a toll *X Y* tengelyen lévő pontos pozíciójának adatait veszed alapul, és adott területeket határozol meg, amelyekhez más és más funkciókat tudsz rendelni: ez lehet egy sima slider, *X Y* slider, gomb vagy kapcsoló is. Ha kész vagy, a behatárolások kinyomtatott térképét a régebbi típusú tableteknél alá tudod tenni a rajzolási felület védőfóliájának, az új típusúaknál pedig matricákkal jelezheted ugyanezt a felületen, hogy mindig pontosan lásd, mikor mit vezérelsz. A vezérlőnek a gombokon kívül további funkciókat a nyomásérzékelés, a radír és a dőlésszög adatainak bevonásával adhatsz (a dőlésszöget viszont sajnos csak a drágább, Intuos típusú tableteken tudod elérni).

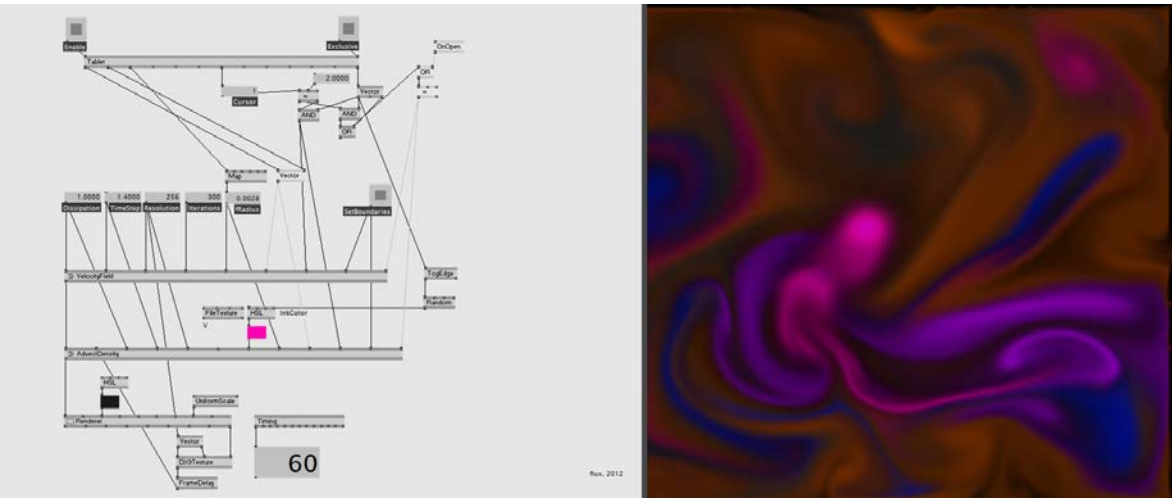
Ezenkívül a tabletes vezérlés alkalmas lehet még rajzolásra, valamint különböző effektek kezelésére is. A példák között ezekből kettőt mutatunk be, az egyik egy egyszerű, nyomásérzékeny rajzolós patch a *Tube2d* könyvtárban, a másik pedig a *Vux FluidSolverGPU* patch tabletre átalakított verziója.



12.4. ábra: Tablet gombok és sliderek



12.5. ábra: Tablet Tube2d patch



12.6. ábra: Tablet Fluid patch

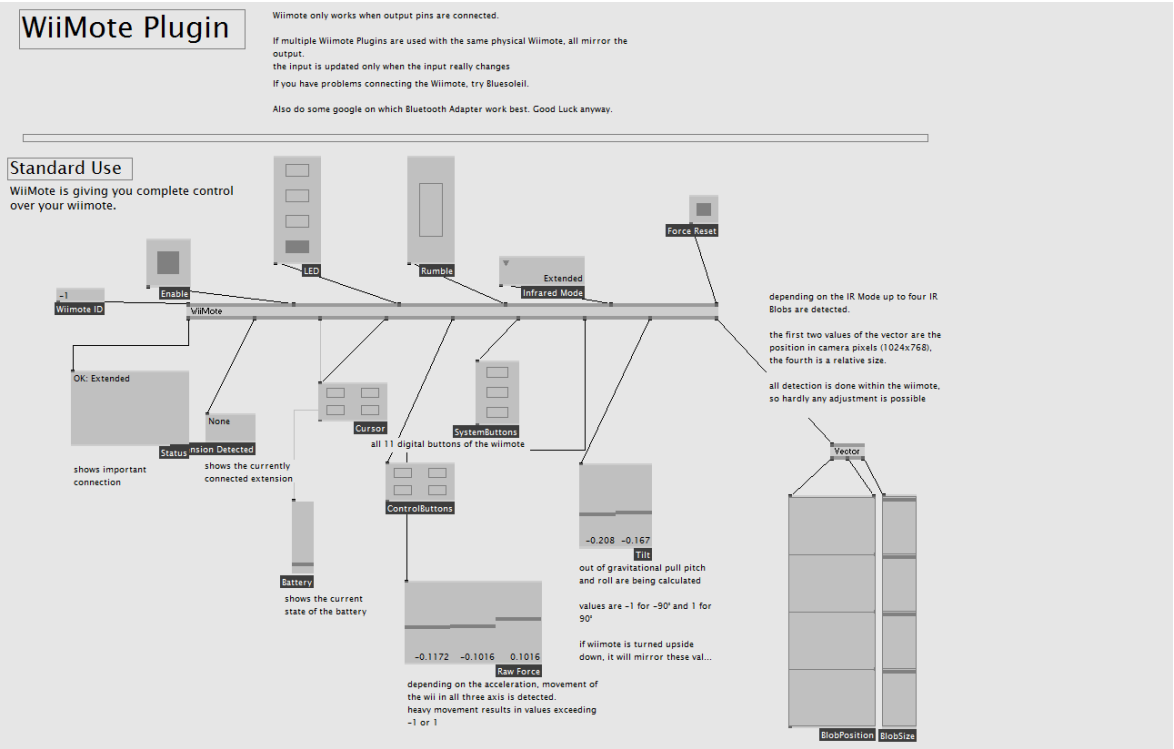
A harmadik vezérlési lehetőség a *Wiimote*, ami a mindenki által ismert a Nintendo Wii eszközt használja. Ehhez a legtöbb hack-et Johnny Lee készítette, és nagy részük az eszköz elején található infrakamerán alapul, ami eredetileg arra szolgált, az eszköz térbeli helyzetének pozicionálását segítse a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

12.7. ábra: Wiimote

A Wii egyszerre 4 pontot tud csak követni, ami nem sok, de bizonyos esetekben nagyon hasznos tud lenni. A telepítése nagyon egyszerű, mivel csak csatlakoztatni kell a géphez Bluetooth-on keresztül. Amikor rákérdez a gép, a jelszó helyét üresen kell hagyni, a többi pedig már megy magától, mivel a program automatikusan feltelepíti a szükséges drivereket.

12.8. ábra: Infratoll Wiimote-hoz

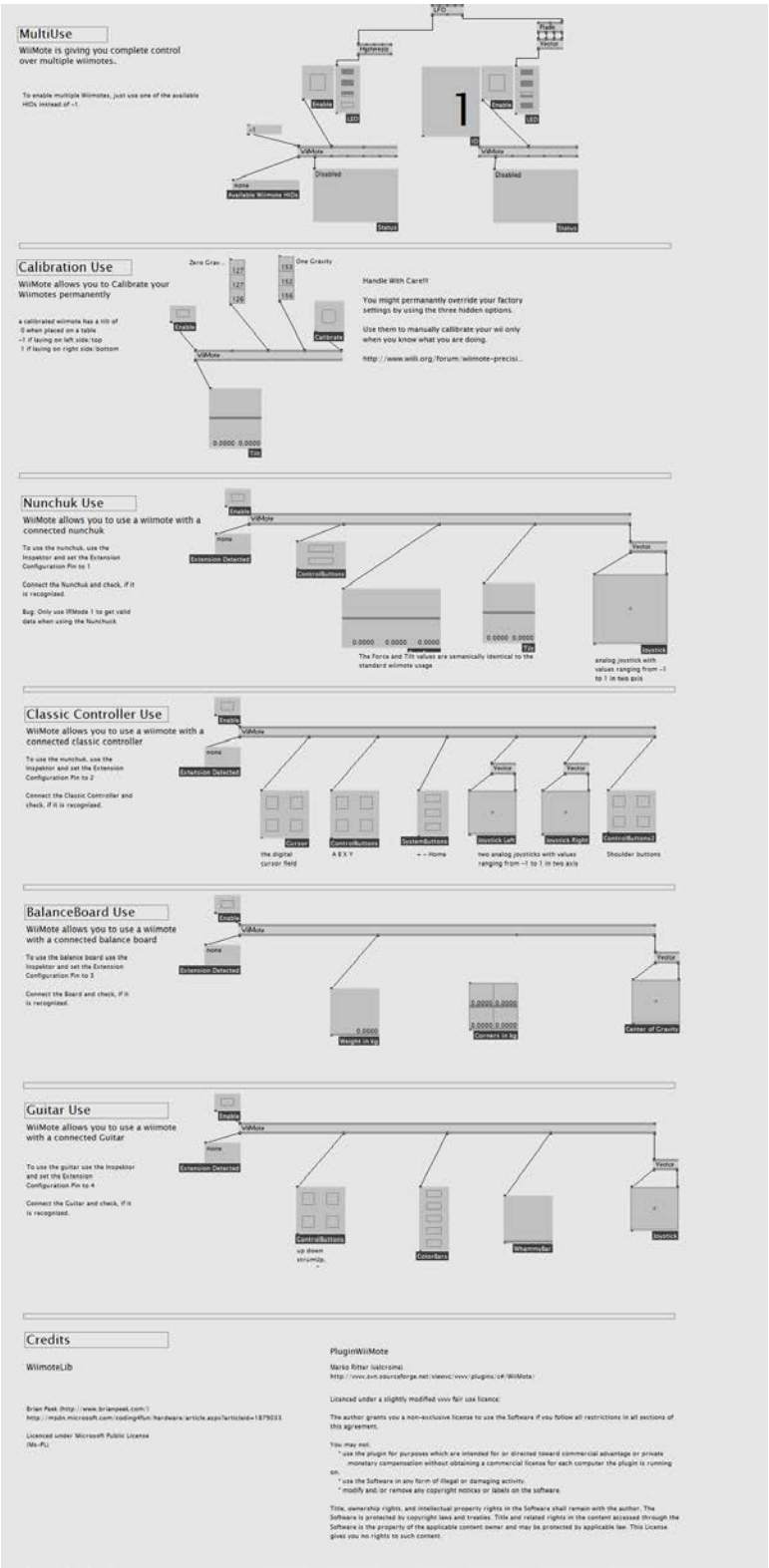
A telepítést követően a *VVVV*-ben ki kell keresni a *WiiMote* nevű modult, amin a *Led* bemeneten tudod kapcsolgatni a ledeket, a *Rumble*-ön pedig az elrendezésüket beállítani, illetve az eszközt rezgetni. Ez utóbbi segítségével az eszközt akár távvezérlésre is használhatod, ha kellően kísérletezős típus vagy, és kiforrasztod az eszközből a ledeket. Ekkor egy tranzisztor segítségével más külső eszközök vezérlésére is alkalmassá válik, illetve a rezgőmotorral azt is meg lehet oldani, hogy mondjuk egy színpadon szereplő előadónak metronómot küldesz, vagy egyszerűen csak jelzel vele, ha éppen váltás van.



12.9. ábra: Wii mote plugin

A *VVVV*-ben egyszerre több Wii mote is használható, ehhez nem kell mást tenned, mint hogy az eltüntetett *Id* bemenetet bekapcsolod az *Inspektor*-ban, és kiválasztod, hogy melyik eszközt szeretnéd használni.

A kimenetek között megtalálod a navigációs gombokat, az A, B, 1, 2 vezérlőgombokat, a három rendszergombot, az accelerométer adatait, a dőlést és végül az infrapozíciókat. A *Help* fájlban számos további bekötési variáció is található a különböző felhasználási lehetőségeknek megfelelően.



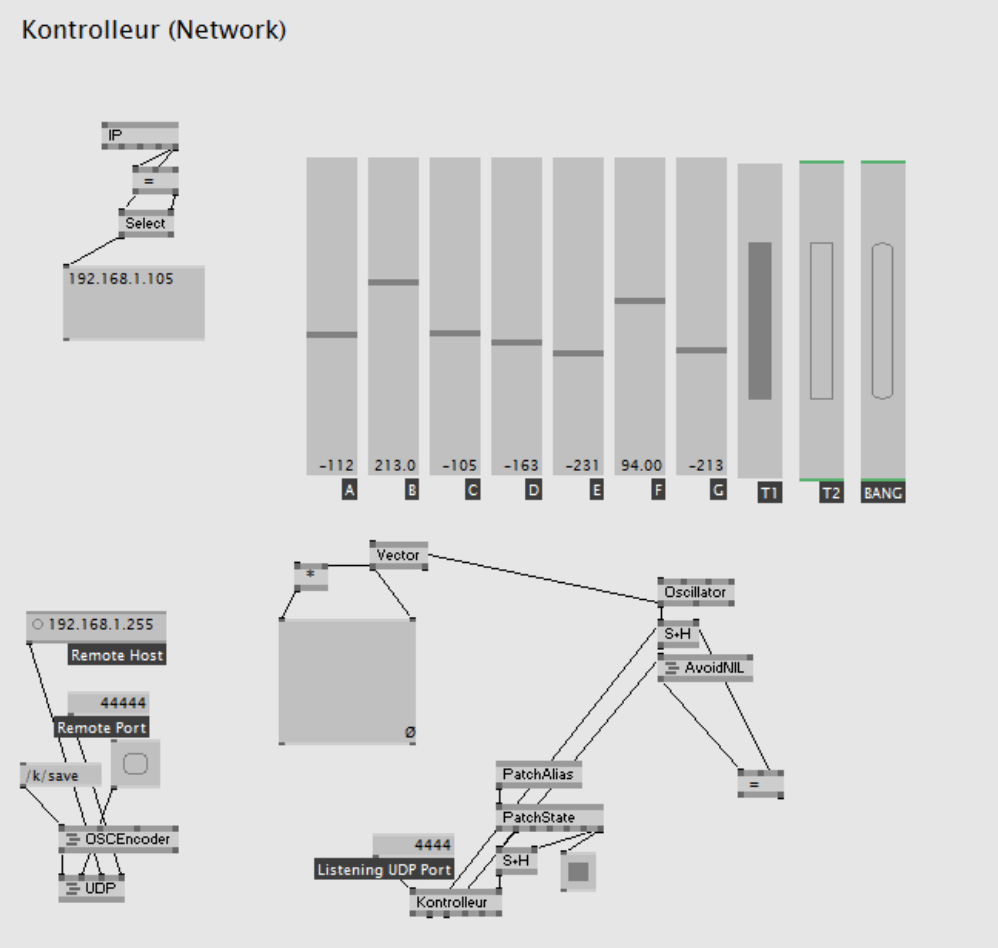
12.10. ábra: Wiimote egyéb beállítások

Végezetül az Android rendszerű eszközökön futtatható vezérlést nézzük meg, ami egy OSC-alapú kommunikációs módszer. Ebből egyelőre még csak a béta verzió elérhető, és csak ICS-en működik stabilan.

Az alkalmazás neve *Kontrolleur*, és a Google Play rendszerén belül a következő linkről telepíthető:
https://play.google.com/store/apps/details?id=processing.test.kontrolleur&referrer=utm_source%3Dbestappsmarket.com%26utm_medium%3Dbestappsmarket.com%26utm_campaign%3DappPage

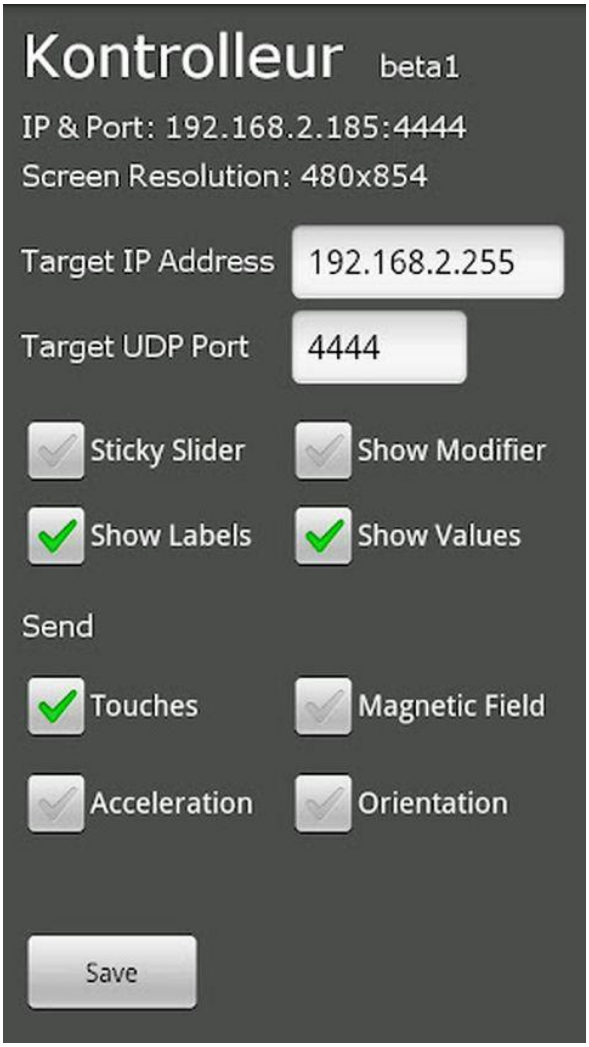
12.11. ábra: Kontrolleur kezelőfelülete

Az aktuális verziót *.apk* kiterjesztéssel megtalálod a csatolt könyvtárban is. Ennek a modulnak az a lényege, hogy amikor a *VVVV*-ben létrehozol egy számdobozt (ez lehet akár slider, akár egy rendes szám), és utána nyomsz egy CTRL+K billentyűkombinációt, az adott számdoboz ugyanúgy megjelenik a telefonos alkalmazásban is, oda-vissza vezérlést téve lehetővé. Vagyis ha a gépeden elkezded a számdoboz értékét változtatni, az érték a telefonon is változik, a telefonról pedig ugyanúgy meg tudod változtatni a programon belüli értéket. A modullal egyszerre több számdobozt is összeköthetsz a telefonnal, és ha ezeket elnevezed *Inspektor*-ban, az adott mező a telefonon is ugyanazt a nevet kapja.



12.12. ábra: Kontrolleur patch

A modul beállításakor fontos, hogy a bejövő és kimenő portok különbözőek legyenek, másképpen nem érkeznek meg az üzenetek rendesen. Ha véletlenül nem tudod a telefonod pontos IP-címét, nyugodtan írd be az utolsó hármas mezőbe a 255-ös számot (vagyis a teljes IP-cím így 192.168.1.255), amivel a program az összes IP-címre elküldi az adott üzenetet. Az alkalmazás további hasznos funkciója, hogy a telefonon külön kiválasztható az is, hogy milyen további paramétereket küldjön el a számítógépnek: ilyenek lehetnek az accelerométer adatai, az érintési pozíciók (ez lehet egyszerre több ujj is), a mágnesszenzor adatai (ezt dokkoláskor használja a telefon) vagy az orientáció.



12.13. ábra: Kontrolleur beállítása

13. Az Arduino és elektronikai ismeretek

A gyártók már jó ideje forgalmaznak számos programozható, és egyben programok futtatására képes IC-t, amelyeknek minden szükséges ki- és bemenete előre ki van vezetve az eszközből, ezek azonban meglehetősen drágák a hétköznapi felhasználók többsége számára. Az Arduino, az olasz Massimo Banzi fejlesztése erre nyújt megoldást: az eszköz leegyszerűsített felépítése és a hozzá kifejleszett, bárki számára könnyen használható szoftverkörnyezet az ebből következő alacsony árral együtt igencsak népszerűvé tette a művészet és a hobbielektronika területéről érkező felhasználók körében.

Az Arduino voltaképpen egy sok ki- és bemenettel rendelkező, egyszerű számítógép alacsony processzorteljesítménnyel és alacsony fogyasztással. Elsősorban arra szokás használni, hogy kapcsolókkal, potméterekkel (vagyis állítható ellenállásokkal), különféle szenzorokkal és egyéb eszközökkel motorokat, kijelzőket, világítást, hangszórót vagy akár teljes számítógépeket vezéreljenek. Sokszor használják ruhatervezők is intelligens ruhák gyártásához: erre a célra kifejlesztettek már egy külön Arduino-típust is a hozzá való szenzorokkal, amiket áramvezető cérnával lehet felvarrni az adott ruhára. De ugyanígy szokták használni 3D-nyomtatáshoz, robotok készítéséhez vagy zenéléshez is.

Eleinte csak pár variáció volt elérhető az Arduino-ból: a normál méretű Duemilanove, a Bluetooth-os verzió és a Mini, viszont mivel az egész hardver Open Source alapon működik és bárki fejlesztheti, mostanra már rendkívül sokféle típus elérhető a piacon. Számos cég forgalmaz már direkt Arduino-ra kifejlesztett saját kiegészítőket is. Mivel az eszköz PIN kiosztása szabványos (vagyis a különböző generációk között nincs eltérés), ezért kiegészítőként a legkülönbözőbb úgynevezett shield-eket párosíthatjuk hozzá, akár egyszerre többet is, sőt magunk is készíthetünk ilyet.

Az Arduino képes standalone módban is futni, így nem szükséges hozzá folyamatos kapcsolat a számítógéppel. Ehhez viszont az eszköz saját szoftverében kell megírnod hozzá a programot, amit letölthetsz a hivatalos oldalukról (http://Arduino.cc/en/Main/Software). Ez a program az AVR C nevű nyelvet használja,

ami egy leegyszerűsített C programnyelv. A kész programot ezzel a szoftverrel tudod feltölteni magára az Arduino-ra, ahol a program az eszközben található Atmel Atmega chipben tárolódik, és egyben ott is fut le.

Ha számítógépes vezérlésre van szükséged, azt soros porton keresztül tudod megoldani, mivel az Arduinonón található egy USB-soros port átalakító, ami kommunikál az Atmel Atmega IC-vel.

A soros porton keresztüli kommunikációhoz is szükséges programot feltölteni, amire többféle lehetőséged van. A legegyszerűbben használható és szinte minden lehetőséget magában foglaló megoldás a Firmata nevű program (http://firmata.org/wiki/Main_Page). Általában ezt használja az összes node-alapú szoftver, mint például a Max-MSP, a PureData vagy a VVVV. Ebben egyszerűen ki tudod választani, hogy a ki- és bemeneteid milyen típusúak legyenek a használattól függően (PWM, Servo, Digital vagy Analogue). Az egyetlen hátránya, hogy a sok lehetőség miatt a folyamatos kommunikáció eléggé lelassítja a válaszidőt, ami bizonyos esetekben problémát okozhat, ha a másodpercenként csak pár üzenet érkezik be a számítógépbe.

Ezért ha nagyon gyorsan akarsz adatokat lekérni az Arduino-ból, érdemes saját soros programot írni rá, ami csak azt a szükséges parancsot hajtja végre, amit te szeretnél, így nem terheli le az Arduino-t más műveletekkel. A saját programok készítésében nagy segítséget nyújthat a szofverben található példaprogramok tárháza, amiben rengeteg hasznos komment is található. Ha elakadsz programozás közben, az adott problémának érdemes utánanézni a fórumokon, mert nagy esély van rá, hogy ugyanezzel a problémával már mások is foglalkoztak, és ki is találták rá a megoldást, ami alapján viszonylag egyszerűen tudod összerakni a saját programodat is.

Itthon sajnos nem igazán lehet beszerezni az eredeti Arduino terméket, csak az AvrDuino nevű másolatot:

(http://avr.tavir.hu/modules.php?name=Content&pa=showpage&pid=80), ezért ha az eredetivel akarsz dolgozni, azt külföldről kell megrendelned. Európán belül megbízható szállítónak számít a Farnell (http://hu.farnell.com/Arduino), az RS-Online (http://hu.rs-online.com/web/generalDisplay.html?id=Arduino) és a Proto Pic (http://proto-pic.co.uk/categories/Arduino/). (Az RS-Online és a Farnell egy-két napos szállítási határidővel dolgozik, míg a Proto Pic másfél hét alatt szállít, amivel érdemes számolni, ha határidős munkát csinálsz éppen.) Ezenkívül lehet próbálkozni még eBay-jel is, viszont fontos, hogy ebben az esetben ne a legolcsóbb kínai változatot vedd meg, mert ezeknél a nyák kivitelezése és az Arduino-ra forrasztott alkatrészek minősége nem a legmegbízhatóbb, így nagyon hamar tönkremehet az eszköz. A fentieken túl a

legnagyobb gyártó a Sparkfun (https://www.sparkfun.com/pages/Arduino_guide), akik saját Arduino-kat, hozzá való shield-eket, és rengeteg szenzort és kiegészítőt is forgalmaznak, ráadásul számos hasznos tutorialt és termékismertetőt is találhatsz az oldalukon. Végül pedig az is egy lehetséges alternatíva, hogy saját magad gyártod le az Arduino-t, ehhez viszont már Atmel programozóval is kell rendelkezned. Ebben az esetben, ha a nyákot egy nyákgyártó céggel készítteted el, akár 2000 Ft-os darabáron is készíthetsz Arduino-t.

Ha elindítod az *Arduino* patchet, akkor láthatod, hogy az első bemeneten lehet beállítani, hogy a használni kívánt PIN be- vagy kimenet legyen-e, a másodikon azt, hogy milyen típus legyen, az utolsó bemeneten pedig a soros port pontos számát (amit a Vezérlőpult eszközkezelőjében tudsz leellenőrizni).

Az Arduino alapverzióján a ki- és bemenetek két sorban helyezkednek el. A fölső sorban 12 db digitális ki/bemenet van, amikből a hullámjellel jelöltek a PWM-képes kimenetek, vagyis amikkel LED-et vagy szervomotort tudsz vezérelni, a maradék hatnál pedig csak ki-be kapcsolni lehet őket. Az alsó sorban 6 db analóg bemenetet, 2 földet, egy 5 voltos és egy 3,3 voltos tápkábelt találsz. Ha például LED-et szeretnél villogtatni, akkor azt egy ellenálláson keresztül bármelyik felső sorban található kimenetre rákötheted.

A LED-nek alapesetben két lába van, egy hosszabb és egy rövidebb. A rövidebbik láb jelöli a földet, míg a hosszabbik a pozitívat, de ezt úgy is meg lehet különböztetni, hogy felülnézetből a LED-nek az egyik oldala nem teljes, az jelöli a föld oldalt. Mindegyik LED-nek saját referenciafeszültsége van, ami azt jelöli, hogy hány volttól hány voltig működik. Ezt fontos betartani, mert ha túl alacsony a feszültség, valószínűleg nem fog világítani, viszont ha túl nagy (ami lehet akár 5 V is), akkor pedig egyből leéghet.

Ezért mindenképpen ellenállást kell használni a LED megfelelő feszültségének beállításához. Szerencsére, erre létezik egy weboldal (http://led.linear1.org/1led.wiz), ahol ez egyszerűen kiszámolható, és a bekötéshez pontos rajzot is generál az oldal. Ezt az oldalt érdemes használni akkor is, ha sok LED-et akarsz egyszerre bekötni, és ehhez pontos útmutatásra van szükséged.

A piros LED-ek nyitó feszültsége általában 2,4 V, míg a többi színé 3 V és 3,3 V közé esik, ezért fontos tudni a LED típusát. Ehhez az interneten találsz Datasheeteket, amik pontosan megadják a különböző LED-típusok paramétereit. A másik fontos tényező, hogy a LED-nek mennyi az áramfelvétele: ezt általában milliamperral, vagyis „mA”-val szokták jelölni. Az Arduino-nak mindegyik kimenete 5 V-os és 40 mA terhelést bír, így 40 mA-nél nagyobb fogyasztású LED-et nem szabad közvetlenül rákötni. Ha mégis szeretnél ennél nagyobb áramfelvételű fogyasztót rákötni, azt már

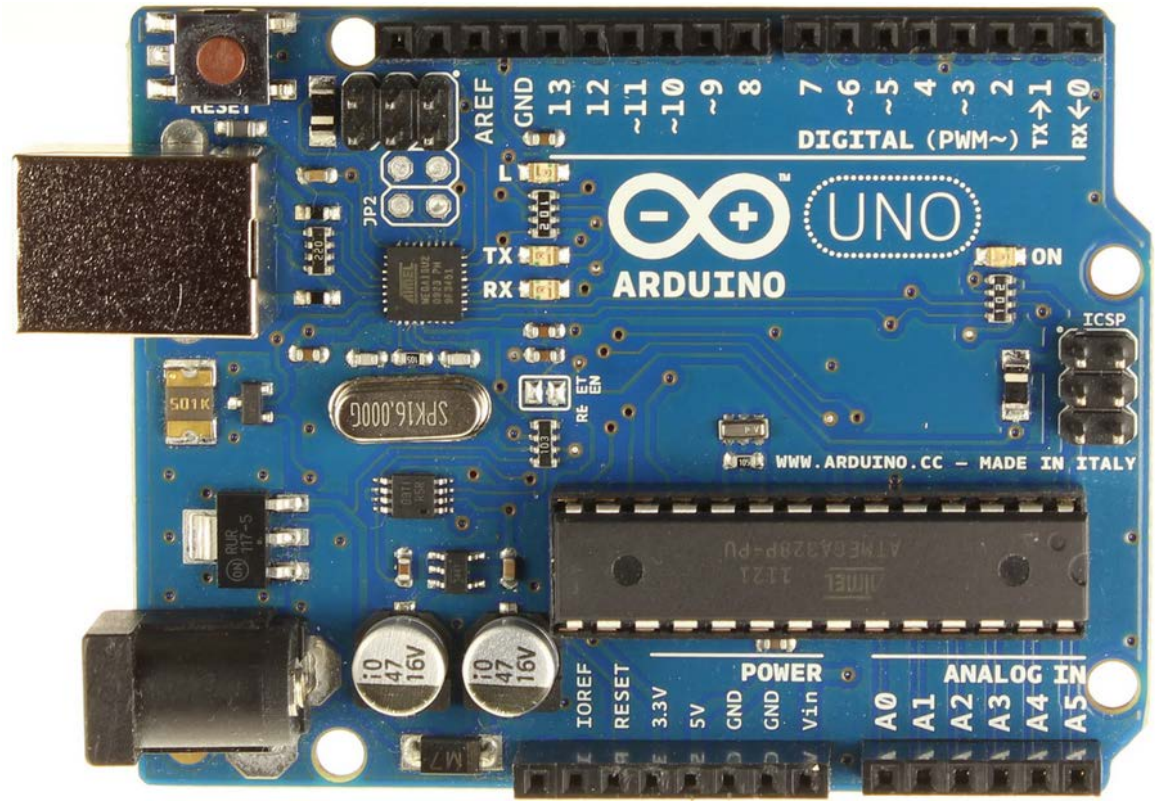
külső áramellátással és tranzisztorral kell megoldanod, de ennek mindenképpen nézz utána, mielőtt belefognál!

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

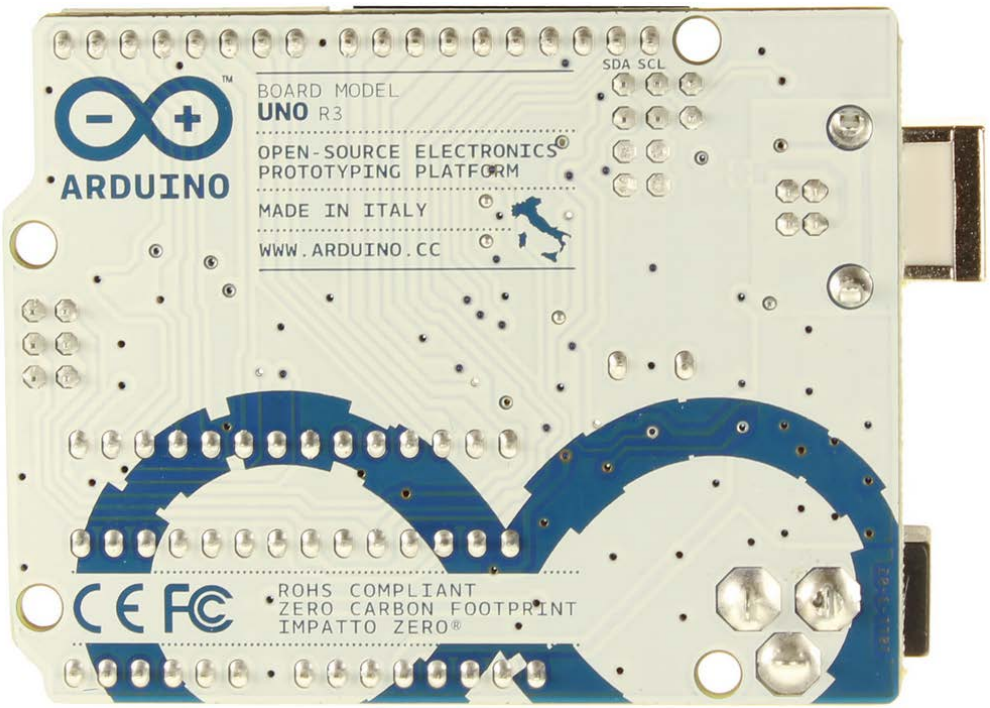
Kapcsoló használata esetén szintén ellenállást kell használni, mivel alaphelyzetben az Arduino bemenete „lebeg”, vagyis nem minden esetben tudja eldönteni, hogy a kapcsolónk éppen milyen állásban van. Ezt úgy tudod kiküszöbölni, ha az alapállapotot egy 10 kohm-os ellenállással leviszed a földre. Ennek 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 rákötni, míg a középsőt egy tetszőleges *Analogue Read*-be, amelynek elkészítésében segít a lentebb látható rajz.

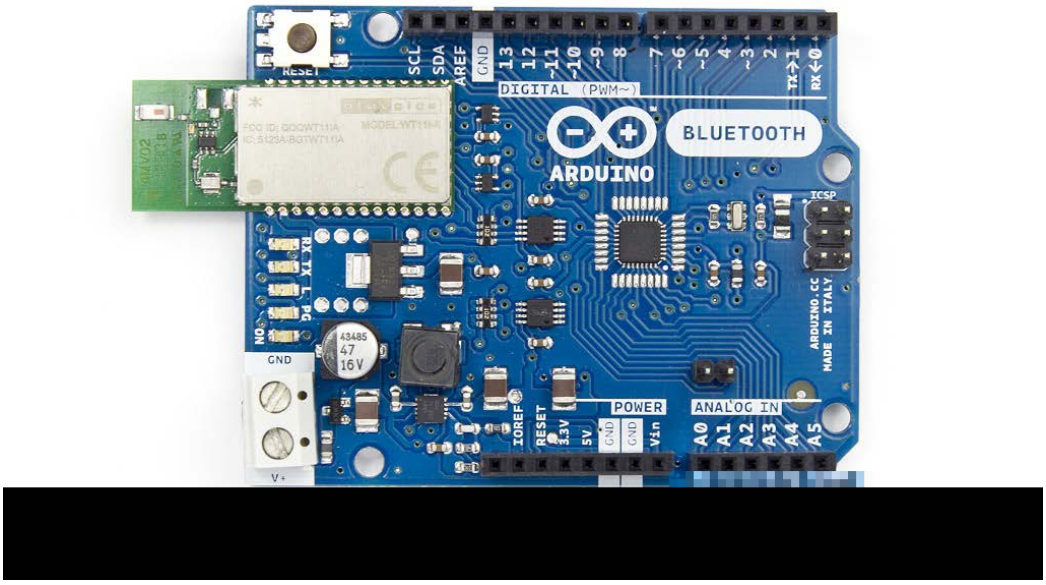
Ezek alapján már akár saját MIDI-keverőt is tudsz készíteni.



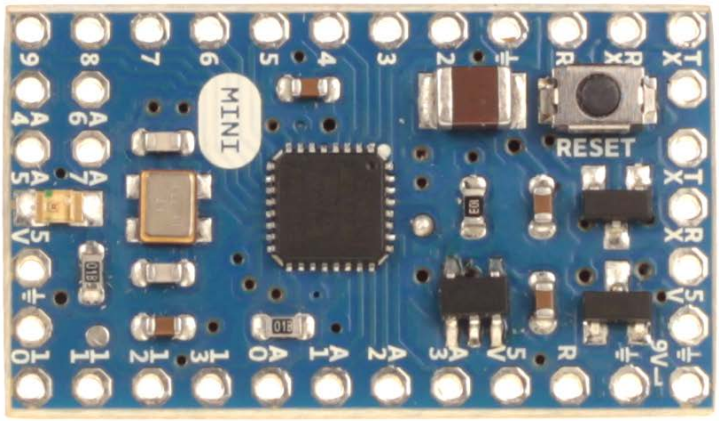
13.1. ábra: Alap Arduino



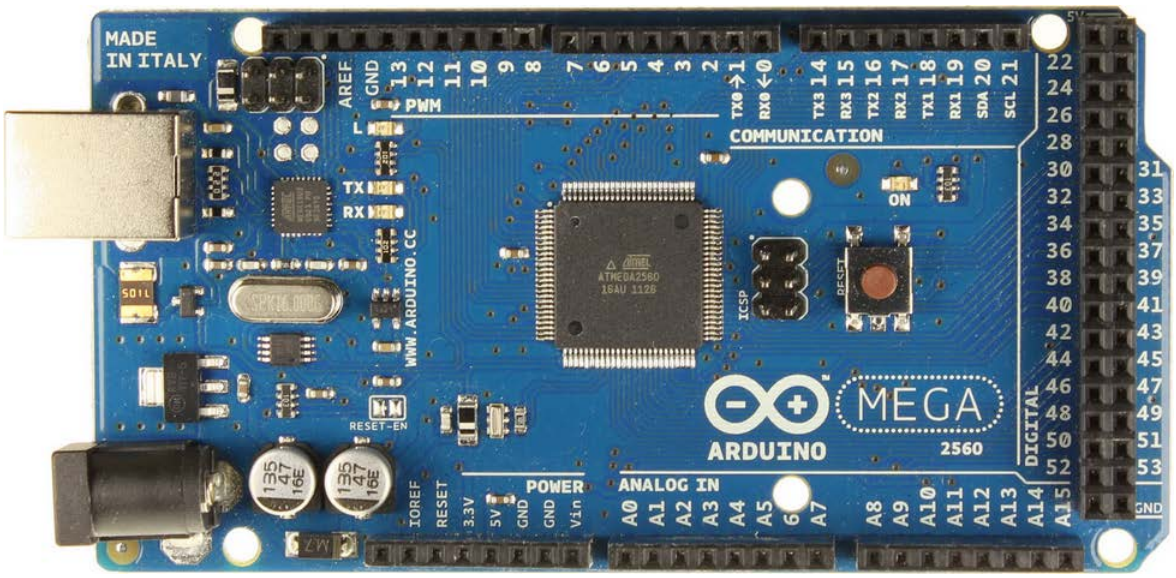
13.2. ábra: Arduino hátlapja



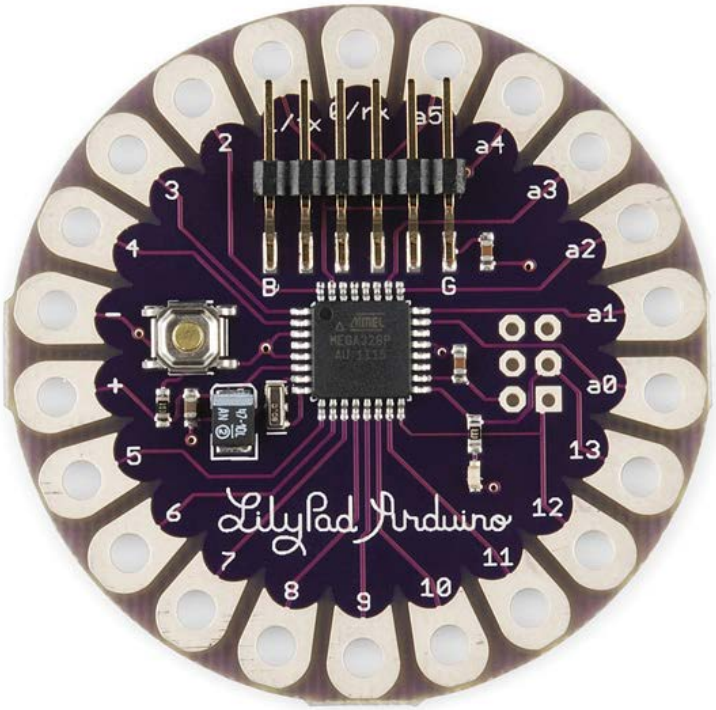
13.3. ábra: Bluetooth Arduino



13.4. ábra: Arduino Mini

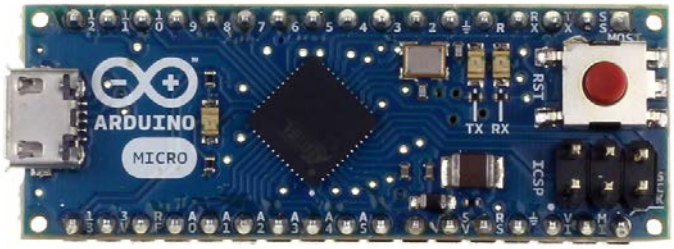


13.5. ábra: Arduino Mega

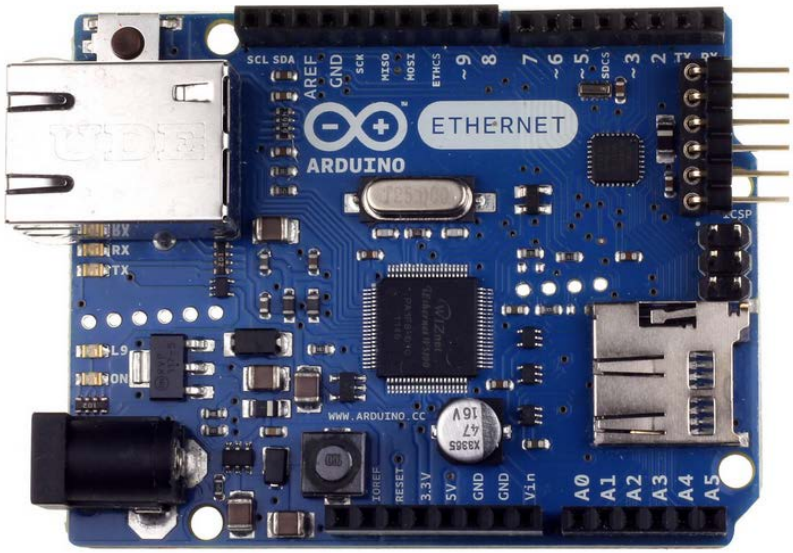


13.6. ábra: Lilypad

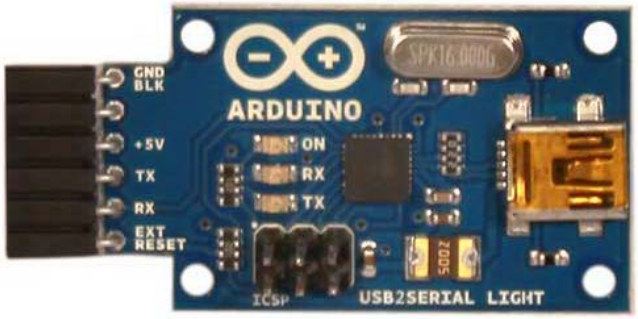
13.7. ábra: Arduino Fio Zigbee-hez



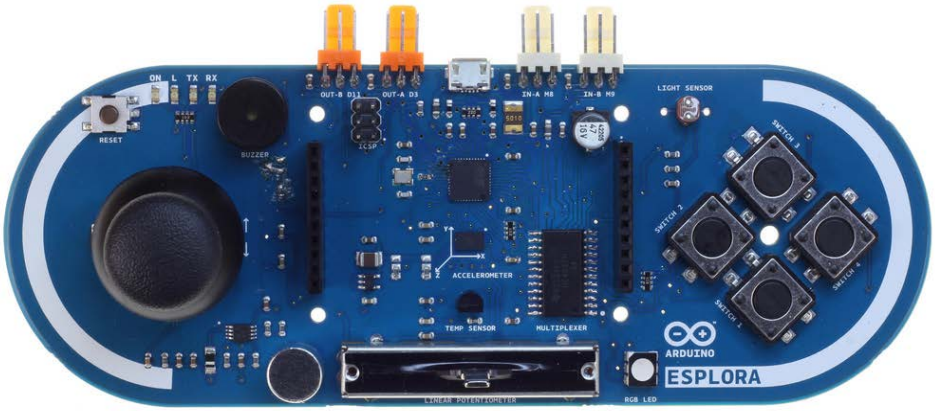
13.8. ábra: Arduino Micro



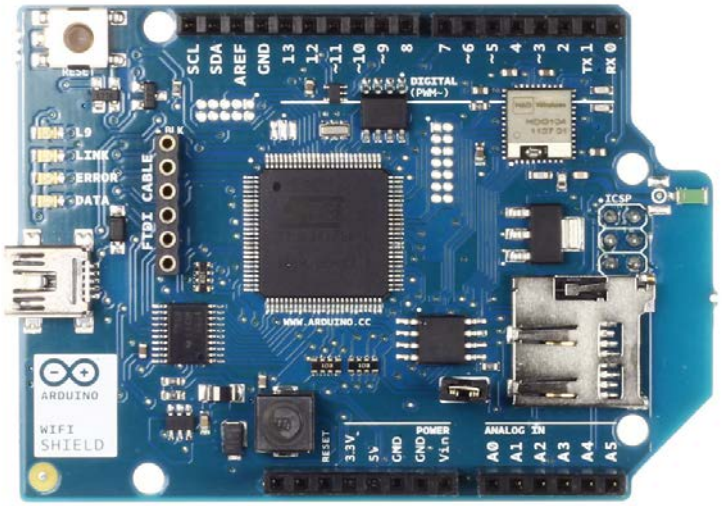
13.9. ábra: Arduino Ethernet



13.10. ábra: USB soros port



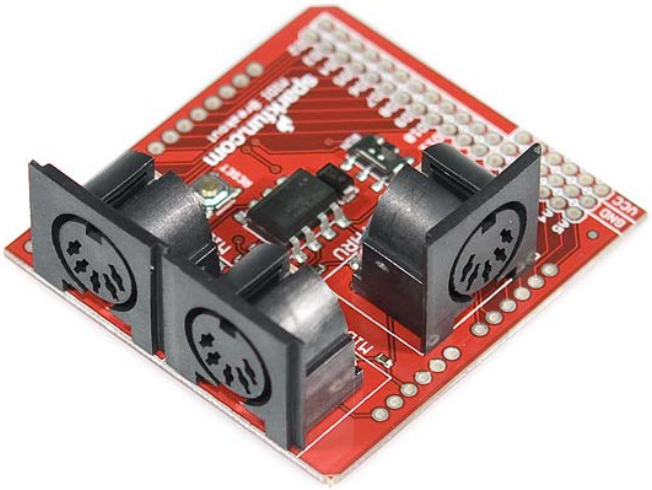
13.11. ábra: Arduino Game



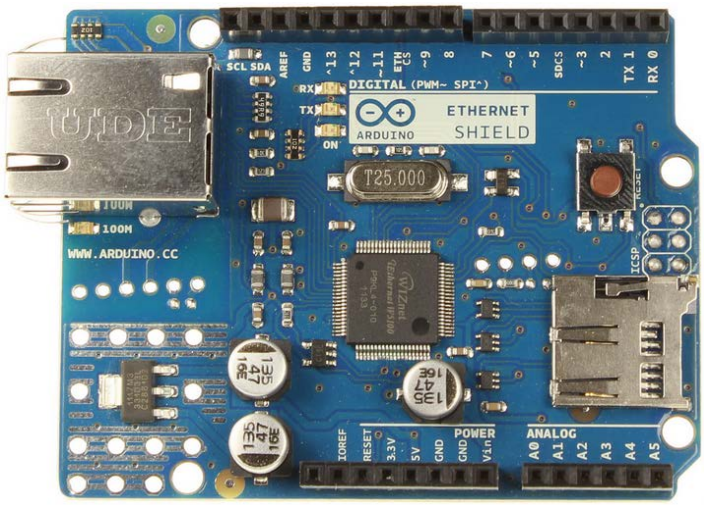
13.12. ábra: Wifi shield

13.13. ábra: GSM shield

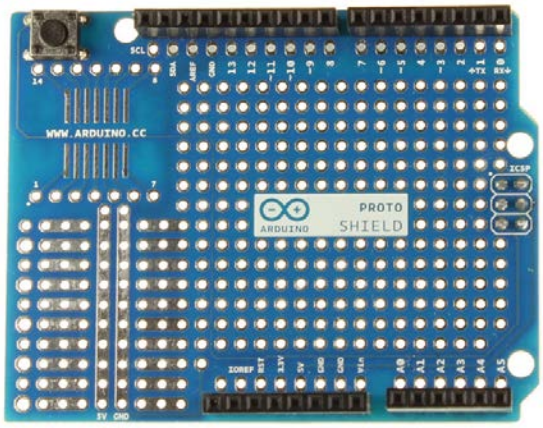
13.14. ábra: Lilypad Mp3



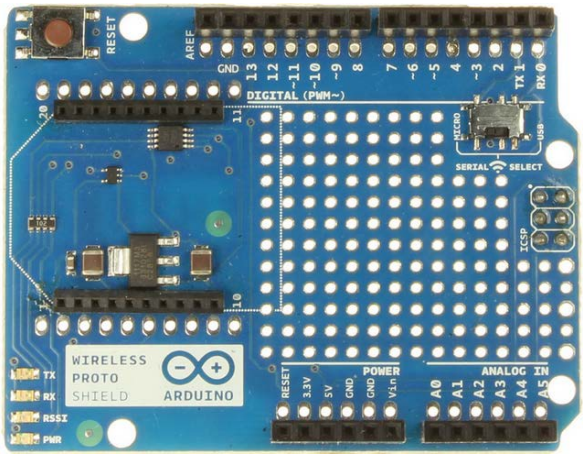
13.15. ábra: Midi shield



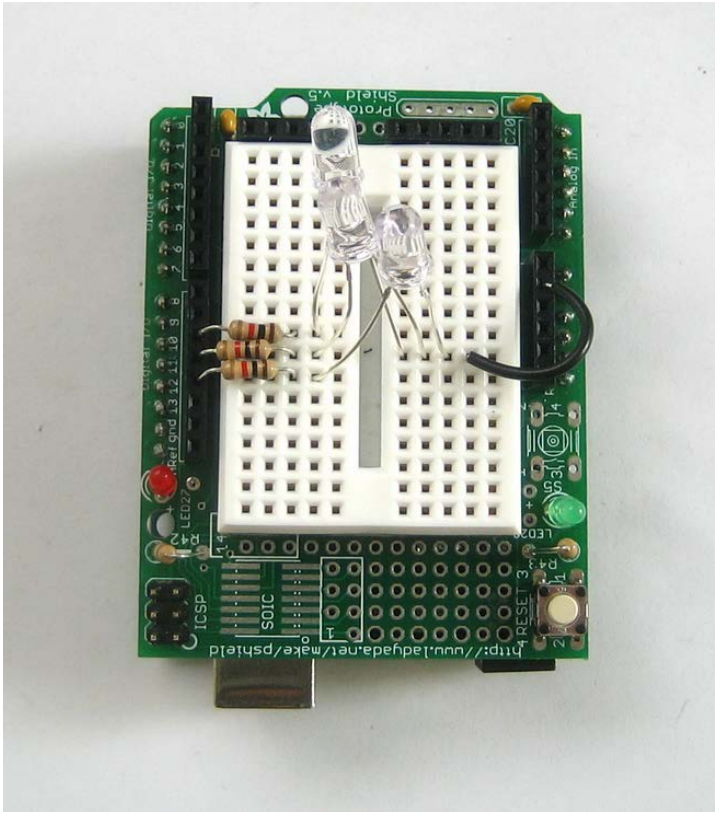
13.16. ábra: Ethernet shield



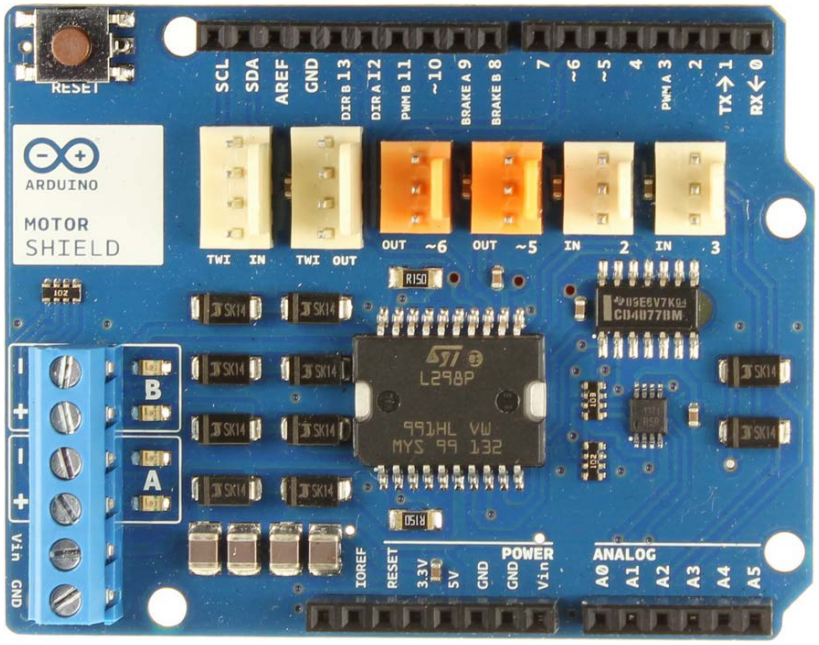
13.17. ábra: Proto shield



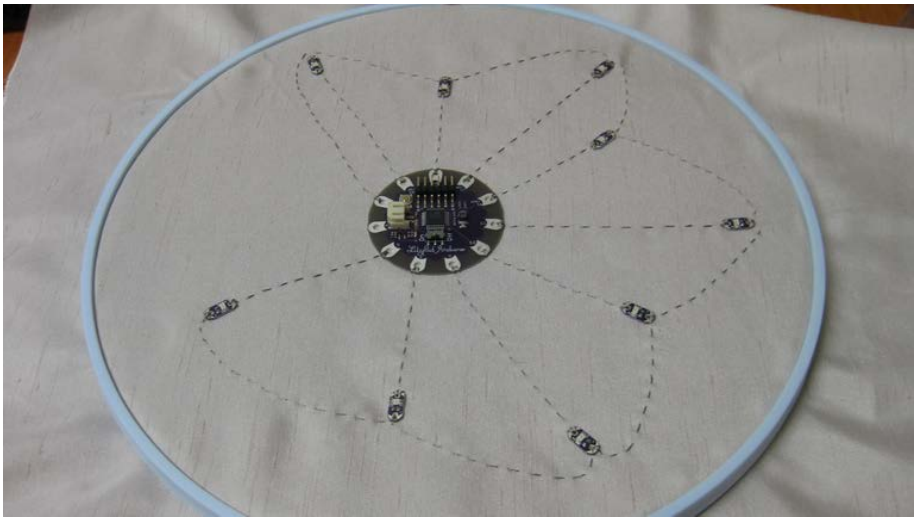
13.18. ábra: Zigbee shield



13.19. ábra: Breadboard shield



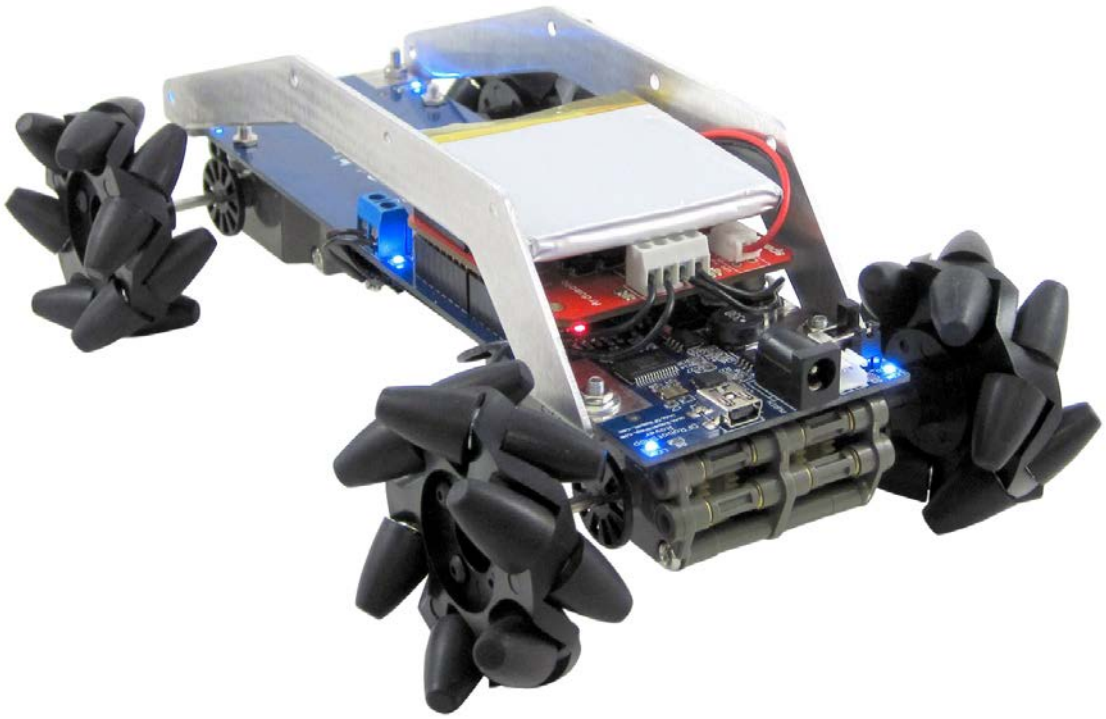
13.20. ábra: Motor shield



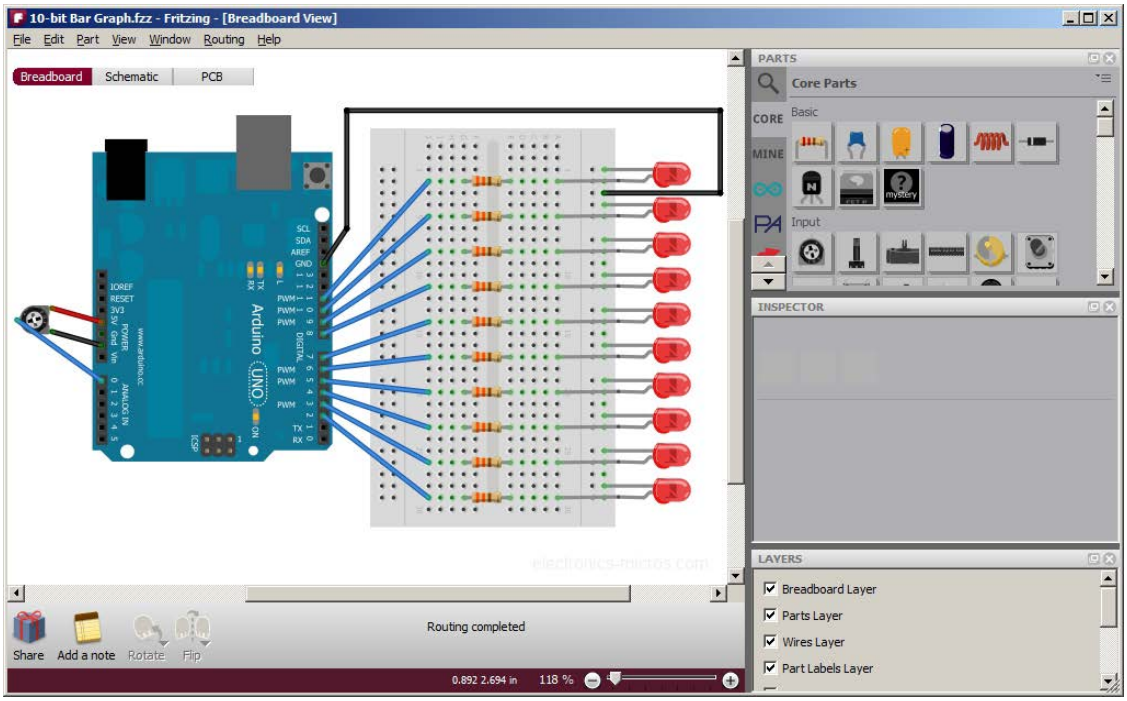
13.21. ábra: Lilypad használatban

13.22. ábra: Lilypad climate dress

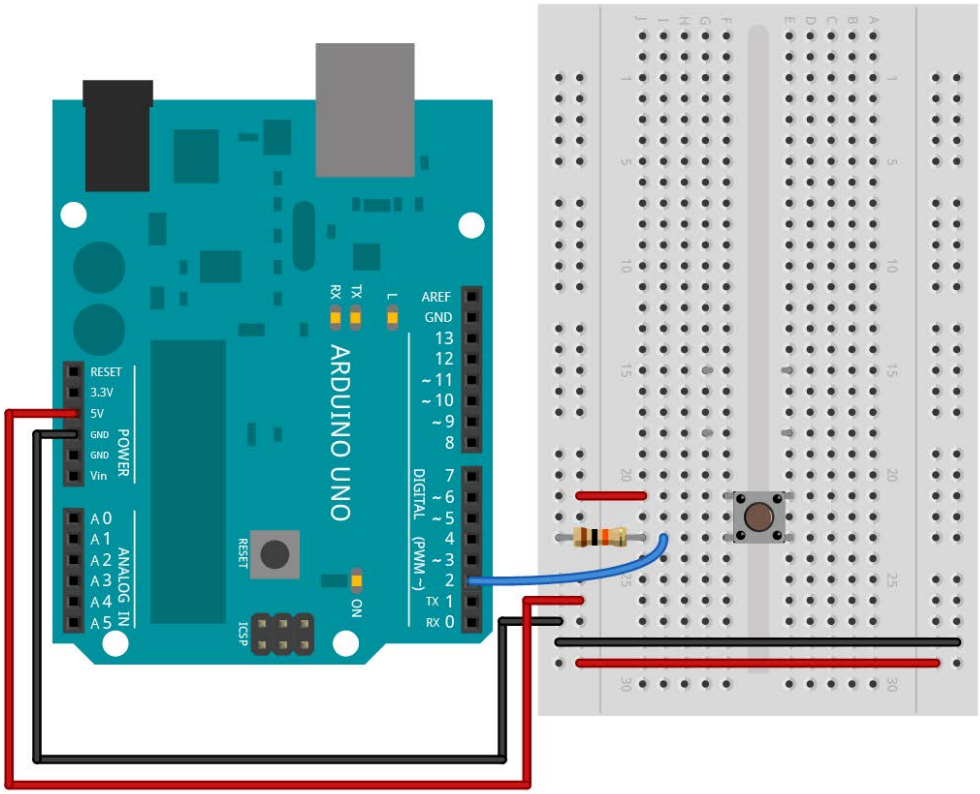
13.23. ábra: Lilypad climate dress



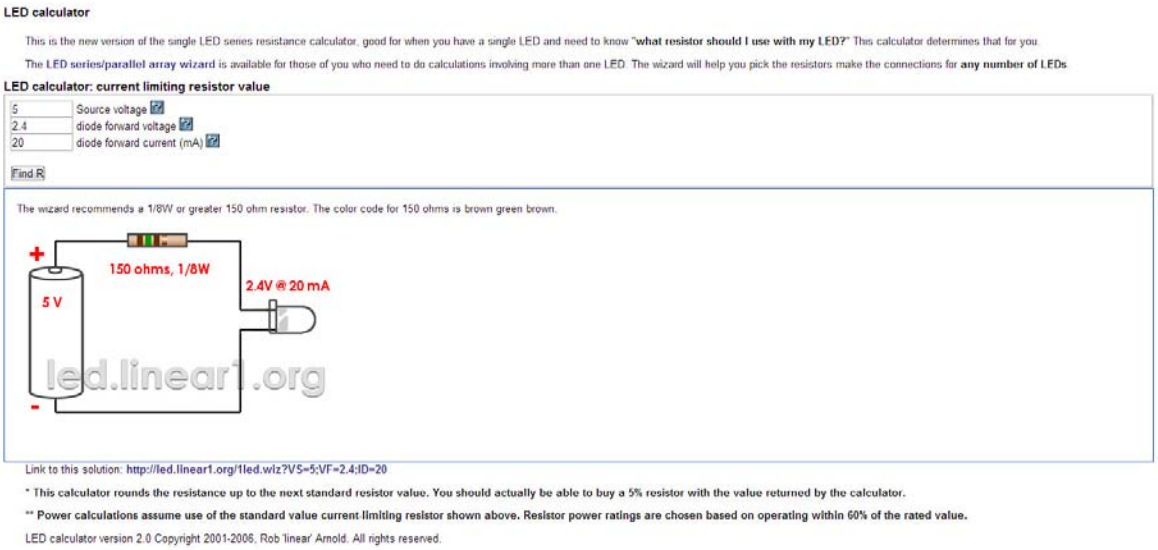
13.24. ábra: Arduino robot



13.25. ábra: Arduino Fritz

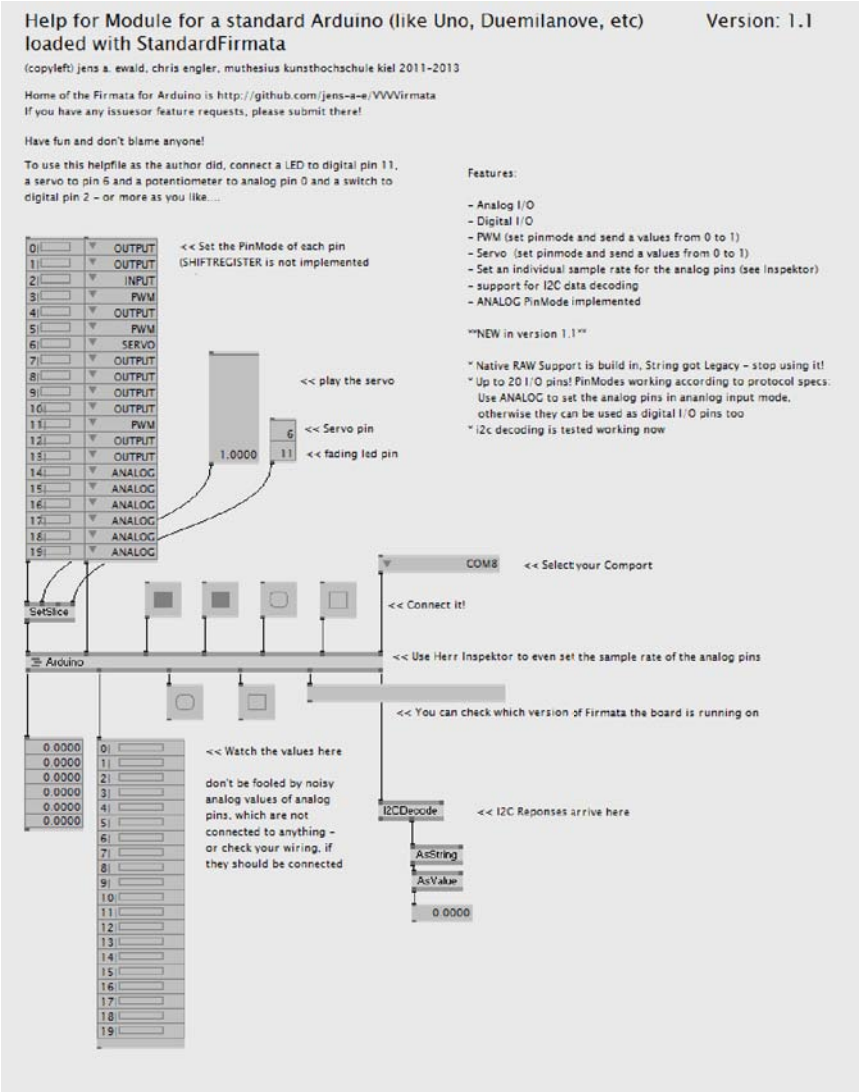


13.26. ábra: Arduino bekötési minta

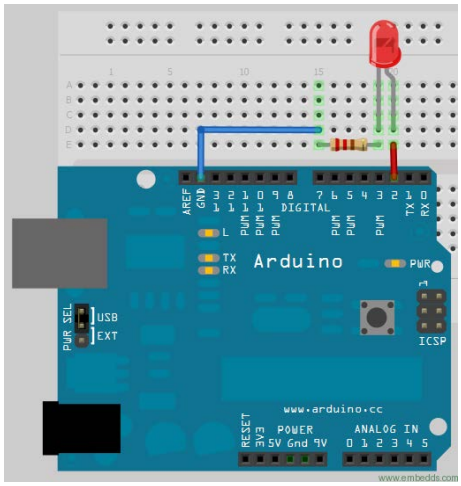


13.27. ábra: Led calculator

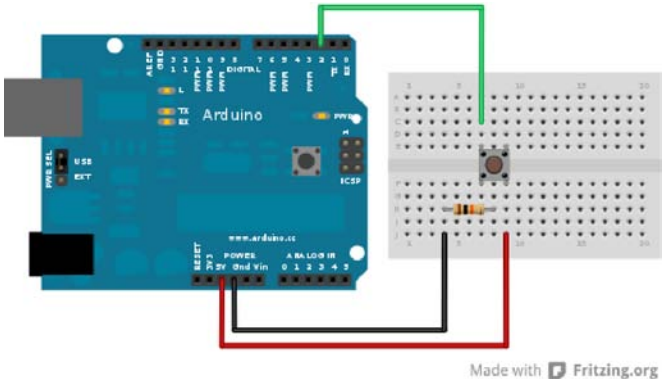
13.28. ábra: Led calculator több ledre



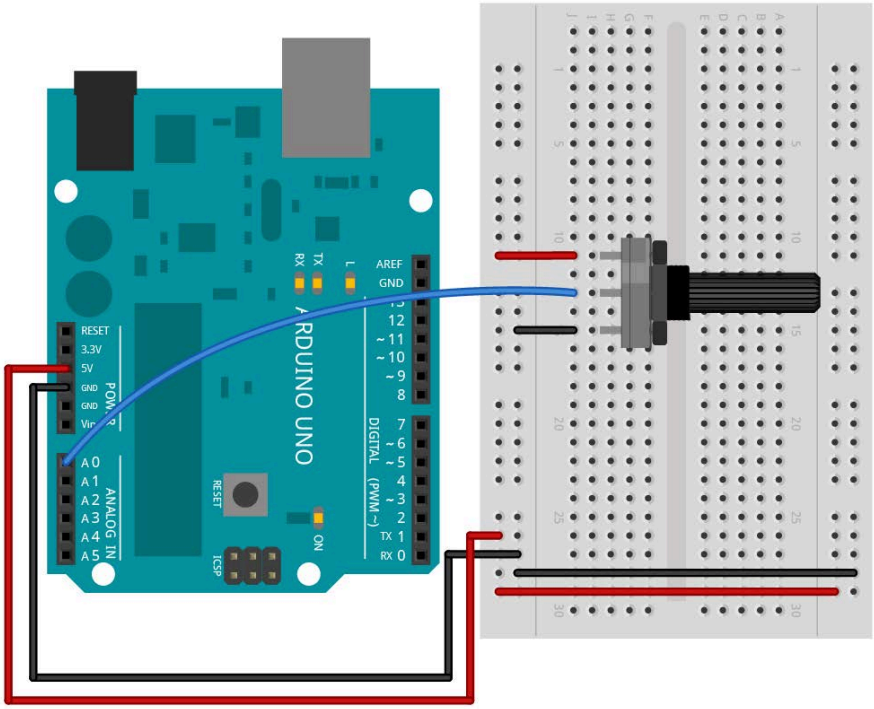
13.29. ábra: Arduino kezelőfelülete



13.30. ábra: Arduino Led bekötése



13.31. ábra: Arduino kapcsoló bekötése



13.32. ábra: Arduino potméter bekötése

14. A gépek közti kommunikáció és különböző vezérlési lehetőségek

Az előző leckékben, a *Kontrolleur* bemutatásakor már találkozhattál az OSC vezérléssel. Ez az OSC UDP alapokon nyugszik, ami egy nagyon gyors kommunikációs fajta: ennél a módszernél a gyorsaság miatt nem garantált az üzenetek megérkezése. Ha inkább az üzenetek biztonságos megérkezése számít, arra a TCP kommunikációs forma a legmegfelelőbb.

Az Android Marketen és az Apple AppStore-ban nagyon sok OSC-képes vezérlőprogramot lehet találni – ezek közül az egyik legjobban használható multiplatform program a TouchOSC. Ezt akár két gép közötti kommunikációra is használhatod, mivel szinte mindegyik programozási nyelvhez van OSC decoder/encoder, TCP cliens/server és UDP cliens/server kiegészítő.

Az OSC üzenettípusok lehetnek tizedes vagy egész számok, illetve szövegalapúak. Ez utóbbit jelezni kell az üzenet elküldésekor, amit a *Type Tags* fül alatt tudsz elvégezni, az *s = string f = float i = integer* beállításokkal. Ez azért fontos, mert különben a fogadó oldal nem tudja dekódolni az adott üzenetet. Ezenkívül az üzenetek címét is be kell állítani, ami az üzenetek szeparálásához kell: például ha az egér *X* és *Y* pozícióját szeretnéd átküldeni, akkor érdemes az *Adress* fülbe beírni a */mouse_x* és */mouse_y* beállításokat egy *String* üzenetboxon keresztül.

Fontos, hogy a címzéseket a fogadó oldalon is be kell írnod, különben a program nem tudja szétszedni az üzenetet. Ezenkívül fontos még, hogy az UDP port beállításakor a port száma mindkét oldalon megegyezzen, és az IP-cím is pontos legyen. Ha esetleg folyamatosan változó IP-címmel dolgozol, a címet helyettesítheted 255-tel is, aminek köszönhetően a hálózaton belül minden IP-cím meg fogja kapni az üzeneted.

Ha esetleg Max-MSP-vel szeretnél több információt, vagy egy kisebb képet megosztani, arra használhatod a *Contribution* oldalon található *Max/Jitter Matrix Nodes* nevű modult is (http://vvvv.org/contribution/maxjitter-matrix-nodes). Mint a nevében is szerepel, ennek a modulnak segítségével Max-os mátrixot lehet küldeni,

illetve fogadni VVVV-ből. A VVVV oldalán az üzenet színinformáció formájában érkezik meg, amit egy *RGB Split*-tel tudsz szétbontani száminformációra.

A következő vezérlési fajta a DMX. Ezt a kommunikációs formát elsősorban a fénytechnikában használják dimmerek, illetve robotlámpák vezérlésére. Egy DMX vonalon 512 csatornát tudsz vezérelni, aminél többre csak akkor lehet szükség, ha robotlámpát, vagy sok RGB LED-et akarsz vezérelni.

Hagyományos izzós lámpák vezérléséhez dimmert kell használni, amiből létezik egy- és többcsatornás változat is, különböző teljesítményekkel. Ezekkel az eszközökkel elsősorban rendezvény és színházi környezetben találkozhatsz, ahol érdemes előre megkérdezni, hogy a dimmerek mire vannak lepatchelve, hogy azzal ne a helyszínen kelljen foglalkoznod. A robotlámpák saját dimmerrel rendelkeznek, így legalább az áramellátásuk már meg van oldva, viszont sokkal több csatornát használnak el, mivel a szervomotorokhoz, a shutterhez, a tükör- és színváltóhoz és a többi funkcióhoz mind-mind külön csatorna szükséges. Léteznek olyan robotlámpák is, amik több száz csatornát használnak el, de azokat már Artnettel vezérlik, ami szintén egy DMX-alapú kommunikációs forma, amiről később ejtünk szót.

Otthoni körülmények között a legkönnyebben elkészíthető és legolcsóbb megoldás, ha nagyteljesítményű LED modulokat és 12 voltos DMX PWM vezérlőt használsz. Itt fontos, hogy mielőtt elkezdesz LED-eket vásárolni, mindenképpen olvass utána a LED-ek hűtésének, illetve hogy csak a megfelelő feszültséggel hajtsd meg őket, különben szinte egyből leégnek.

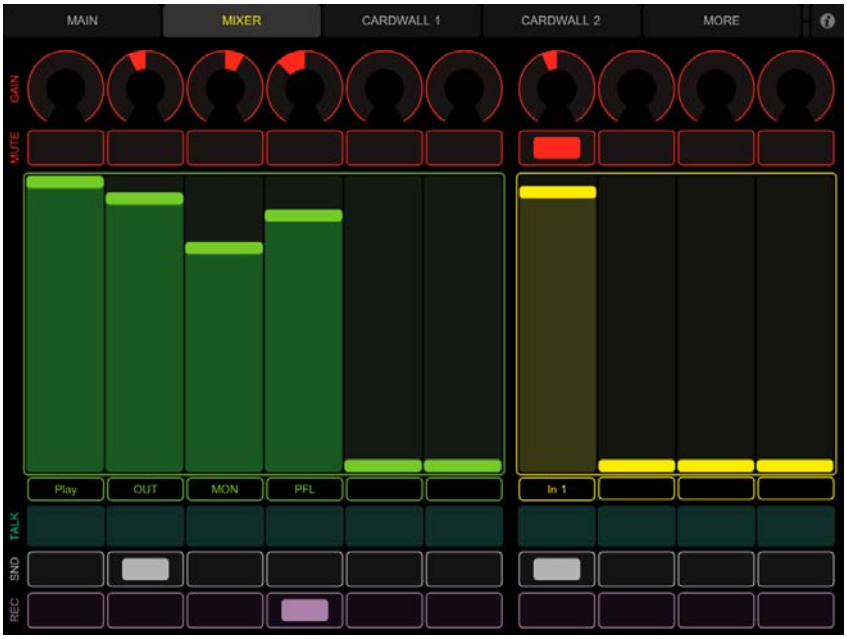
A különböző lámpaparkokat nagyon sokáig csak fénypultokkal lehetett vezérelni, melyeknek az ára akár a több millió forintot is elérhette. Napjainkra viszont már szerencsére eléggé elterjedt a lámpák USB-alapú DMX vezérlése, amihez mindössze egy laptopra van szükség, ráadásul bizonyos esetekben sokkal komplexebb vezérlést lehet ezzel elérni, mint a hagyományos pultokon. A legfontosabb USB-vezérlők gyártói az Ecue, a Sunlight és az Enttec – ezek közül az Enttec termékeit érdemes a leginkább használni, mert ár–érték arányban ők bizonyultak a legjobbnak, mivel a többi cégnél az eszköz mellé még a szoftvert is külön meg kell fizetni. Az Enttec sokféle USB-s eszközt gyárt, de ezek közül a pro kategóriásak működnek biztosan a VVVV-vel. Ha esetleg olcsóbban szeretnél hozzájutni Enttec-hez, akkor érdemes körülnézni eBay-en, mert egy olasz cég (http://www.splabs.it/?portfolio=usb-dmx-converter) készített egy vele kompatibilis verziót, amit a gép képes Enttec-termékként kezelni.

A csatolt fájlok között megtalálod az Enttec vezérléséhez szükséges drivert és a példafájlokat. Mivel az USB DMX egyfajta soros kommunikáció, ezért a Start menüben, a Vezérlőpulton belül az Eszközkezelő menüben meg kell nézni, hogy a DMX melyik soros portra lett installálva, és az alapján beállítani a VVVV-ben. Ezek után már csak

vezérelni kell a csatornákat, amit az első bemeneten tudsz megtenni egy vagy több *Setslice* segítségével, 0 és 1 között.

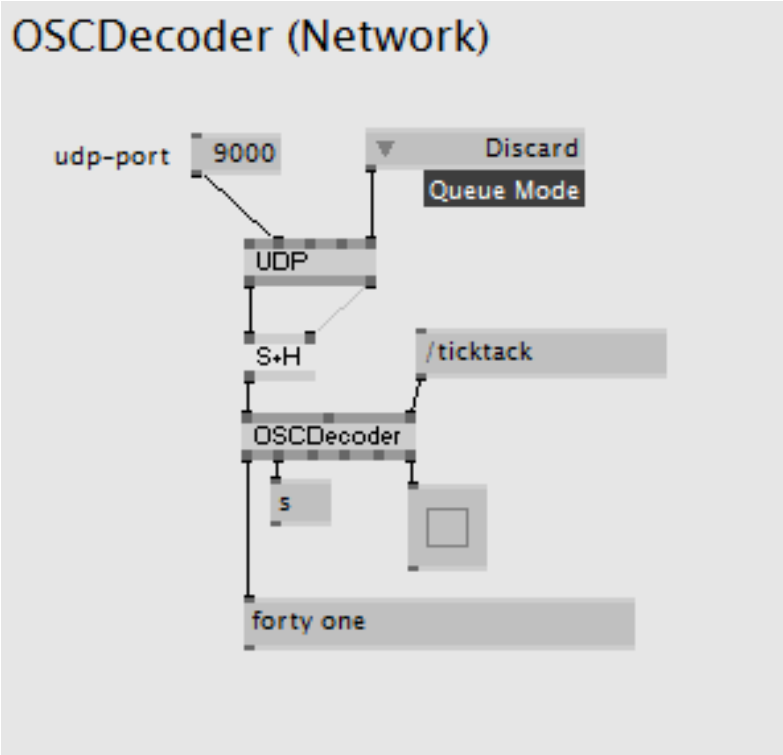
Végezetül következzen az Artnet. Az Artnet egy modernizált DMX-alapú kommunikációs forma. A legszembetűnőbb sajátossága, hogy itt mindegyik eszköznek saját IP-címe van, a vezérlés pedig UTP-kábelen keresztül történik. Ennek köszönhetően a számítógép szokásos ethernet kimenetén keresztül, egy hagyományos 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 robotprojektort alapból Artnettel szerelnek, így mindenképpen érdemes előre eldönteni, hogy a DMX vagy az Artnet-e a számodra legjobb megoldás. Ha nagyon sok lámpát, vagy sok csatornát használó robotlámpákat kell használni, akkor mindenképpen Artnet a legmegfelelőbb, mert így nem kell egyszerre több USB DMX átalakítót használnod. Egyedül az Artnet-DMX konverter kerül kicsit többe 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.

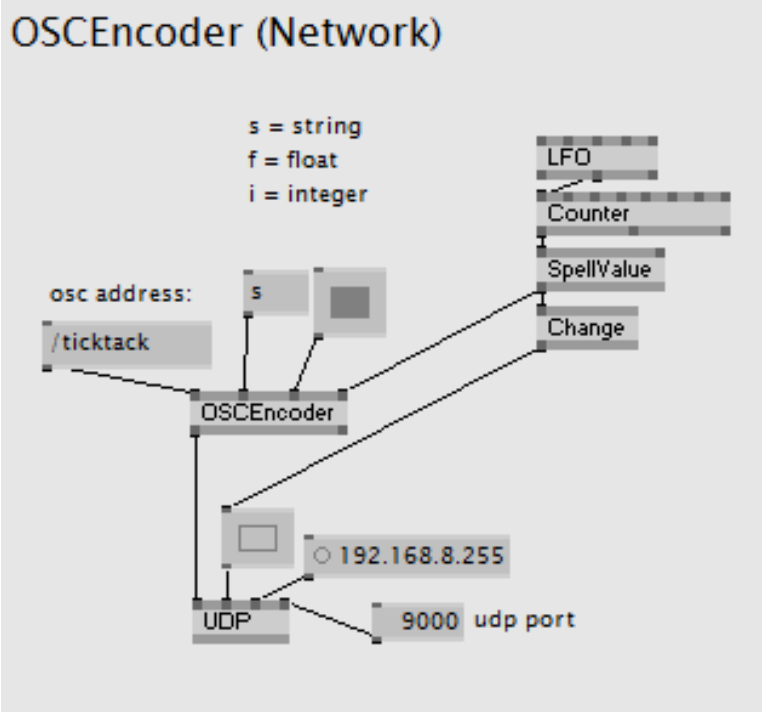


14.1. ábra: Touch Osc Ipaden

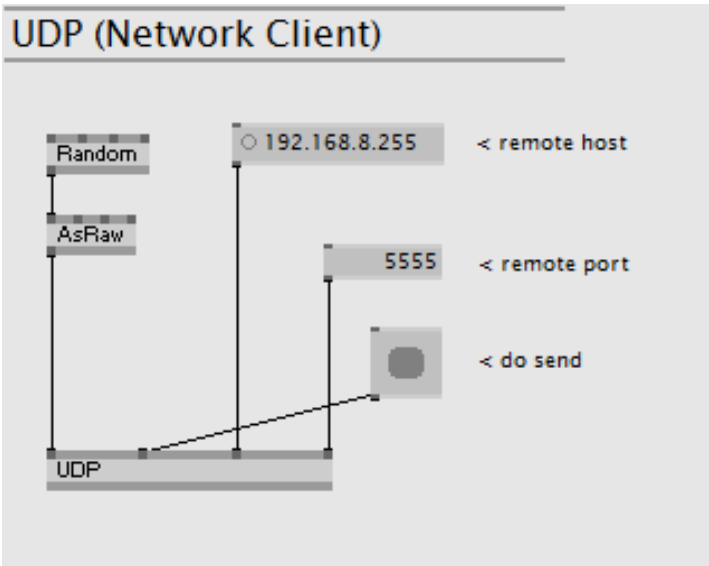
14.2. ábra: Touch Osc Iphone-on



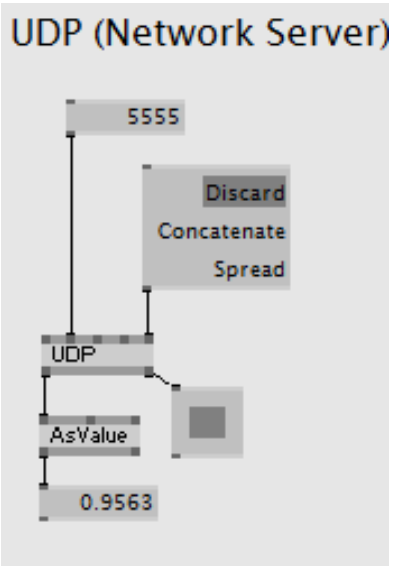
14.3. ábra: Osc fogadó



14.4. ábra: Udp küldő



14.5. ábra: Osc küldő



14.6. ábra: Osc fogadó

14.7. ábra: Sunlite USB DMX

14.8. ábra: Sunlite USB DMX

14.9. ábra: Enttec USB DMX Pro

14.10. ábra: USB DMX Pro klón

14.11. ábra: USB DMX Pro klón

14.12. ábra: Szokványos dimmer

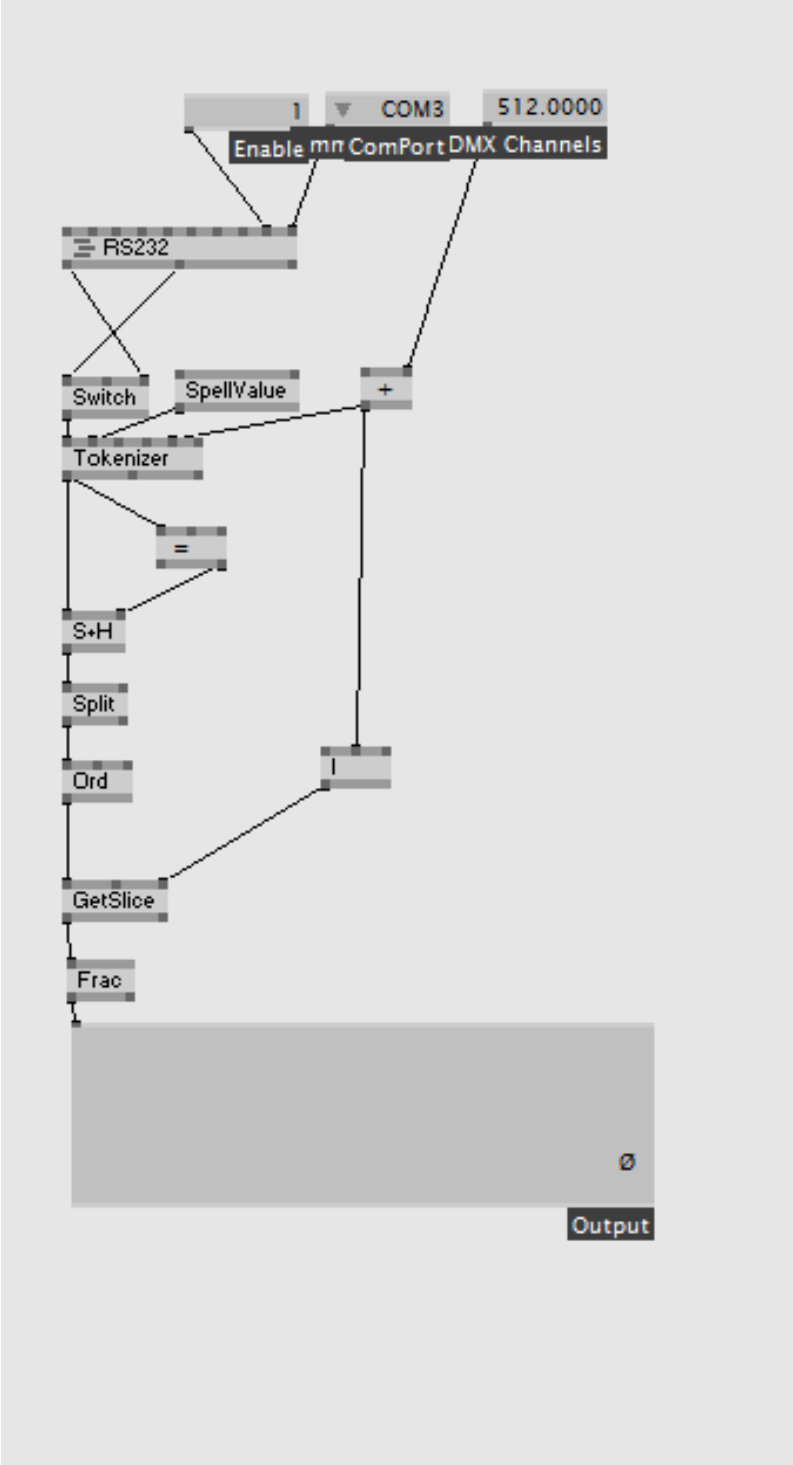
14.13. ábra: Compulite fénypult

14.14. ábra:High End Hog fénypult

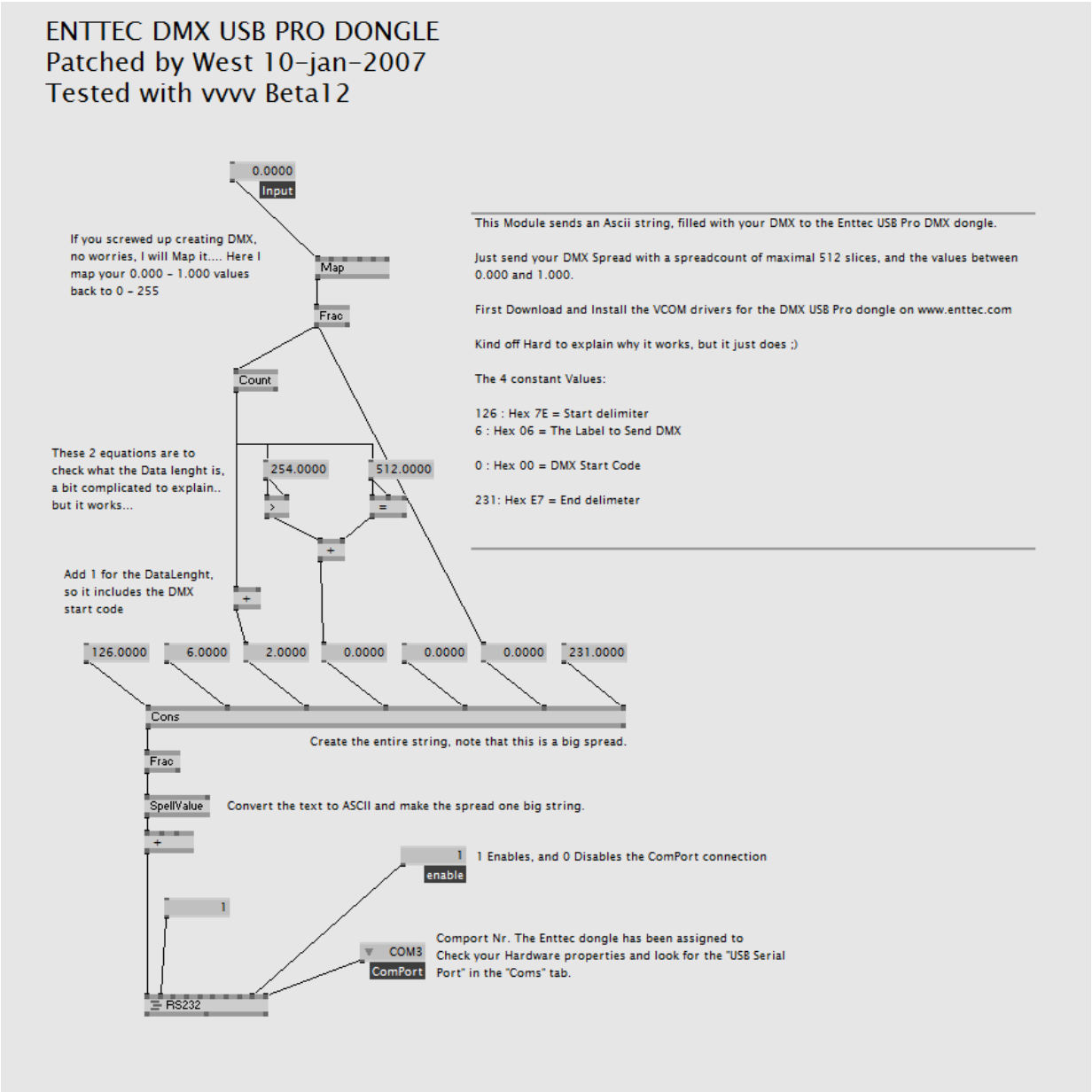
14.15. ábra: Dnx Pwm átalakító

14.16. ábra: Robot lámpa

14.17. ábra: DL3 Projektor



14.18. ábra: Enttec fogadó kezelőfelülete

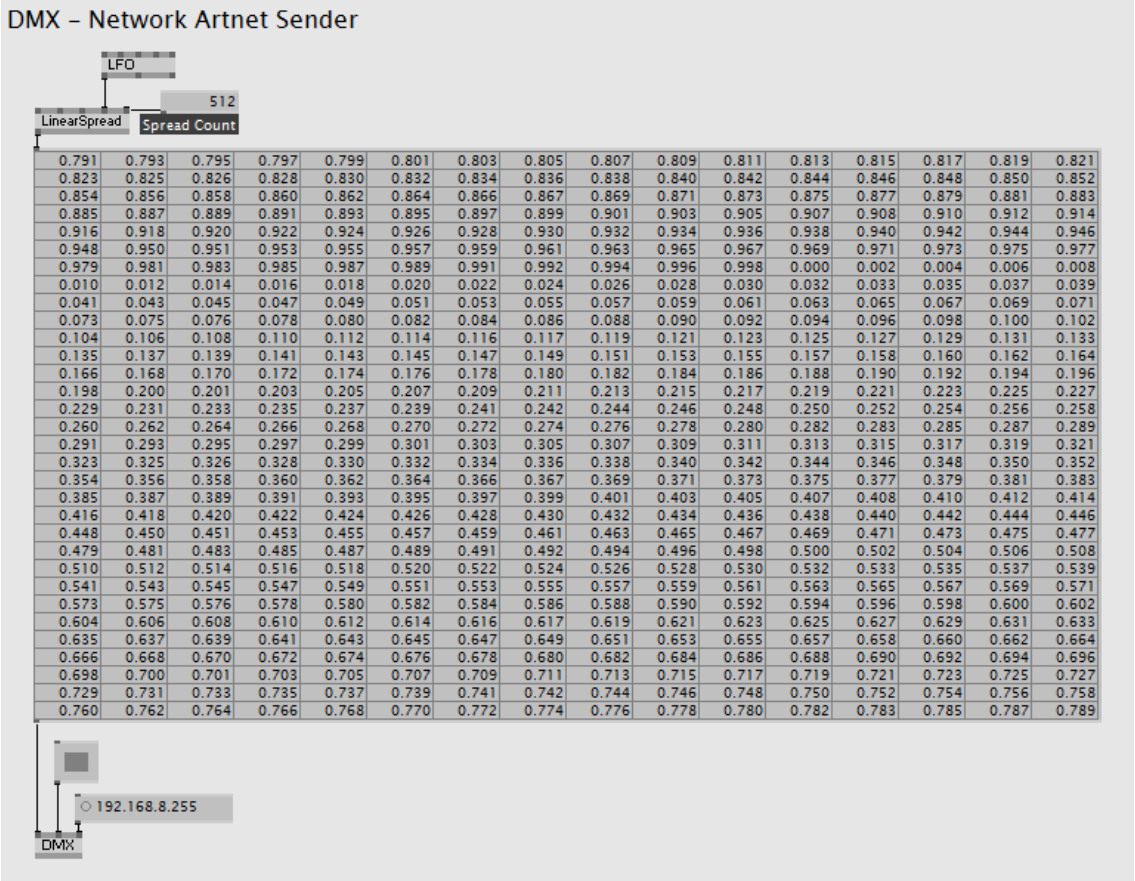


14.19. ábra: Enttec kezelőfelülete

14.20. ábra: Olcsó Artnet Dmx bemenetei

14.21. ábra: Olcsó Artnet Dmx kimenetei

14.22. ábra: Enttec Artnet



0.791	0.793	0.795	0.797	0.799	0.801	0.803	0.805	0.807	0.809	0.811	0.813	0.815	0.817	0.819	0.821
0.823	0.825	0.826	0.828	0.830	0.832	0.834	0.836	0.838	0.840	0.842	0.844	0.846	0.848	0.850	0.852
0.854	0.856	0.858	0.860	0.862	0.864	0.866	0.867	0.869	0.871	0.873	0.875	0.877	0.879	0.881	0.883
0.885	0.887	0.889	0.891	0.893	0.895	0.897	0.899	0.901	0.903	0.905	0.907	0.908	0.910	0.912	0.914
0.916	0.918	0.920	0.922	0.924	0.926	0.928	0.930	0.932	0.934	0.936	0.938	0.940	0.942	0.944	0.946
0.948	0.950	0.951	0.953	0.955	0.957	0.959	0.961	0.963	0.965	0.967	0.969	0.971	0.973	0.975	0.977
0.979	0.981	0.983	0.985	0.987	0.989	0.991	0.992	0.994	0.996	0.998	0.000	0.002	0.004	0.006	0.008
0.010	0.012	0.014	0.016	0.018	0.020	0.022	0.024	0.026	0.028	0.030	0.032	0.033	0.035	0.037	0.039
0.041	0.043	0.045	0.047	0.049	0.051	0.053	0.055	0.057	0.059	0.061	0.063	0.065	0.067	0.069	0.071
0.073	0.075	0.076	0.078	0.080	0.082	0.084	0.086	0.088	0.090	0.092	0.094	0.096	0.098	0.100	0.102
0.104	0.106	0.108	0.110	0.112	0.114	0.116	0.117	0.119	0.121	0.123	0.125	0.127	0.129	0.131	0.133
0.135	0.137	0.139	0.141	0.143	0.145	0.147	0.149	0.151	0.153	0.155	0.157	0.158	0.160	0.162	0.164
0.166	0.168	0.170	0.172	0.174	0.176	0.178	0.180	0.182	0.184	0.186	0.188	0.190	0.192	0.194	0.196
0.198	0.200	0.201	0.203	0.205	0.207	0.209	0.211	0.213	0.215	0.217	0.219	0.221	0.223	0.225	0.227
0.229	0.231	0.233	0.235	0.237	0.239	0.241	0.242	0.244	0.246	0.248	0.250	0.252	0.254	0.256	0.258
0.260	0.262	0.264	0.266	0.268	0.270	0.272	0.274	0.276	0.278	0.280	0.282	0.283	0.285	0.287	0.289
0.291	0.293	0.295	0.297	0.299	0.301	0.303	0.305	0.307	0.309	0.311	0.313	0.315	0.317	0.319	0.321
0.323	0.325	0.326	0.328	0.330	0.332	0.334	0.336	0.338	0.340	0.342	0.344	0.346	0.348	0.350	0.352
0.354	0.356	0.358	0.360	0.362	0.364	0.366	0.367	0.369	0.371	0.373	0.375	0.377	0.379	0.381	0.383
0.385	0.387	0.389	0.391	0.393	0.395	0.397	0.399	0.401	0.403	0.405	0.407	0.408	0.410	0.412	0.414
0.416	0.418	0.420	0.422	0.424	0.426	0.428	0.430	0.432	0.434	0.436	0.438	0.440	0.442	0.444	0.446
0.448	0.450	0.451	0.453	0.455	0.457	0.459	0.461	0.463	0.465	0.467	0.469	0.471	0.473	0.475	0.477
0.479	0.481	0.483	0.485	0.487	0.489	0.491	0.492	0.494	0.496	0.498	0.500	0.502	0.504	0.506	0.508
0.510	0.512	0.514	0.516	0.518	0.520	0.522	0.524	0.526	0.528	0.530	0.532	0.533	0.535	0.537	0.539
0.541	0.543	0.545	0.547	0.549	0.551	0.553	0.555	0.557	0.559	0.561	0.563	0.565	0.567	0.569	0.571
0.573	0.575	0.576	0.578	0.580	0.582	0.584	0.586	0.588	0.590	0.592	0.594	0.596	0.598	0.600	0.602
0.604	0.606	0.608	0.610	0.612	0.614	0.616	0.617	0.619	0.621	0.623	0.625	0.627	0.629	0.631	0.633
0.635	0.637	0.639	0.641	0.643	0.645	0.647	0.649	0.651	0.653	0.655	0.657	0.658	0.660	0.662	0.664
0.666	0.668	0.670	0.672	0.674	0.676	0.678	0.680	0.682	0.684	0.686	0.688	0.690	0.692	0.694	0.696
0.698	0.700	0.701	0.703	0.705	0.707	0.709	0.711	0.713	0.715	0.717	0.719	0.721	0.723	0.725	0.727
0.729	0.731	0.733	0.735	0.737	0.739	0.741	0.742	0.744	0.746	0.748	0.750	0.752	0.754	0.756	0.758
0.760	0.762	0.764	0.766	0.768	0.770	0.772	0.774	0.776	0.778	0.780	0.782	0.783	0.785	0.787	0.789

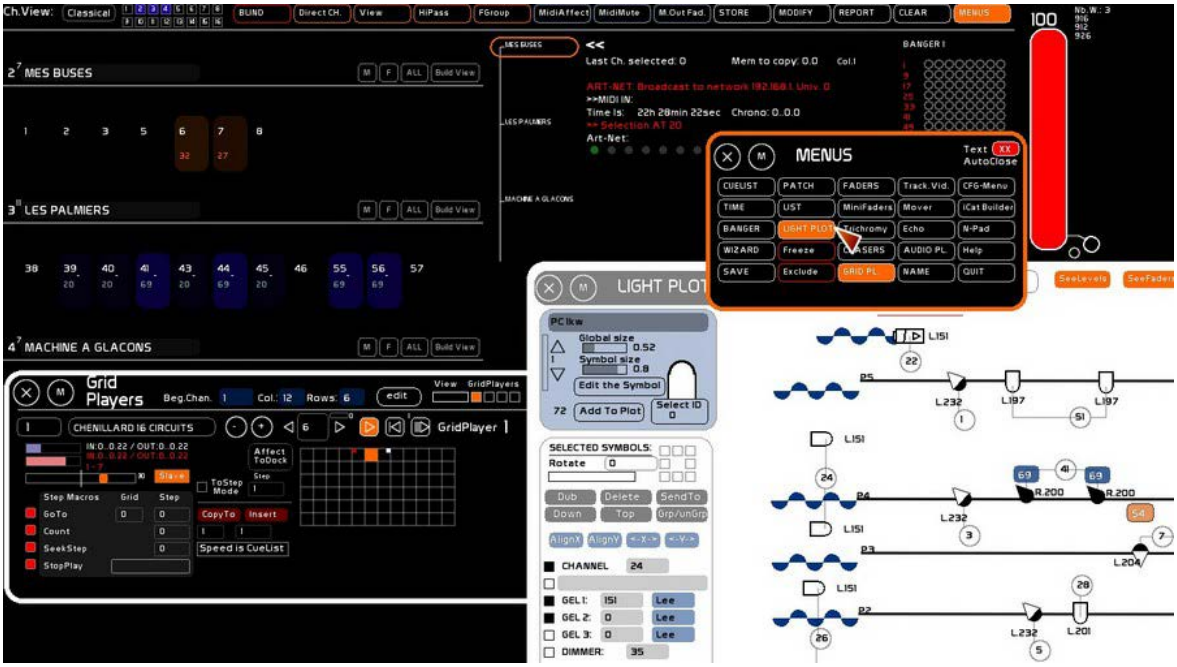
14.23. ábra: Artnet Dmx küldő

DMX – Network Artnet Reciever

DMX 0.0000

0.992	0.996	0.996	1.000	0.000	0.004	0.004	0.008	0.008	0.012	0.012	0.016	0.016	0.020	0.020	0.024
0.024	0.027	0.027	0.031	0.031	0.035	0.035	0.039	0.039	0.043	0.043	0.047	0.047	0.051	0.051	0.055
0.055	0.059	0.059	0.063	0.063	0.067	0.067	0.071	0.071	0.075	0.075	0.078	0.078	0.082	0.082	0.086
0.086	0.090	0.090	0.094	0.094	0.098	0.098	0.102	0.102	0.106	0.106	0.110	0.110	0.114	0.114	0.118
0.118	0.122	0.122	0.125	0.125	0.129	0.129	0.133	0.133	0.137	0.137	0.141	0.141	0.145	0.145	0.149
0.149	0.153	0.153	0.157	0.157	0.161	0.161	0.165	0.165	0.169	0.169	0.173	0.173	0.176	0.176	0.180
0.180	0.184	0.184	0.188	0.188	0.192	0.192	0.196	0.196	0.200	0.200	0.204	0.204	0.208	0.208	0.212
0.212	0.216	0.216	0.220	0.220	0.224	0.224	0.227	0.227	0.231	0.231	0.235	0.235	0.239	0.239	0.243
0.243	0.247	0.247	0.251	0.251	0.255	0.255	0.259	0.259	0.263	0.263	0.267	0.267	0.271	0.271	0.275
0.275	0.278	0.278	0.282	0.282	0.286	0.286	0.290	0.290	0.294	0.294	0.298	0.298	0.302	0.302	0.306
0.306	0.310	0.310	0.314	0.314	0.314	0.318	0.318	0.322	0.322	0.325	0.325	0.329	0.329	0.333	0.333
0.337	0.337	0.341	0.341	0.345	0.345	0.349	0.349	0.353	0.353	0.357	0.357	0.361	0.361	0.365	0.365
0.369	0.369	0.373	0.373	0.376	0.376	0.380	0.380	0.384	0.384	0.388	0.388	0.392	0.392	0.396	0.396
0.400	0.400	0.404	0.404	0.408	0.408	0.412	0.412	0.416	0.416	0.420	0.420	0.424	0.424	0.427	0.427
0.431	0.431	0.435	0.435	0.439	0.439	0.443	0.443	0.447	0.447	0.451	0.451	0.455	0.455	0.459	0.459
0.463	0.463	0.467	0.467	0.471	0.471	0.475	0.475	0.478	0.478	0.482	0.482	0.486	0.486	0.490	0.490
0.494	0.494	0.498	0.498	0.502	0.502	0.506	0.506	0.510	0.510	0.514	0.514	0.518	0.518	0.522	0.522
0.525	0.525	0.529	0.529	0.533	0.533	0.537	0.537	0.541	0.541	0.545	0.545	0.549	0.549	0.553	0.553
0.557	0.557	0.561	0.561	0.565	0.565	0.569	0.569	0.573	0.573	0.576	0.576	0.580	0.580	0.584	0.584
0.588	0.588	0.592	0.592	0.596	0.596	0.600	0.600	0.604	0.604	0.608	0.608	0.612	0.612	0.616	0.616
0.620	0.620	0.624	0.624	0.627	0.627	0.631	0.631	0.635	0.635	0.639	0.639	0.643	0.643	0.647	0.647
0.651	0.651	0.655	0.655	0.659	0.659	0.663	0.663	0.667	0.667	0.671	0.671	0.675	0.675	0.678	0.678
0.682	0.682	0.686	0.686	0.690	0.690	0.694	0.694	0.698	0.698	0.702	0.702	0.706	0.706	0.710	0.710
0.714	0.714	0.718	0.718	0.722	0.722	0.725	0.725	0.729	0.729	0.733	0.733	0.737	0.737	0.741	0.741
0.745	0.745	0.749	0.749	0.753	0.753	0.757	0.757	0.761	0.761	0.765	0.765	0.769	0.769	0.773	0.773
0.776	0.776	0.780	0.780	0.784	0.784	0.788	0.788	0.792	0.792	0.796	0.796	0.800	0.800	0.804	0.804
0.808	0.808	0.812	0.812	0.812	0.816	0.816	0.820	0.820	0.824	0.824	0.827	0.827	0.831	0.831	0.835
0.835	0.839	0.839	0.843	0.843	0.847	0.847	0.851	0.851	0.855	0.855	0.859	0.859	0.863	0.863	0.867
0.867	0.871	0.871	0.875	0.875	0.878	0.878	0.882	0.882	0.886	0.886	0.890	0.890	0.894	0.894	0.898
0.898	0.902	0.902	0.906	0.906	0.910	0.910	0.914	0.914	0.918	0.918	0.922	0.922	0.925	0.925	0.929
0.929	0.933	0.933	0.937	0.937	0.941	0.941	0.945	0.945	0.949	0.949	0.953	0.953	0.957	0.957	0.961
0.961	0.965	0.965	0.969	0.969	0.973	0.973	0.976	0.976	0.980	0.980	0.984	0.984	0.988	0.988	0.992

14.24. ábra: Artnet Dmx Fogadó



14.25. ábra: Little Cat Midi kezelőfelülete

15. Kezelőfelületek készítése

A kezelőfelületek készítésére a legjobb példa az IRIS nevű projekt, amit Milo publikált a VVVV *Contribution* oldalán (http://iris.nsynk.de/). Ez egy teljesen új multitouch kezelőfelület a VVVV-hez, amit DirectX renderben lehet használni, és nem szükséges hozzá node-szerkesztő – leginkább a Touch Designer nevű programra hasonlít. Ebbe a rendszerbe a kezelőfelületen kívül még a *Boygrouping*-ot is beleépítették, amivel kiválaszthatod, hogy csak egy gépről akarsz-e vetíteni, vagy sok gépet akarsz egyszerre szinkronban vezérelni (ez utóbbi a sokprojektoros rendszereknél elengedhetetlen).

A VVVV-ben alapesetben két kezelőfelület-készítési lehetőség van: az egyiket *Gui,* a másikat *Gui Multitouch* néven találod meg a node-menüben, de ezeken kívül akár a sajátodat is elkészítheted a *Quad, Tagpoint, Pickpoint* és más modulok segítségével.

A VVVV első multitouch kezelőfelületét, a *Gui*-t először a Vux publikálta. A két kezelő nagyon hasonlít egymásra, annyi különbséggel, hogy az újabb verzióban néhány funkció már nem elérhető. Mindkettő lényege, hogy a kimenetén egy olyan *Transform*-ot kapsz, ami pozicionálja a *Quad*-ot, a bemenetein pedig a *Transform*-mal tudod beállítani a pozícióját, vagyis hogy hány sorból és oszlopból álljon, egy gombnak mekkora legyen a mérete az *X* és *Y* tengelyeken, illetve hogy mi legyen a gombok értéke. Ez utóbbit akkor kell használni, ha mentett beállítást szeretnél visszatölteni.

Itt a következő gombtípusok közül választhatsz: *ToggleButton, Slider, SliderXY, RotaryButton,* illetve a *Gui* verzióban még választható a *RadioButton* is, ami egy *Bang*-nek felel meg. Ez utóbbit viszont *Togedge* és *Framedelay* segítségével magad is el tudod készíteni *ToggleButton*nal, csak mindig nullázni kell az egér lenyomása után a modult.

A legjelentősebb különbség a két kezelő között, hogy a régi *Gui*-nál színeket is meg lehetett adni (amit egyébként sokkal egyszerűbb külön elkészíteni), illetve, hogy az új *Gui*-ból hiányzik a *MouseOver* funkció.

Ha esetleg 3D-s kezelőfelületet szeretnél készíteni, arra is van lehetőség.

Ebből két variáció létezik a VVVV-ben: az egyik a *Button (3D Quad),* amit térben elhelyezett *Quad*-okkal tudsz használni. Ennek az első bemenetébe kell belekötni a *Quad*-ok *Transform*-pozícióját, a másodikba az egérpozíciót, végül pedig a *Camera*

View Projection-t. Ennek köszönhetően a modul első kimenetén megkapod, hogy melyik *Quad*-ok fölött helyezkedik el az egér, a másodikon (ha több *Quad*-od van egymás mögött), hogy melyik a legelső, végül pedig az egérpozíciót.

A másik térbeli megoldás a *Button (3D Mesh)* modul, amibe 3D-s formákat tudsz betölteni, aminek az eredménye akár lehet egy több *Mesh*-ből álló collada-fájl is, aminek az elemei közül az egérpozíció alapján tudsz majd választani. Az előző *Button*-hoz képest itt annyi különbség van, hogy a *Mesh*-t is bele kell kötni a modulba. Fontos, hogy ez a modul csak számadatokat küld ki, amikkel részeket lehet kijelölni, vagyis mindenképpen kell mellé egy megjelenítő is (*PhongDirectional* vagy egy egyszerű *Constant*).

Az elkészült kezelőfelületet használhatod egérrel, Wacom tablettel, multitouch monitorral, vagy akár saját építésű multitouch interface-szel is. Ha esetleg multitouch monitor vásárlásán gondolkodsz, érdemes körbenézni a használtpiacon, hogy találsz-e Elo márkájú terméket, mivel ez a gyártó készíti a legjobban működő, ütésbisztos monitorokat. Az utóbbi időben nagyon sok cég kezdett el touchmonitorokat gyártani, de még a nagyon elismert HP-nek sem sikerült eddig igazán használható verziót készítenie.

Ha kijelöléssel vagy csak *MouseOver*-el szeretnél dolgozni, akkor érdemes a *TagPoints,* illetve a *PickPoints* nevű modulokat használnod. A *TagPoints*-ot használva az egér bal gombjával tudsz kijelölni (több kijelölésnél a pontok mindig hozzáadódnak az előzőhöz), a jobb gombbal pedig invertálni tudod a kijelölést. A *PickPoints* viszont csak akkor működik, ha egy objektum fölött van az egered, és akkor klikkelsz – hasonló módon, mint a *Button (3D Mesh)*-nél.

Ezek után nézzük meg a fájlba írás részleteit, mivel ez elengedhetetlen, ha valamikor vissza szeretnéd állítani a gombjaid legutolsó állapotát!

Erre kétféle lehetőséged van: az egyik módszernél az adatokat RAW-ba mented el, ami a lehető leggyorsabb eljárás, viszont a kész fájl nem szerkeszthető szövegszerkesztőben, mivel nyers adatként van tárolva.

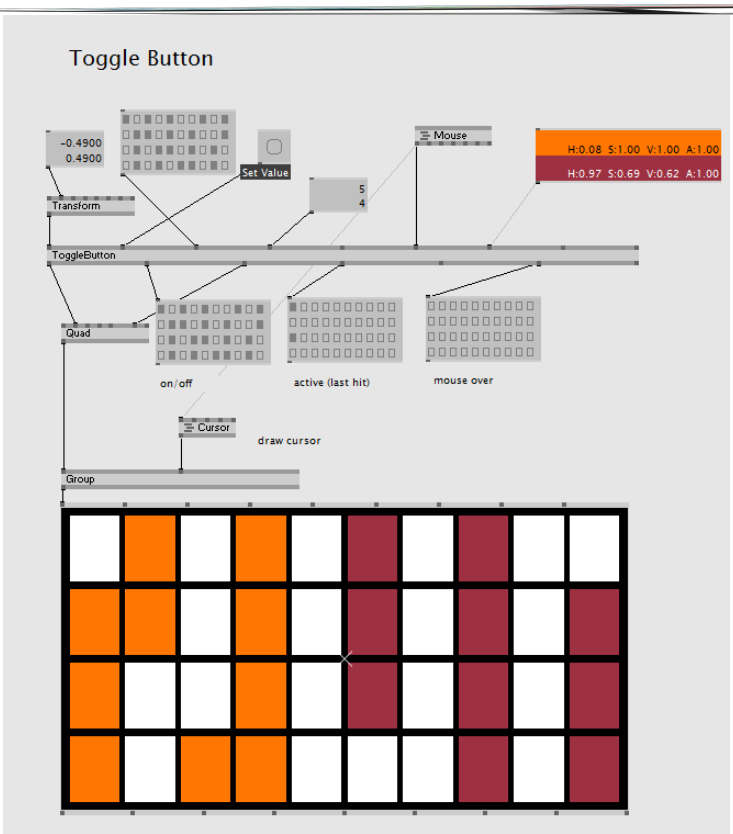
A másik módszer, ha szövegként mented el az anyagodat, aminek az a nagy előnye, hogy azt a VVVV-től függetlenül is bármikor tudod szerkeszteni. Hátránya viszont, hogy az olvasás-írás ennél a módszernél eléggé lassú, így sok fájlba írás esetén nem ez a legjobb megoldás. Mindenesetre bármelyik módszert használod, ha számot szeretnél menteni, azt előbb mindig át kell alakítani RAW-ba *AsRaw*-val, vagy *AsString*-gel.

A file-ba íráshoz a *Write* modulokat tudod használni: itt mindegyik esetben az első bemenetre kell bekötni a menteni kívánt adatot, a másodikra a fájl nevét és elérését, a harmadikon az átalakítás típusát tudod megadni, a negyedik pedig mindegyik esetben maga az írás.

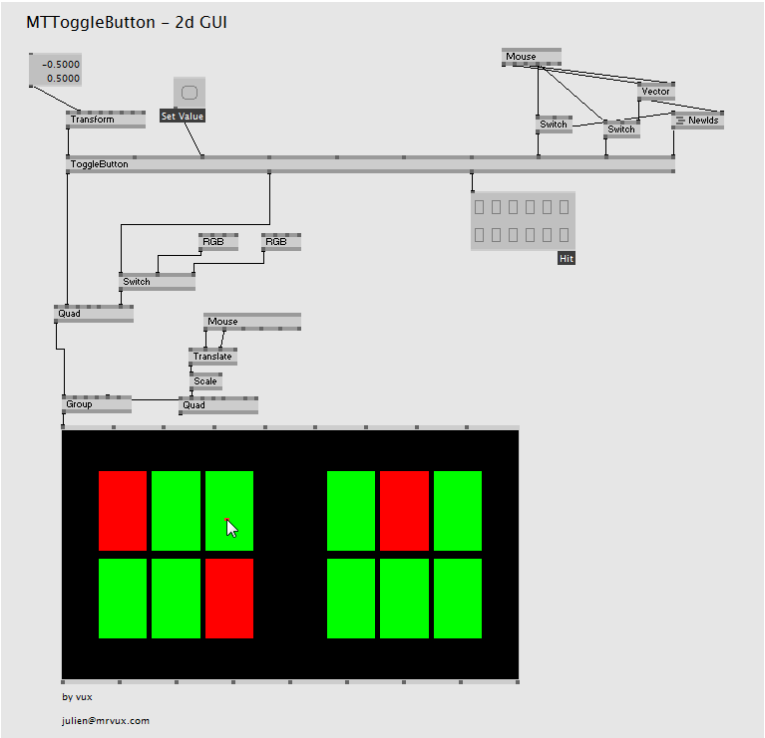
A fájl betöltése lehet automatikus vagy manuális. A manuális módszernél egyszerűen csak egy gombot kell bekötnöd a megfelelő *Read* modulba, és kiválasztani a fájl elérési útvonalát – a kimenetét aztán *AsValue* modul segítségével tudod átalakítani számmá, amit a gombjaid *ValueInput* bemenetére kell rákötni. A gombok *SetValue* PIN-jére pedig *Framedelay*-jel kell rákötni a betöltésre használt gombot (a *Framedelay* itt azért szükséges, mert a beolvasás időbe telik, és a változtatást ezért késleltetni kell).

Az automatikus mentés/betöltéshez a gombok *Hit* kimenetére kell kötni egy *OR (spectral)* modult, aminek a kimenetét egy *TogEdge*-be kell kötni. Az *UpEdge* kimenetét a *Write*-ra, míg a *DownEdge* kimenetét pedig két egymás után kötött *FrameDelay*-re kell kötni. Az első *FrameDelay*-t pedig a *Reader*-be, és az előbb említett beolvasási késés miatt a másodikat a gombok *SetValue* bemenetére. Így már minden klikkelés után automatikusan mentődik minden, illetve a gombok értékei is betöltődnek.

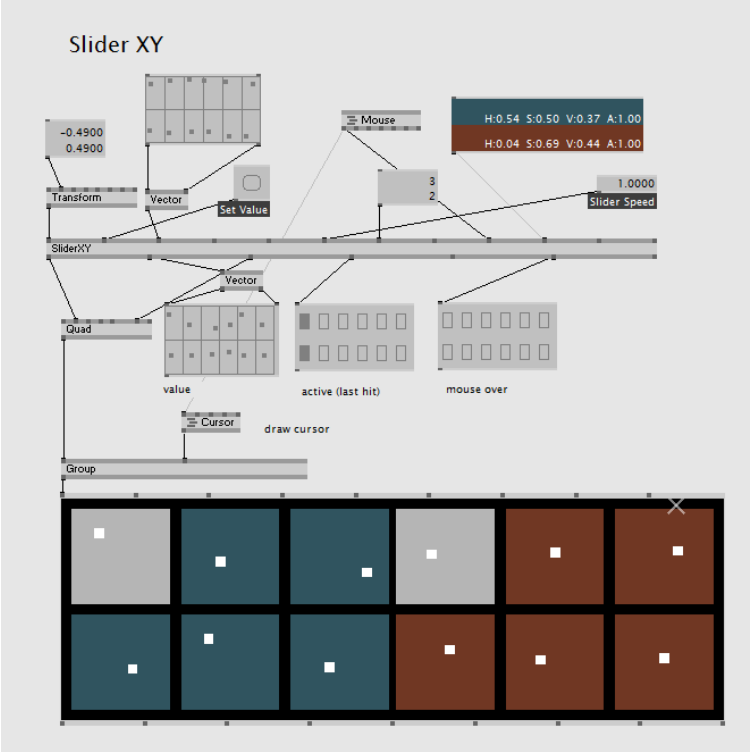
Ha a program indításakor is szeretnél automatikus betöltést, ehhez nem kell más, mint egy *OR* modul, amit a *Reader* modul *Read* bemenetére kötsz úgy, hogy az *OR* első bemenetére kötöd az eddig használt *FrameDelay*-t, a másikra pedig egy *OnOpen* modult. Előfordulhat, hogy annyira sok ideig töltődik be a programod, hogy a méreténél fogva túl hamar hajtódik végre az *OnOpen* – ilyen esetekben a *Delay* modul segítségével érdemes ezt késleltetni néhány másodperc erejéig.



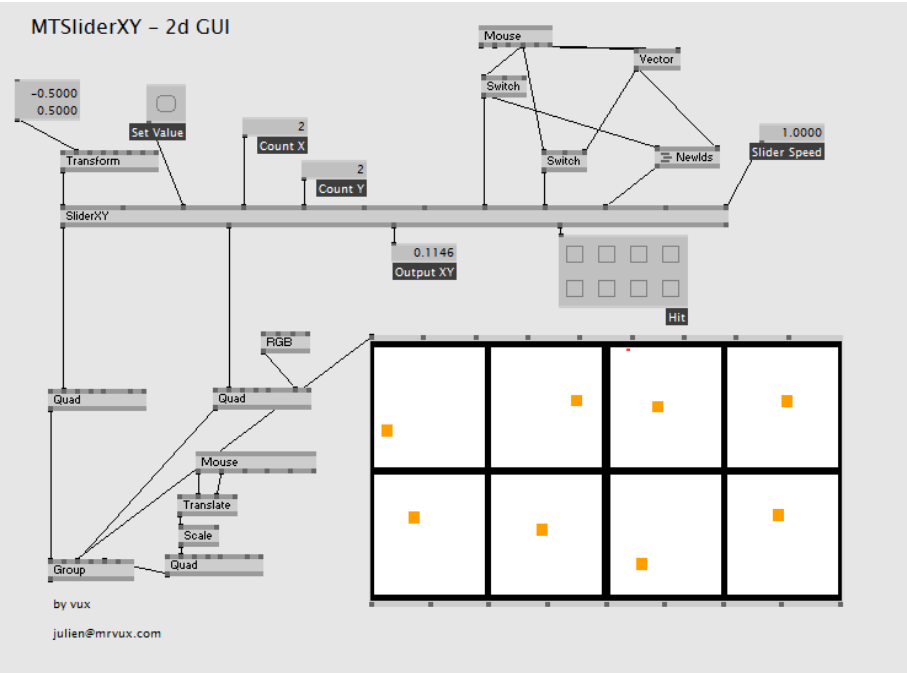
15.1. ábra: Toggle Button Gui



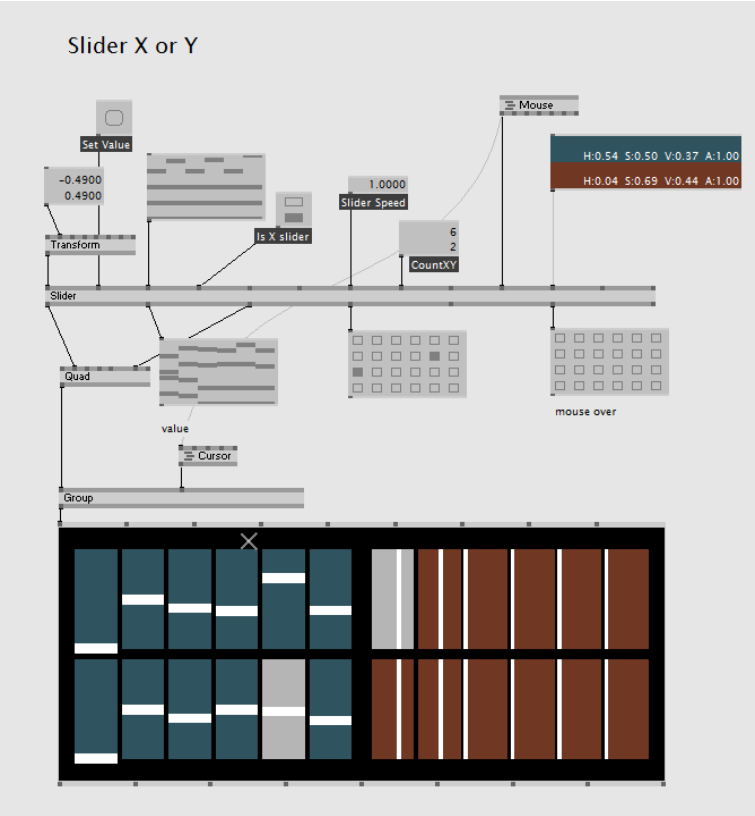
15.2. ábra: Toggle Button Gui Multitouch



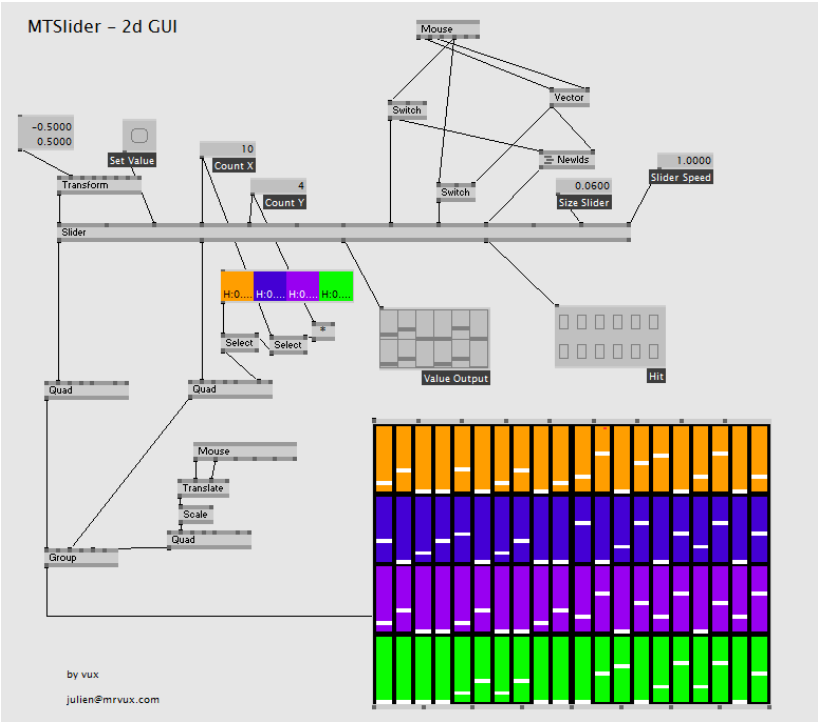
15.3. ábra: XY Slider Gui



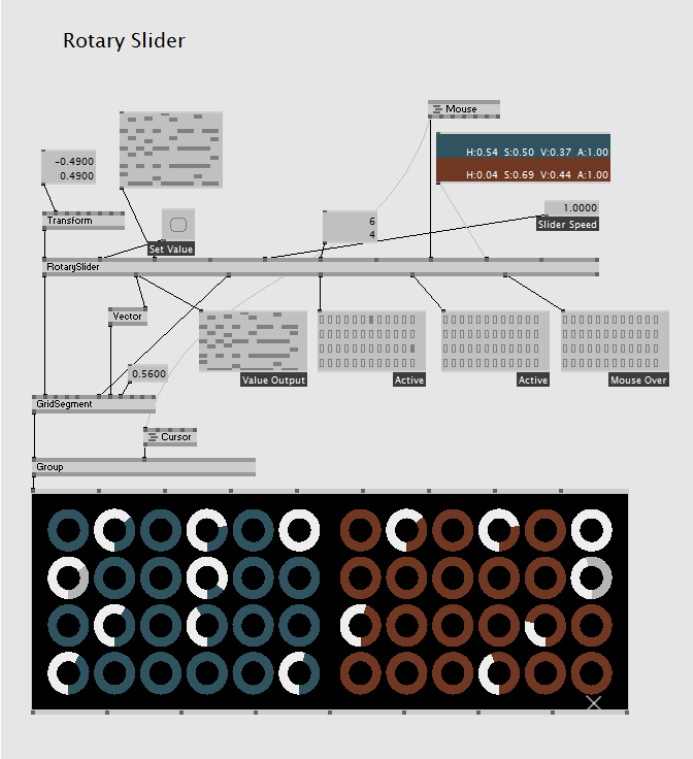
15.4. ábra: XY Slider Gui Multitouch



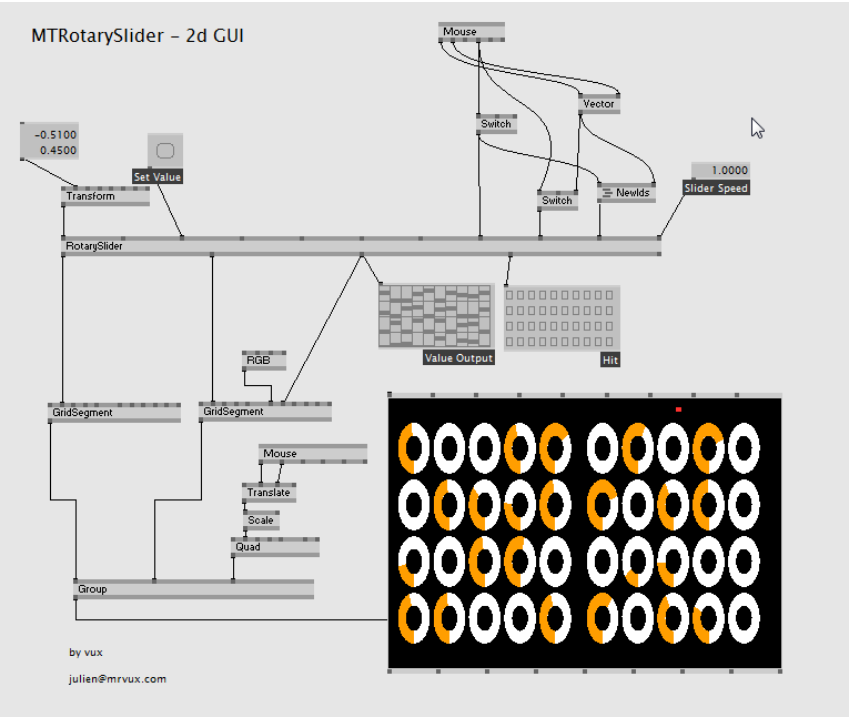
15.5. ábra: Slider Gui



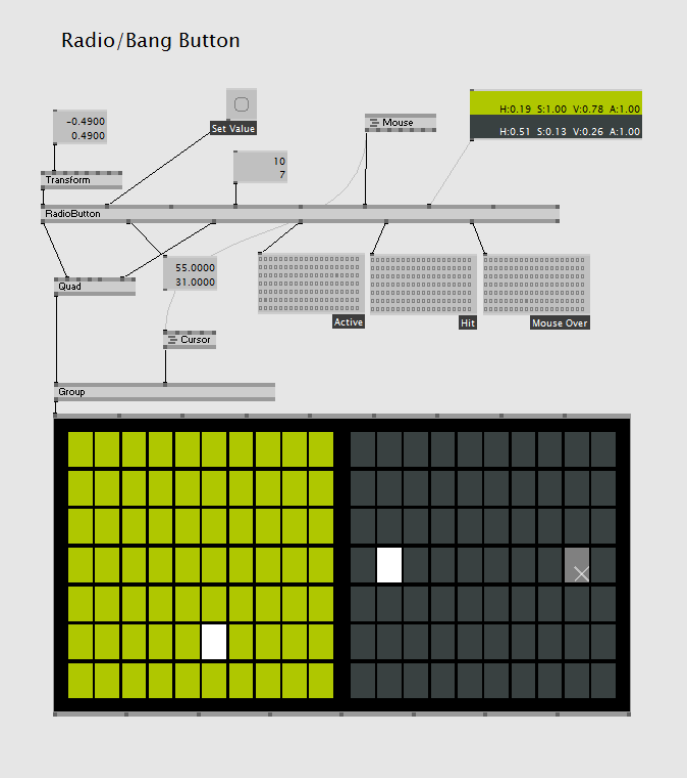
15.6. ábra: Slider Gui Multitouch



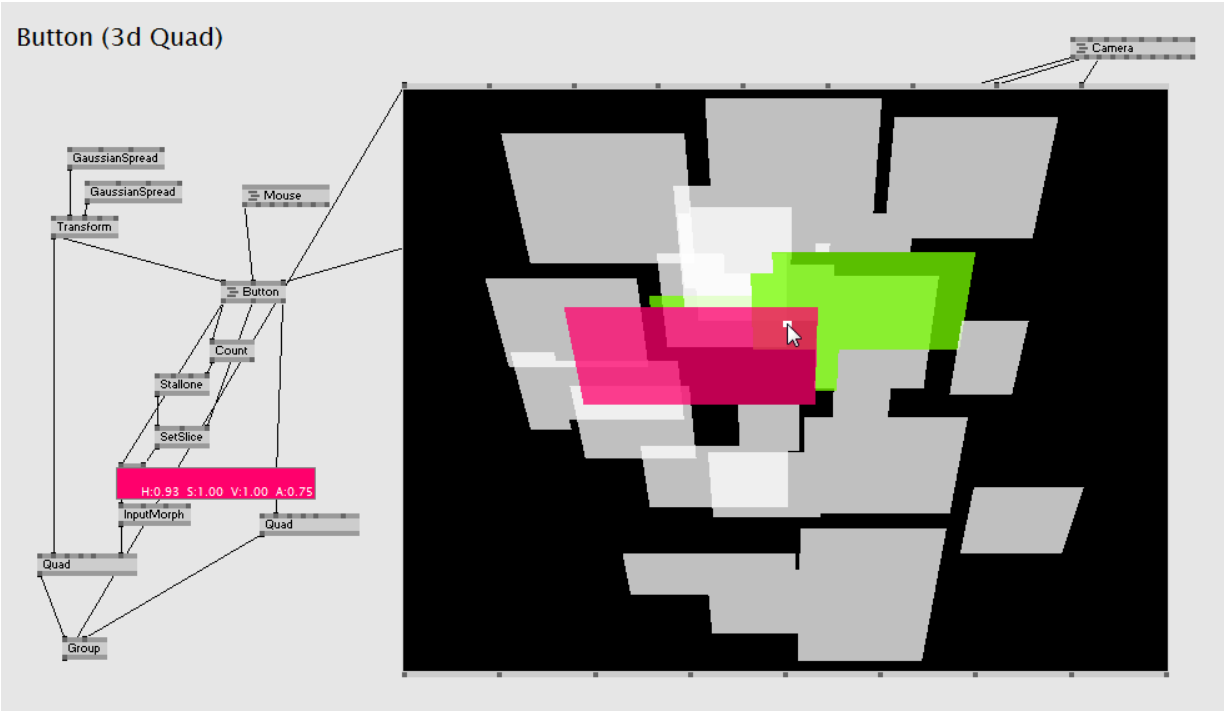
15.7. ábra: Rotary Slider Gui



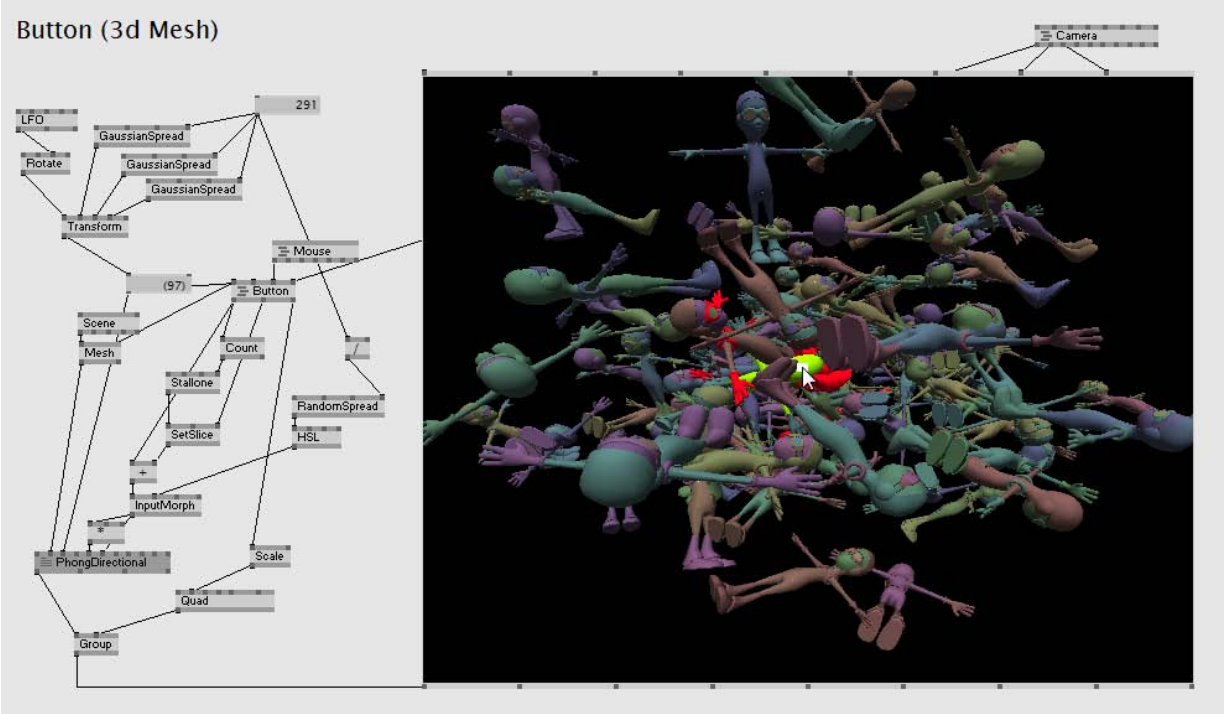
15.8. ábra: Rottary Slider Gui Multitouch



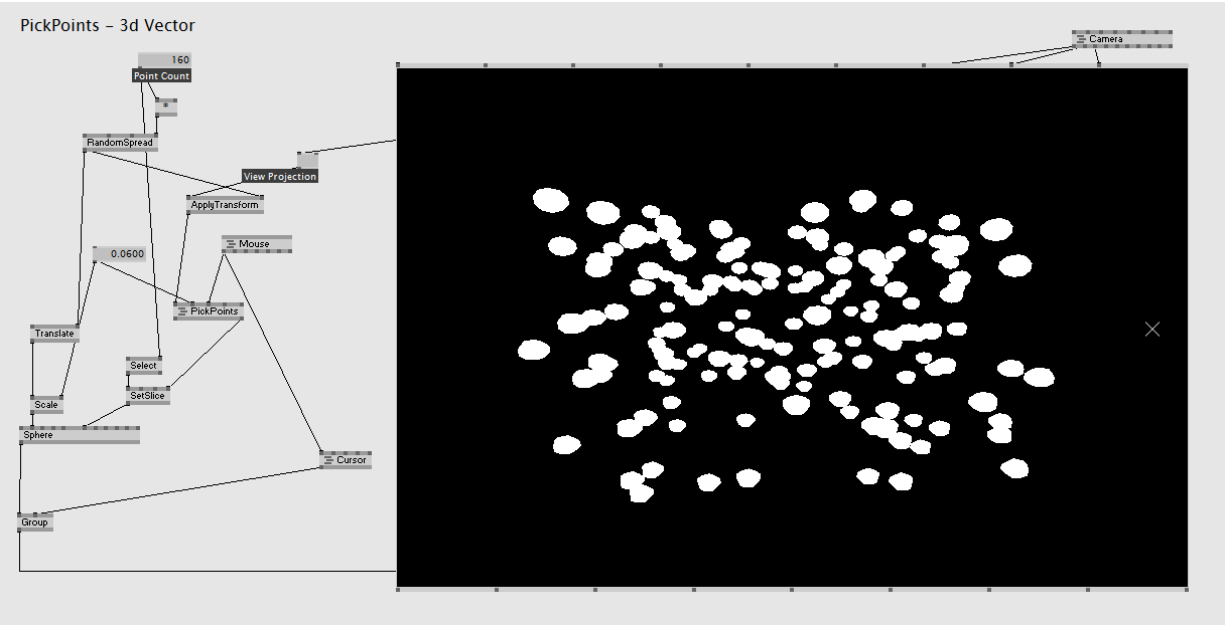
15.9. ábra: Radio Button



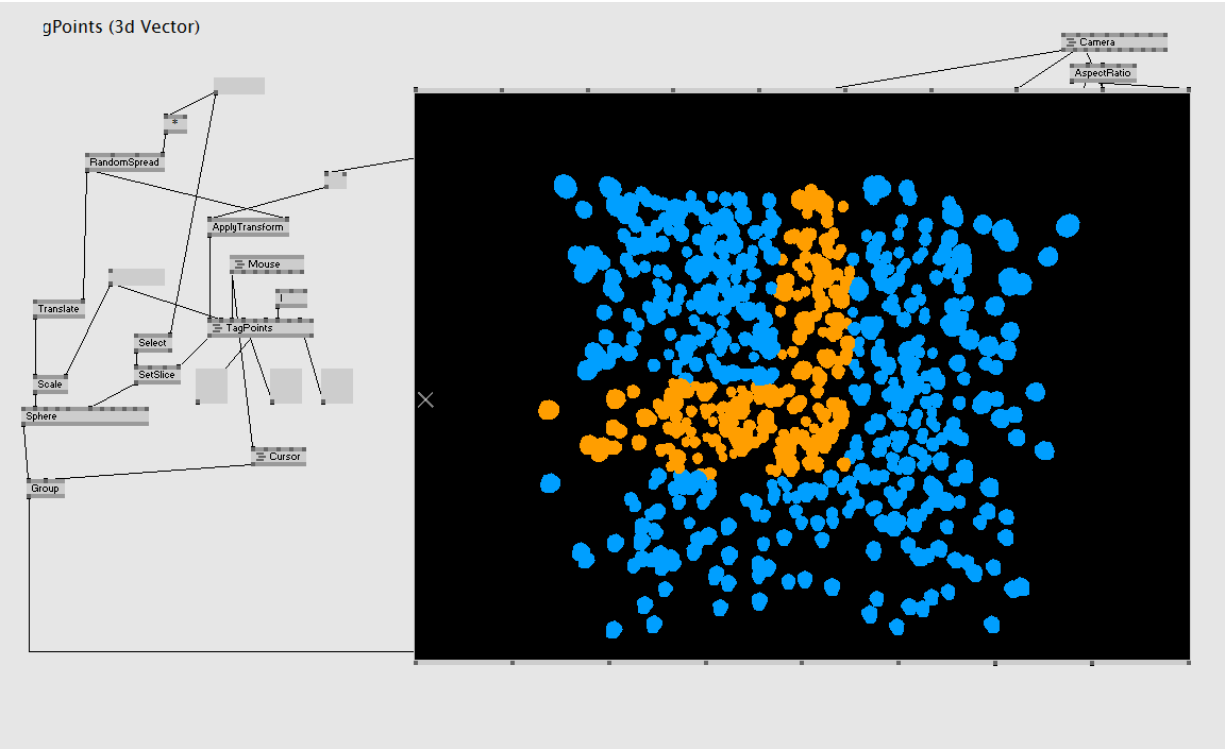
15.10. ábra: Button 3D Quad



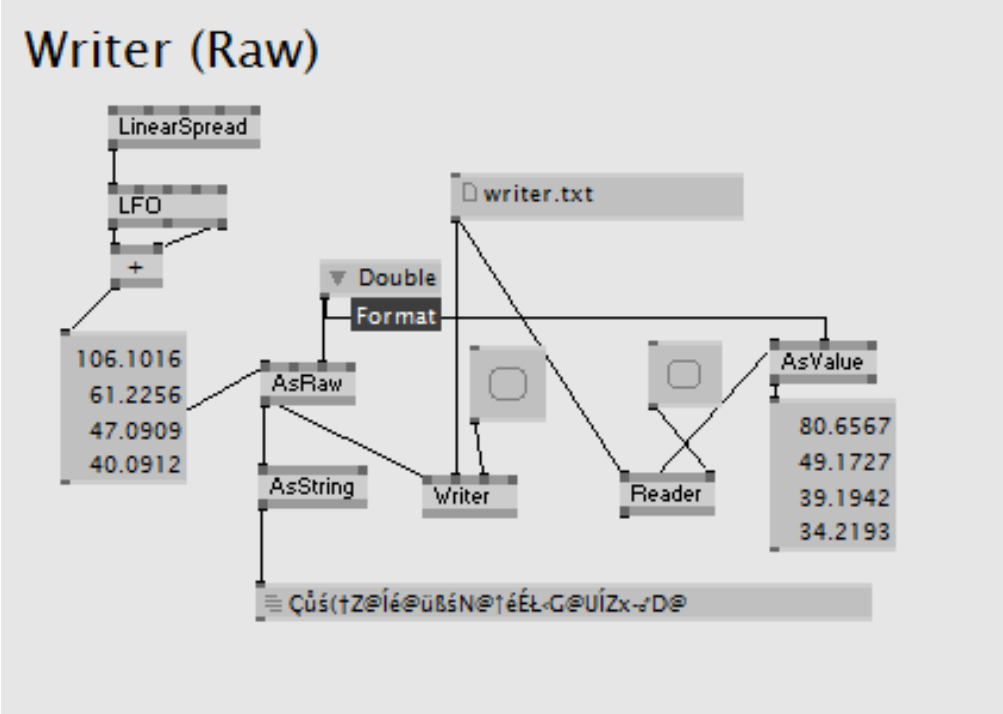
15.11. ábra: Button 3D Mesh



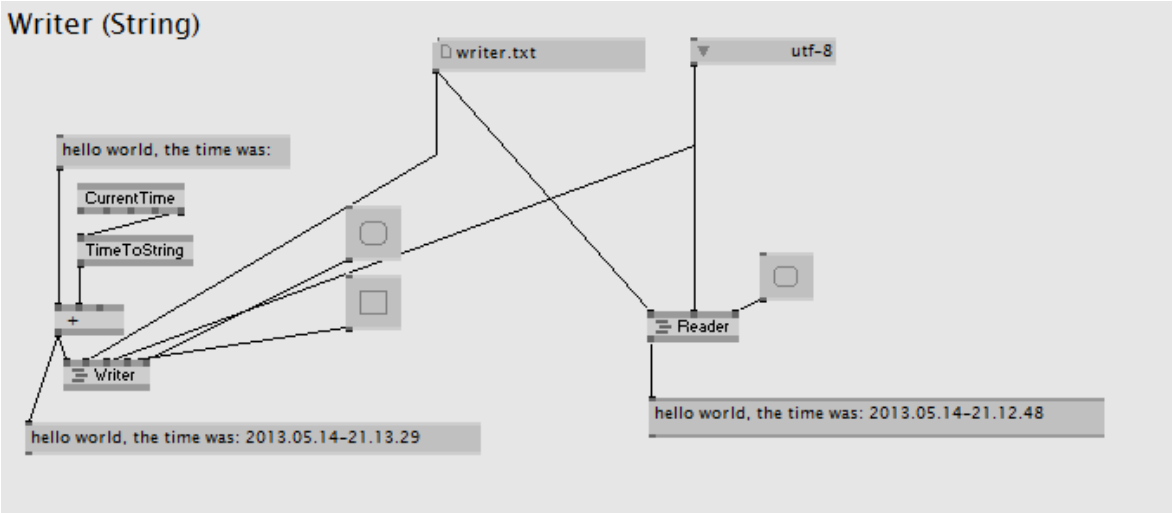
15.12. ábra: PickPoints



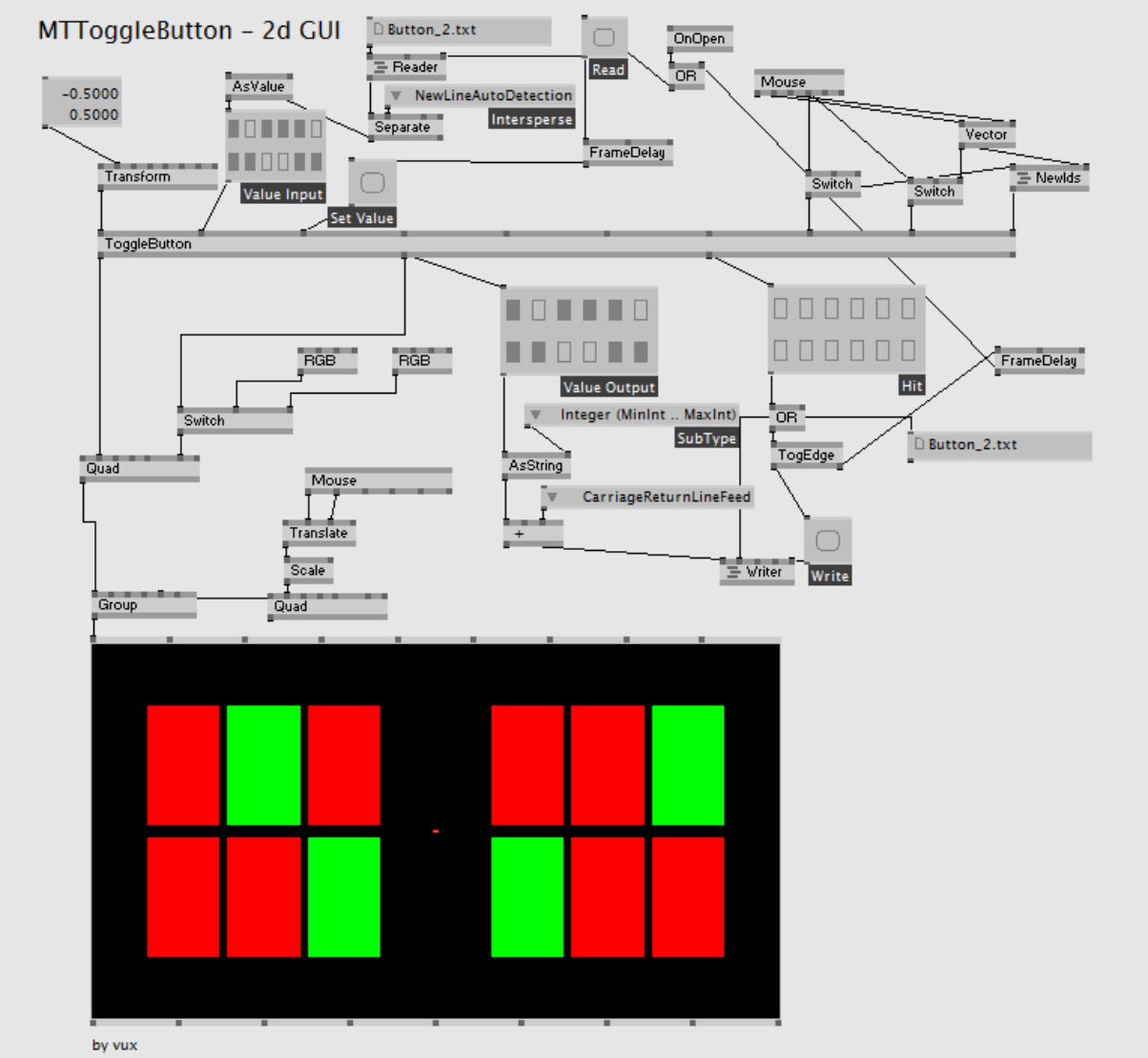
15.13. ábra: TagPoints



15.14. ábra: Raw mentés



15.15. ábra: String mentés

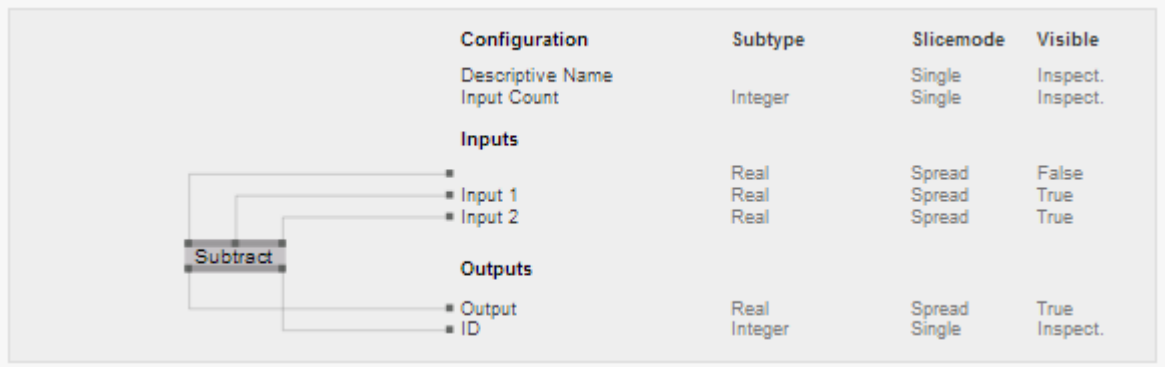


15.16. ábra: Automatikus mentés

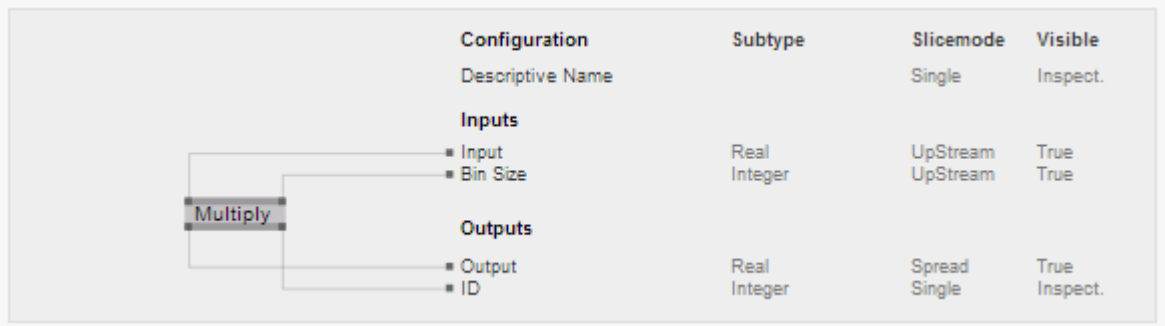
Node list

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

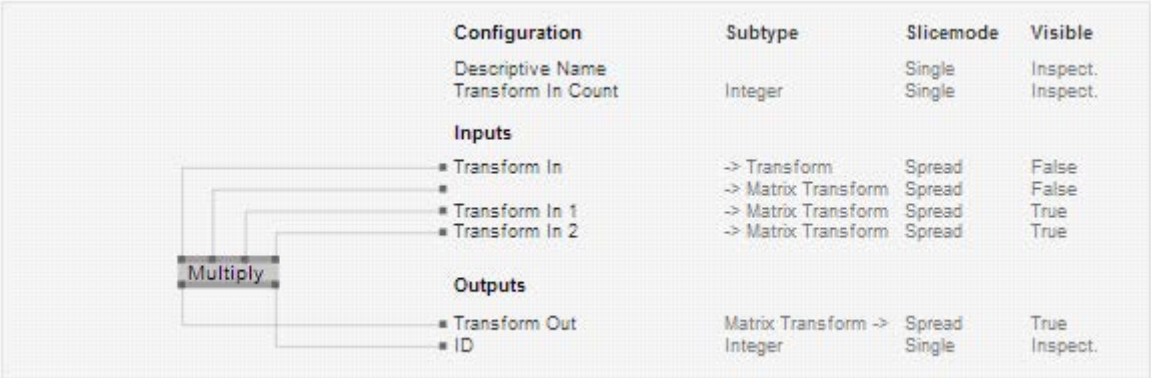
- (Value)



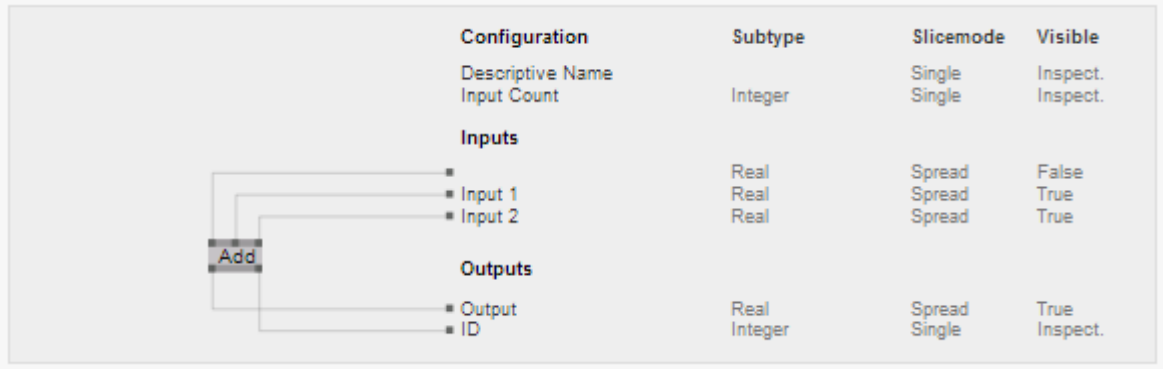
*** (Value Spectral)**



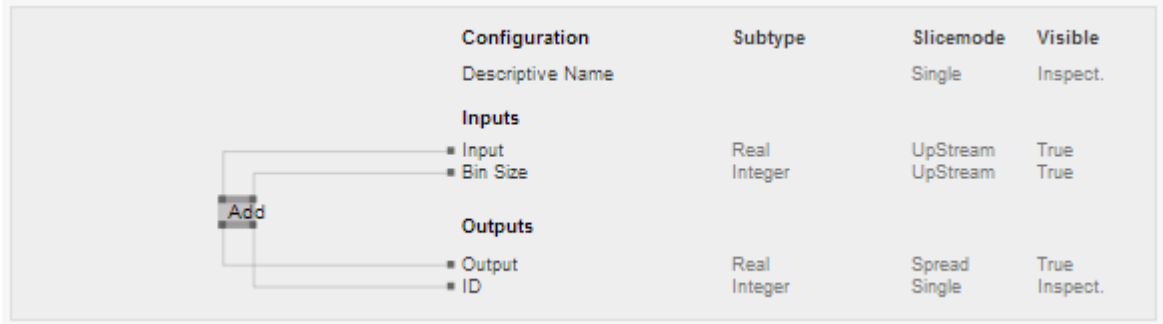
*** (Transform)**



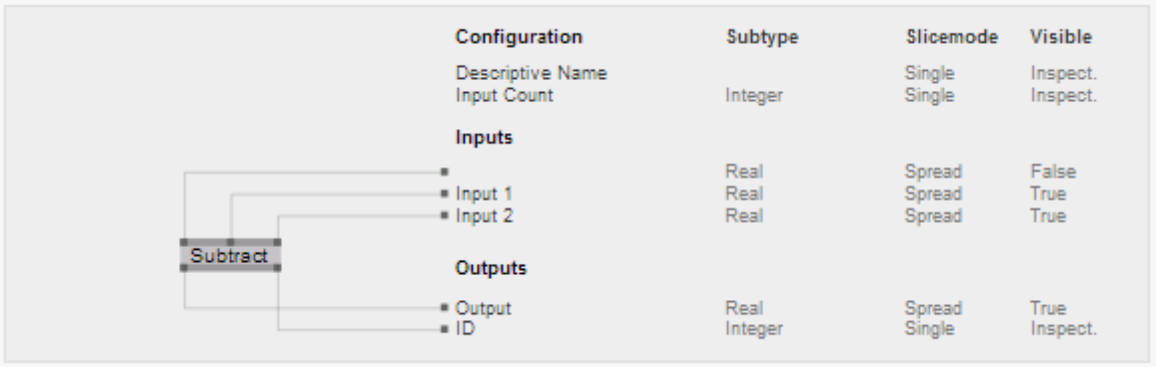
+ (Value)



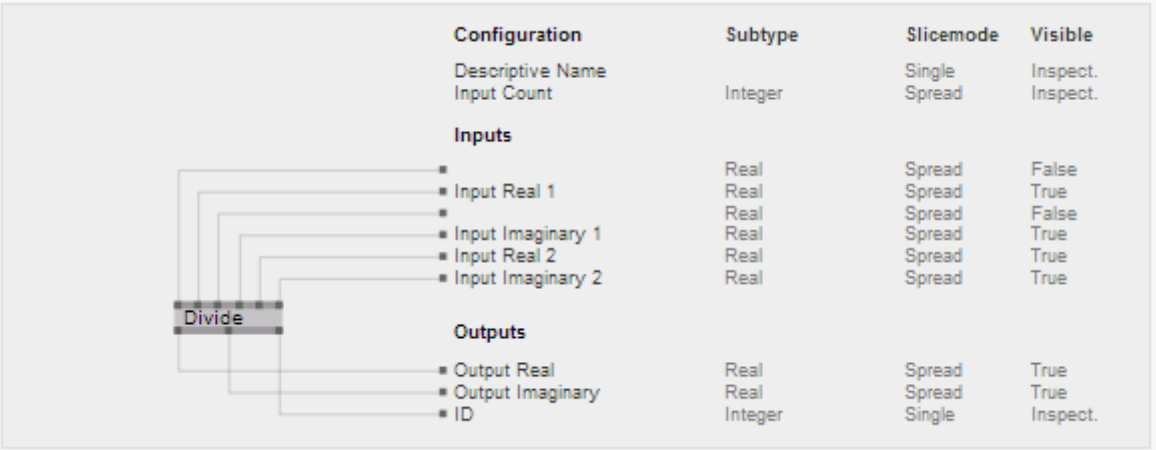
+ (Value Spectral)



- (Value)

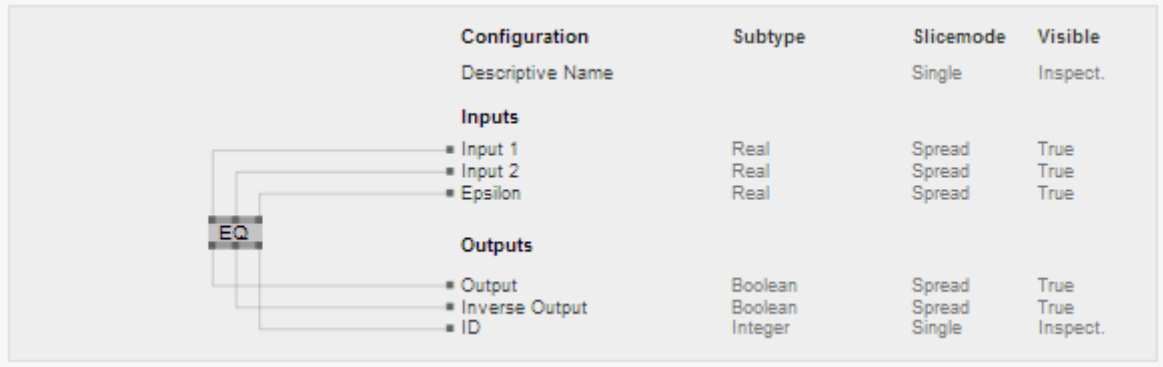


/ (Complex)



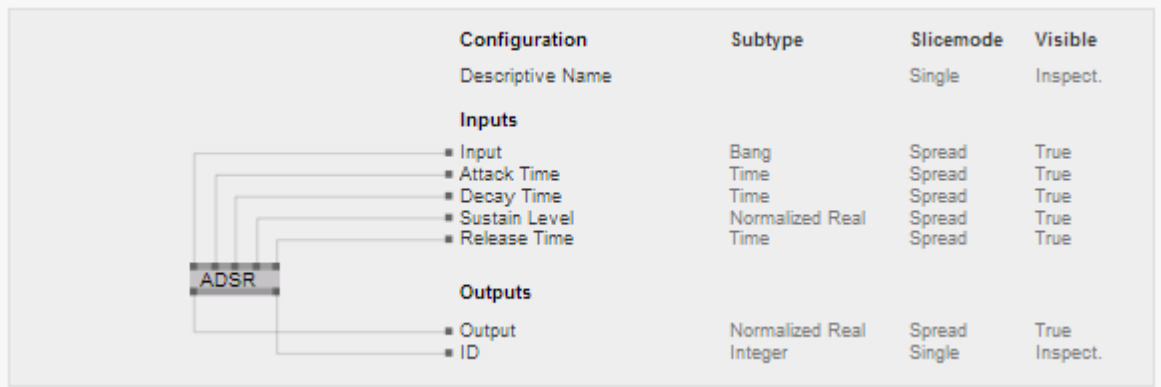
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Input Count	Integer	Spread	Inspect.
Inputs			
	Real	Spread	False
Input Real 1	Real	Spread	True
	Real	Spread	False
Input Imaginary 1	Real	Spread	True
Input Real 2	Real	Spread	True
Input Imaginary 2	Real	Spread	True
Outputs			
Output Real	Real	Spread	True
Output Imaginary	Real	Spread	True
ID	Integer	Single	Inspect.

= (Value)



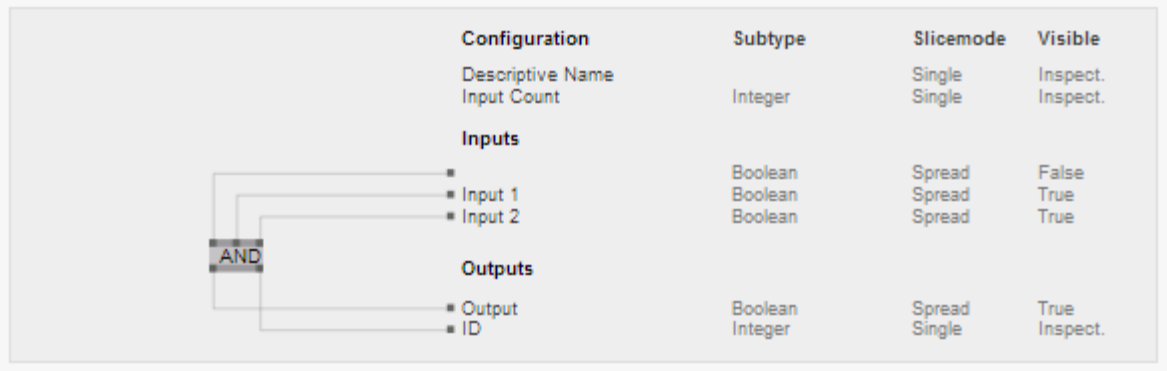
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Input 1	Real	Spread	True
Input 2	Real	Spread	True
Epsilon	Real	Spread	True
Outputs			
Output	Boolean	Spread	True
Inverse Output	Boolean	Spread	True
ID	Integer	Single	Inspect.

ADSR (Animation)

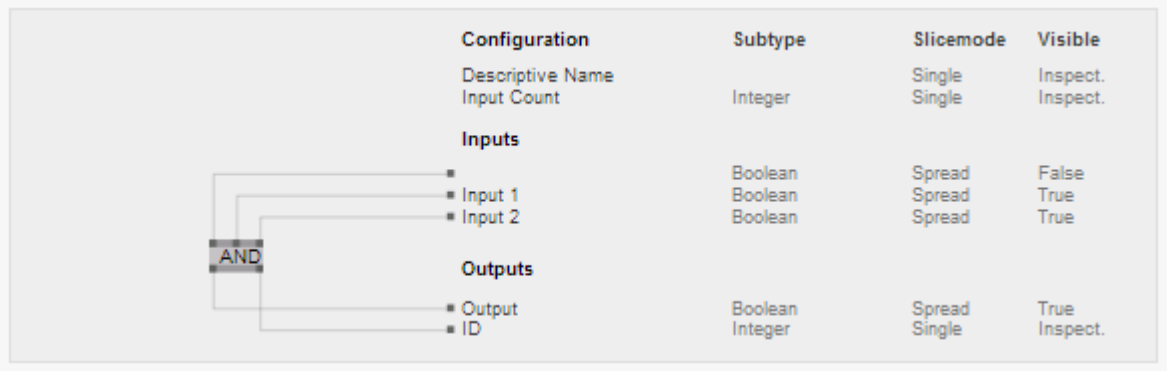


Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Input	Bang	Spread	True
Attack Time	Time	Spread	True
Decay Time	Time	Spread	True
Sustain Level	Normalized Real	Spread	True
Release Time	Time	Spread	True
Outputs			
Output	Normalized Real	Spread	True
ID	Integer	Single	Inspect.

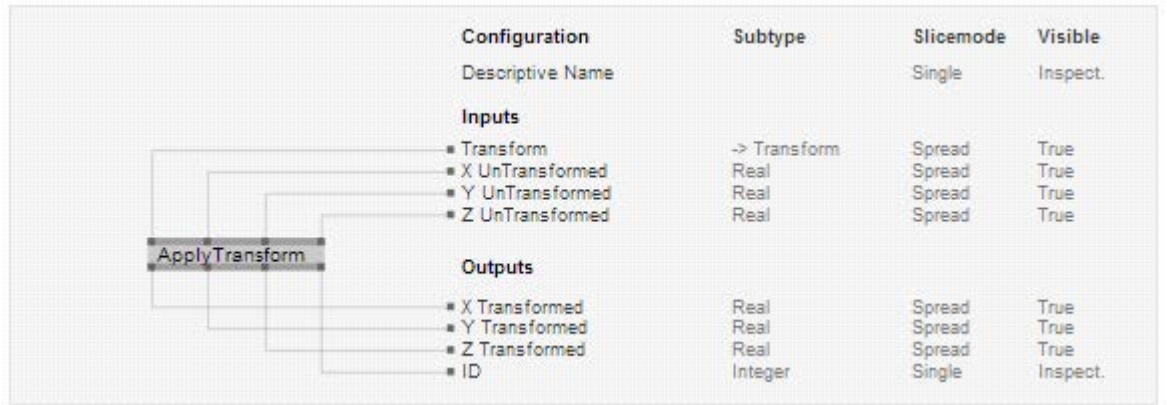
AND (Boolean)



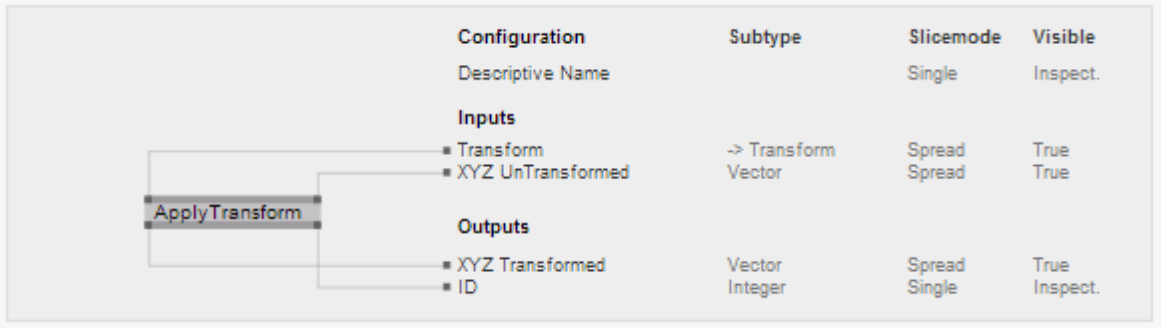
AND (Boolean Spectral)



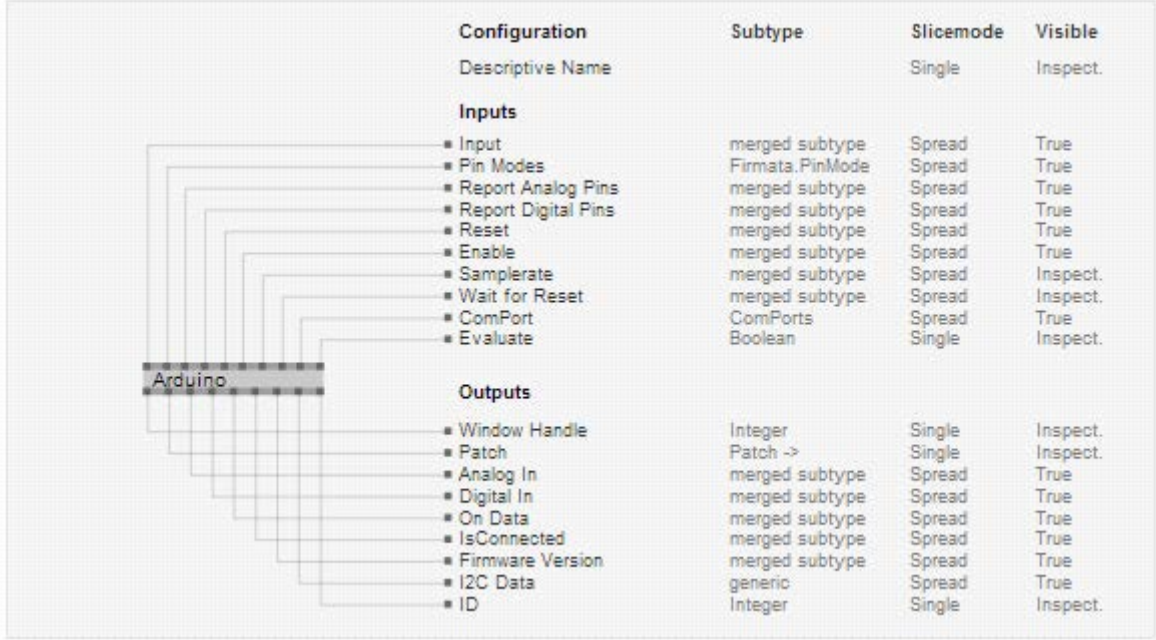
ApplyTransform (Transform)



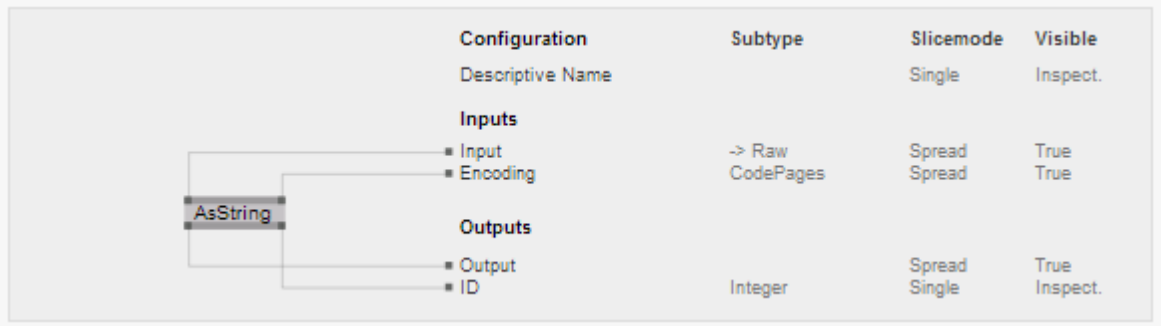
ApplyTransform (Transform Vector)



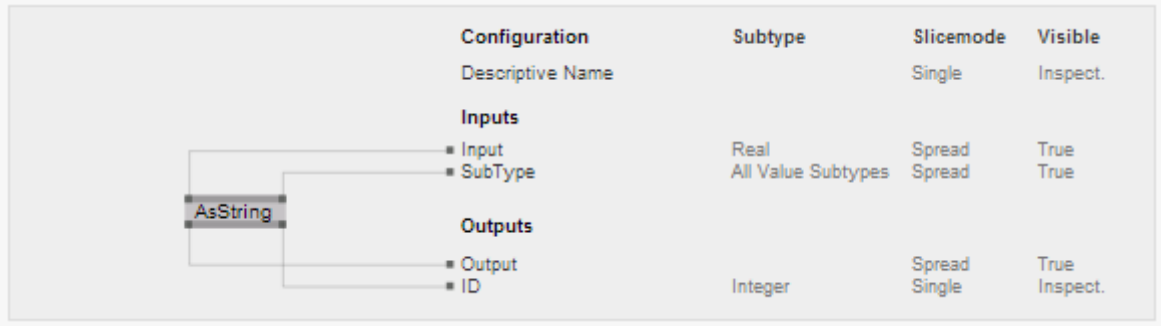
Arduino (Devices StandardFirmata 2.x)



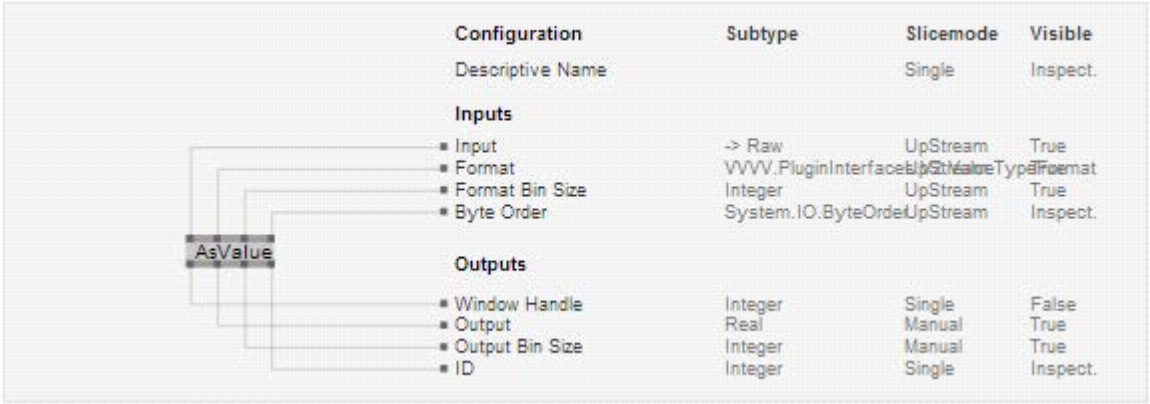
AsString (Raw)



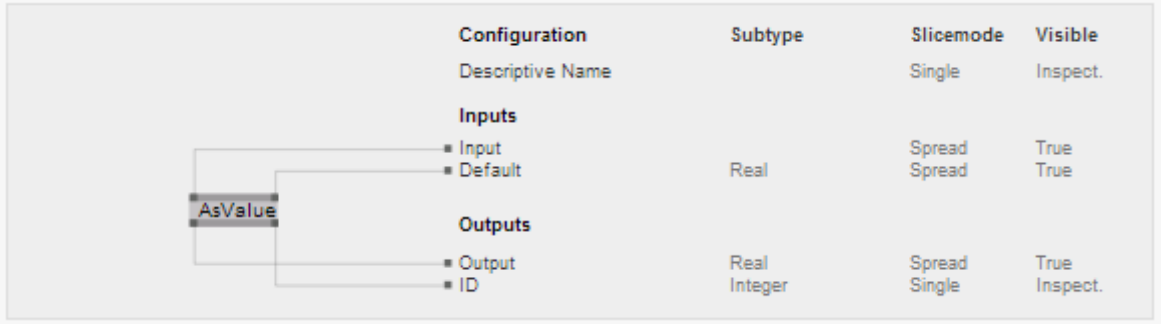
AsString (Value)



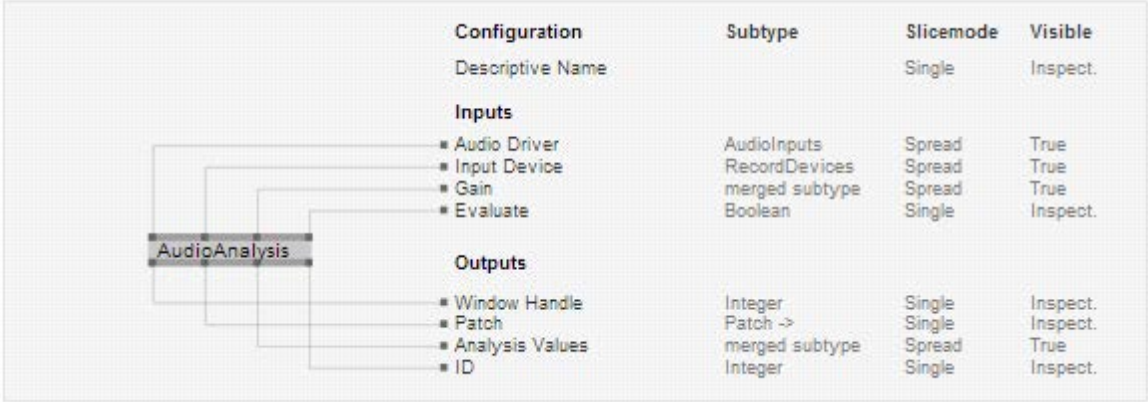
AsValue (Raw)



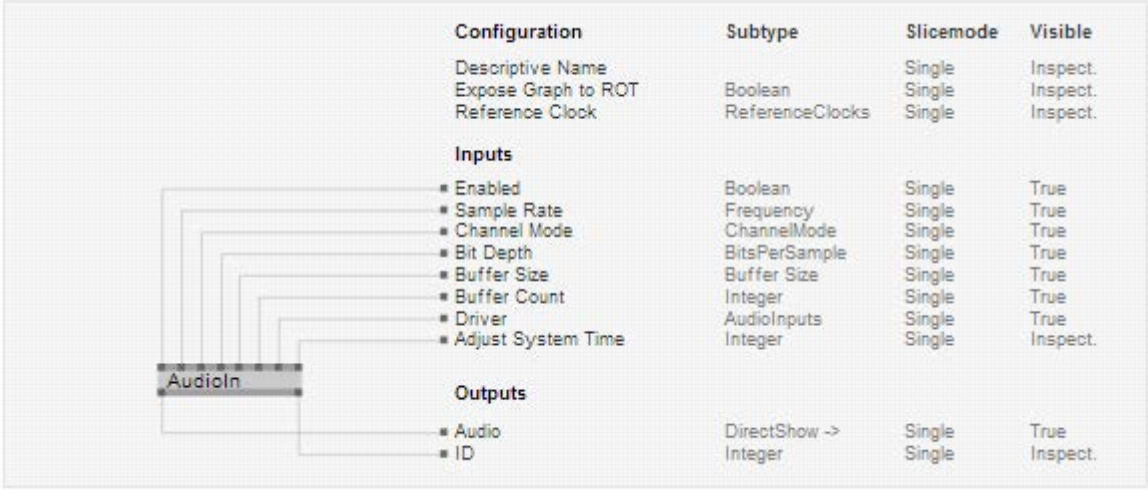
AsValue (String)



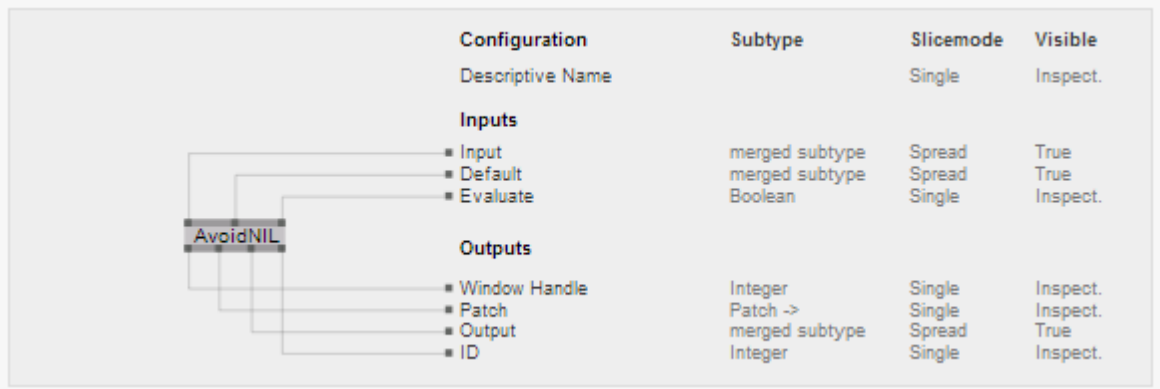
AudioAnalysis (DShow9)



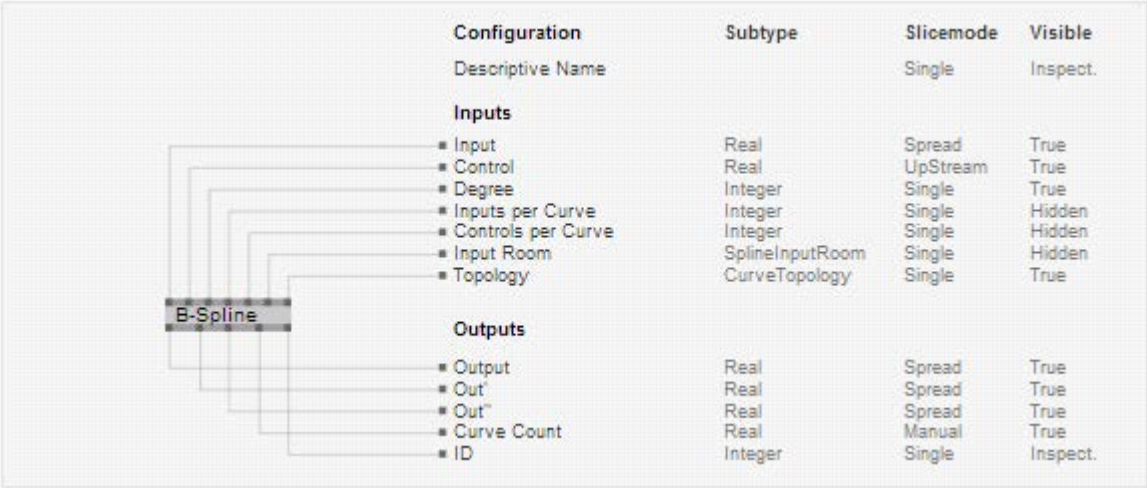
AudioIn (DShow9)



AvoidNIL (String)

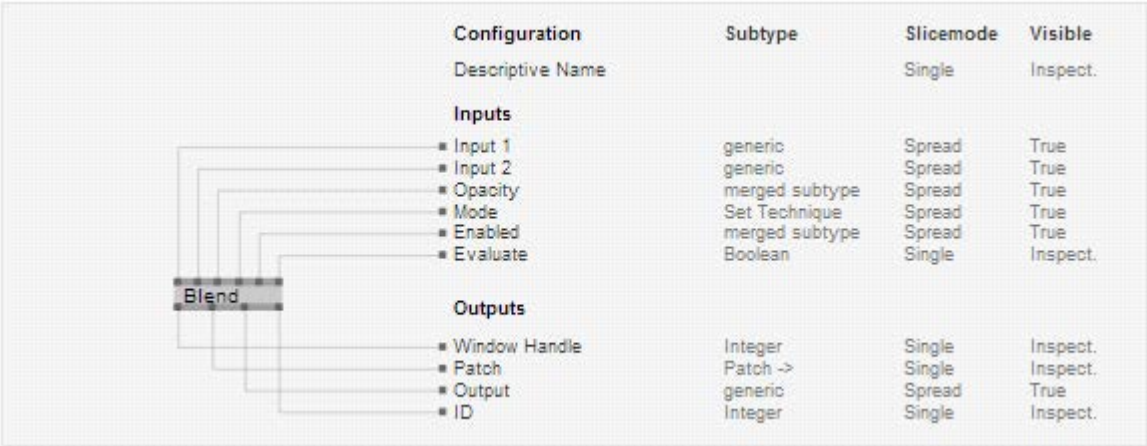


B-Spline (Value)



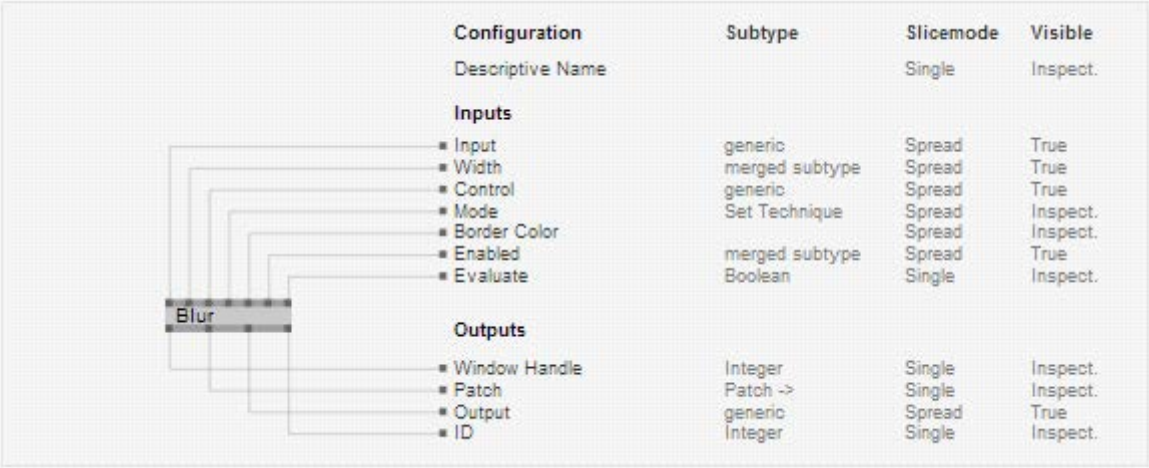
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Input	Real	Spread	True
Control	Real	UpStream	True
Degree	Integer	Single	True
Inputs per Curve	Integer	Single	Hidden
Controls per Curve	Integer	Single	Hidden
Input Room	SplineInputRoom	Single	Hidden
Topology	CurveTopology	Single	True
Outputs			
Output	Real	Spread	True
Out'	Real	Spread	True
Out"	Real	Spread	True
Curve Count	Real	Manual	True
ID	Integer	Single	Inspect.

Blend (EX9.Texture Mixer)



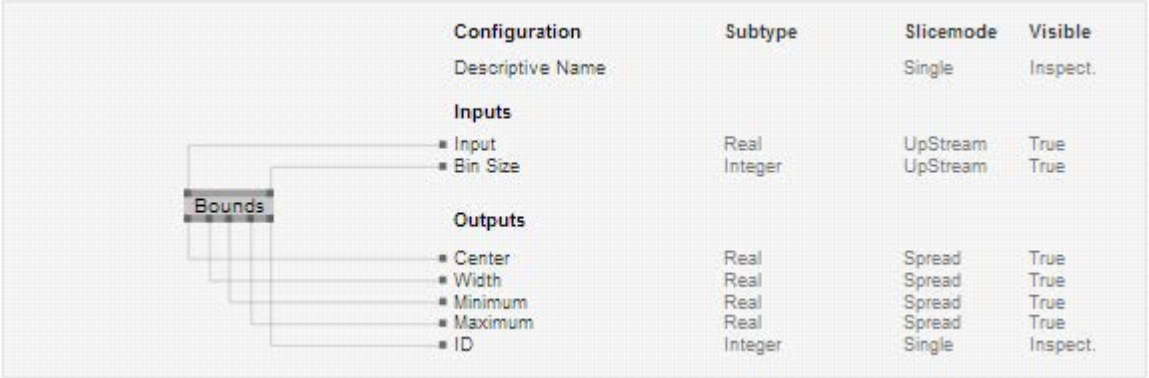
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Input 1	generic	Spread	True
Input 2	generic	Spread	True
Opacity	merged subtype	Spread	True
Mode	Set Technique	Spread	True
Enabled	merged subtype	Spread	True
Evaluate	Boolean	Single	Inspect.
Outputs			
Window Handle	Integer	Single	Inspect.
Patch	Patch ->	Single	Inspect.
Output	generic	Spread	True
ID	Integer	Single	Inspect.

Blur (EX9.Texture Filter)



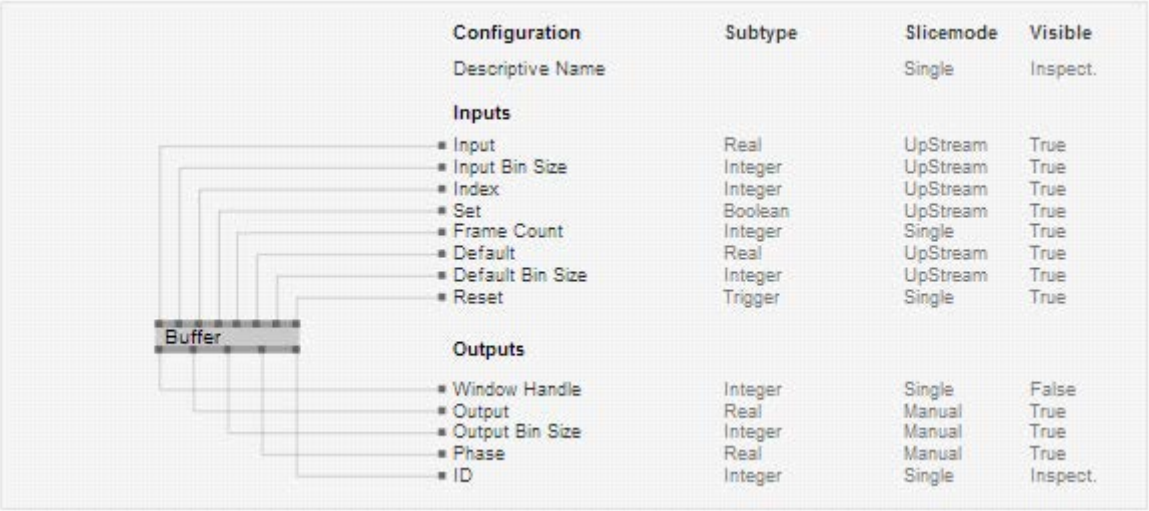
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Input	generic	Spread	True
Width	merged subtype	Spread	True
Control	generic	Spread	True
Mode	Set Technique	Spread	Inspect.
Border Color		Spread	Inspect.
Enabled	merged subtype	Spread	True
Evaluate	Boolean	Single	Inspect.
Outputs			
Window Handle	Integer	Single	Inspect.
Patch	Patch ->	Single	Inspect.
Output	generic	Spread	True
ID	Integer	Single	Inspect.

Bounds (Spectral)



Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Input	Real	UpStream	True
Bin Size	Integer	UpStream	True
Outputs			
Center	Real	Spread	True
Width	Real	Spread	True
Minimum	Real	Spread	True
Maximum	Real	Spread	True
ID	Integer	Single	Inspect.

Buffer (Spreads)

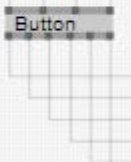


Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Input	Real	UpStream	True
Input Bin Size	Integer	UpStream	True
Index	Integer	UpStream	True
Set	Boolean	UpStream	True
Frame Count	Integer	Single	True
Default	Real	UpStream	True
Default Bin Size	Integer	UpStream	True
Reset	Trigger	Single	True
Outputs			
Window Handle	Integer	Single	False
Output	Real	Manual	True
Output Bin Size	Integer	Manual	True
Phase	Real	Manual	True
ID	Integer	Single	Inspect.

Button (3d Mesh)

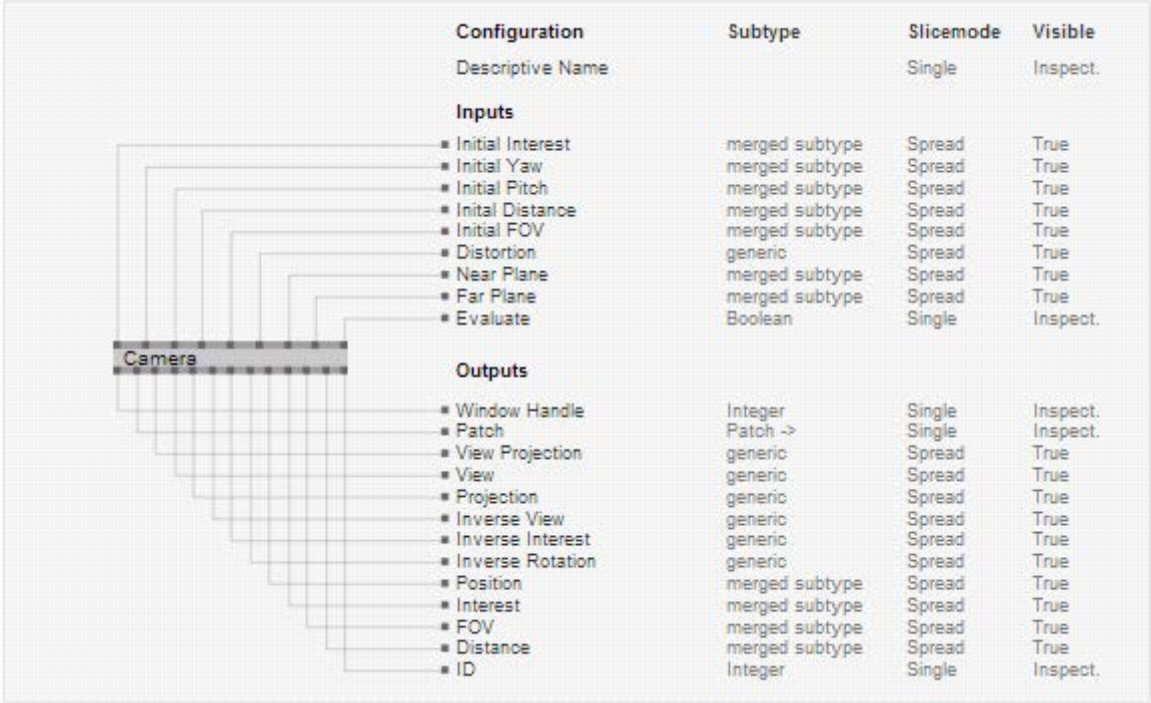
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Mesh	generic	Spread	True
Transform Mesh	generic	Spread	True
Mouse	generic	Spread	True
View Projection	generic	Spread	True
Evaluate	Boolean	Single	Inspect.
Outputs			
Window Handle	Integer	Single	Inspect.
Patch	Patch ->	Single	Inspect.
Mouse Over	merged subtype	Spread	True
Index of Nearest	merged subtype	Spread	True
Cursor Transform	generic	Spread	True
ID	Integer	Single	Inspect.

Button (3d Quad)

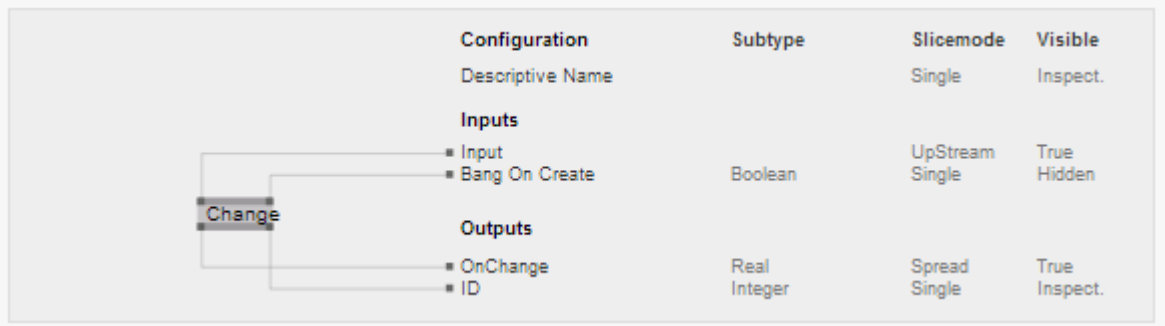


Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Transform Quad	generic	Spread	True
Mouse	generic	Spread	True
View Projection	generic	Spread	True
Evaluate	Boolean	Single	Inspect.
Outputs			
Window Handle	Integer	Single	Inspect.
Patch	Patch ->	Single	Inspect.
Mouse Over	merged subtype	Spread	True
Index of Nearest	merged subtype	Spread	True
Cursor Transform	generic	Spread	True
ID	Integer	Single	Inspect.

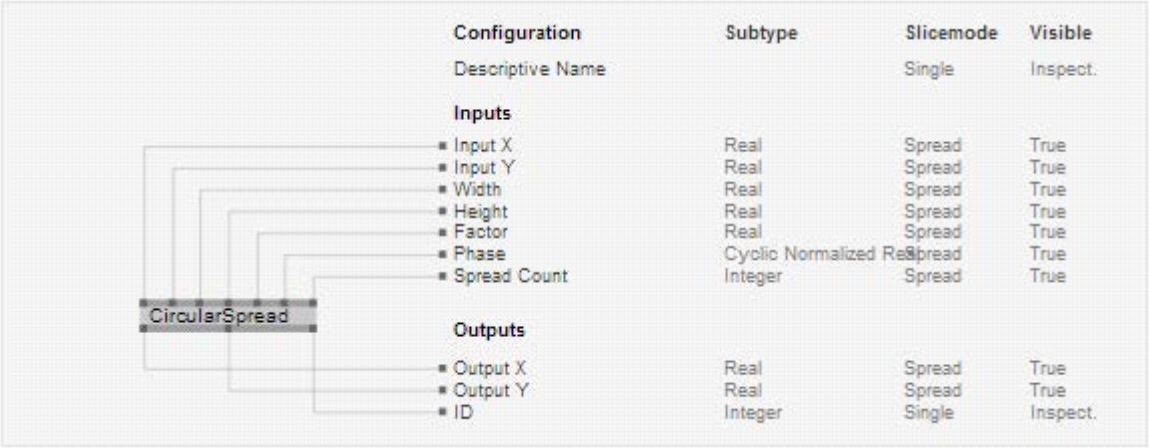
Camera (Transform Softimage)



Change (String)

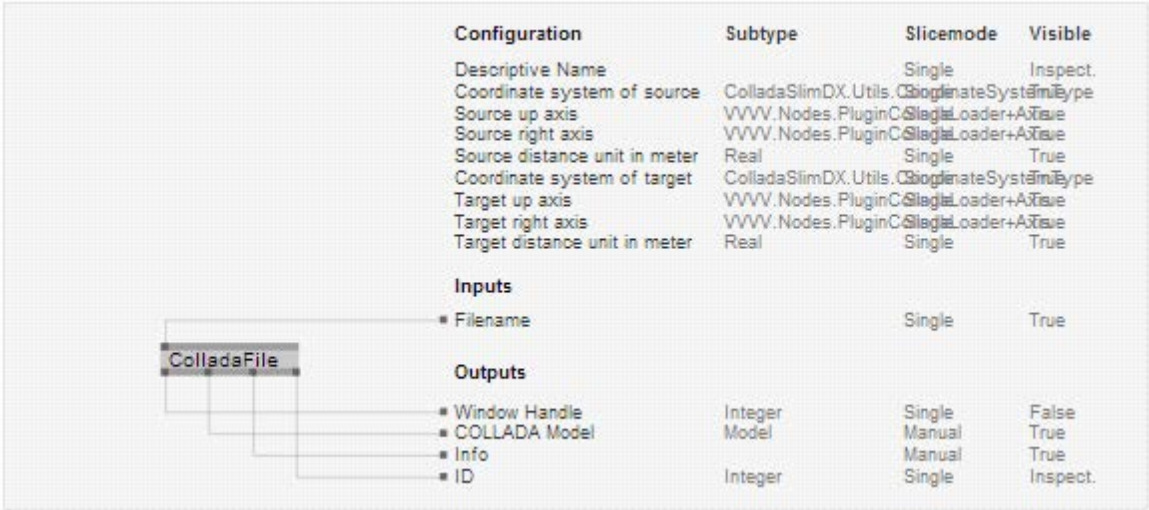


CircularSpread (Spreads)



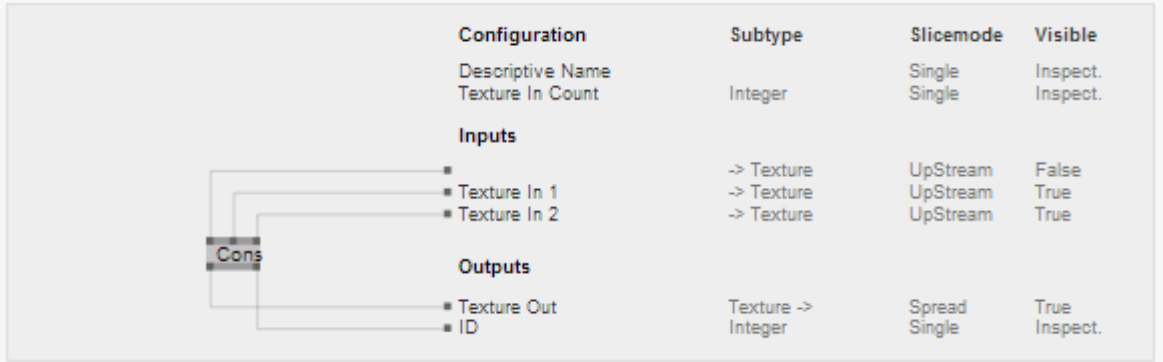
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Input X	Real	Spread	True
Input Y	Real	Spread	True
Width	Real	Spread	True
Height	Real	Spread	True
Factor	Real	Spread	True
Phase	Cyclic Normalized Real	Spread	True
Spread Count	Integer	Spread	True
Outputs			
Output X	Real	Spread	True
Output Y	Real	Spread	True
ID	Integer	Single	Inspect.

ColladaFile (EX9.Geometry)



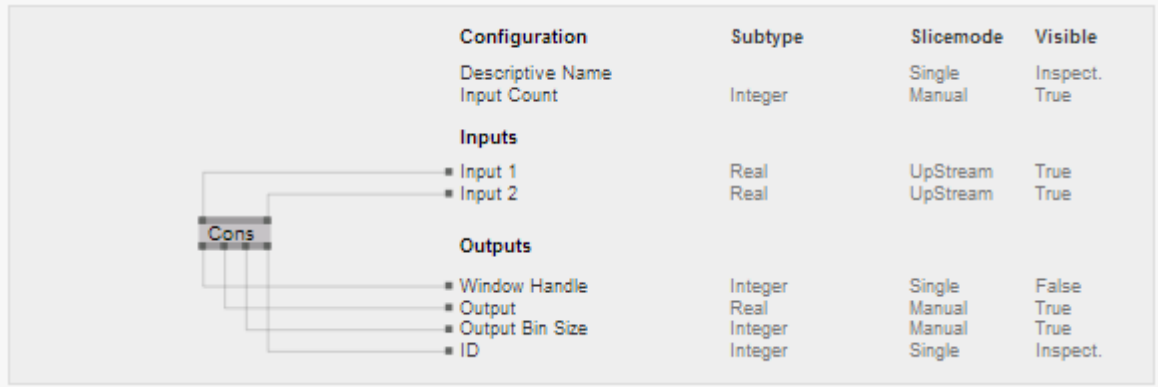
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Coordinate system of source	ColladaSlimDX.Utils.CoordinateSystemType	Single	True
Source up axis	VVVV.Nodes.PluginCollada.ColladaLoader+AxisT	Single	True
Source right axis	VVVV.Nodes.PluginCollada.ColladaLoader+AxisT	Single	True
Source distance unit in meter	Real	Single	True
Coordinate system of target	ColladaSlimDX.Utils.CoordinateSystemType	Single	True
Target up axis	VVVV.Nodes.PluginCollada.ColladaLoader+AxisT	Single	True
Target right axis	VVVV.Nodes.PluginCollada.ColladaLoader+AxisT	Single	True
Target distance unit in meter	Real	Single	True
Inputs			
Filename		Single	True
Outputs			
Window Handle	Integer	Single	False
COLLADA Model	Model	Manual	True
Info		Manual	True
ID	Integer	Single	Inspect.

Cons (EX9.Texture)



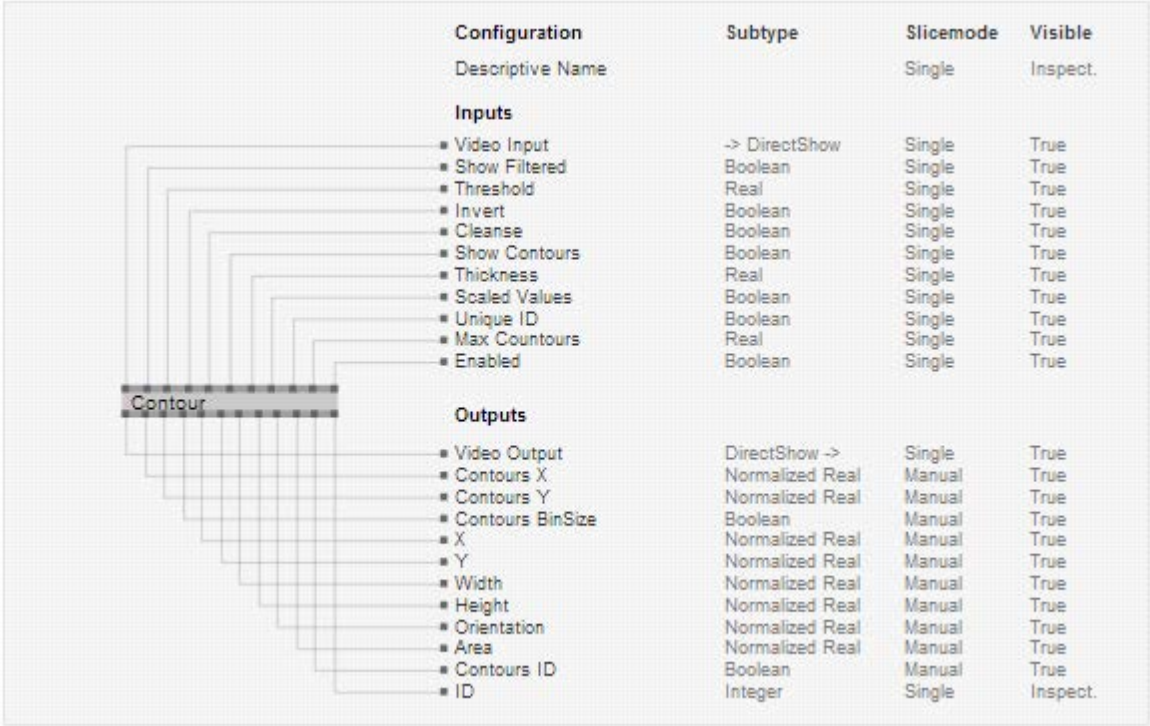
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Texture In Count	Integer	Single	Inspect.
Inputs			
	-> Texture	UpStream	False
Texture In 1	-> Texture	UpStream	True
Texture In 2	-> Texture	UpStream	True
Outputs			
Texture Out	Texture ->	Spread	True
ID	Integer	Single	Inspect.

Cons (Spreads)



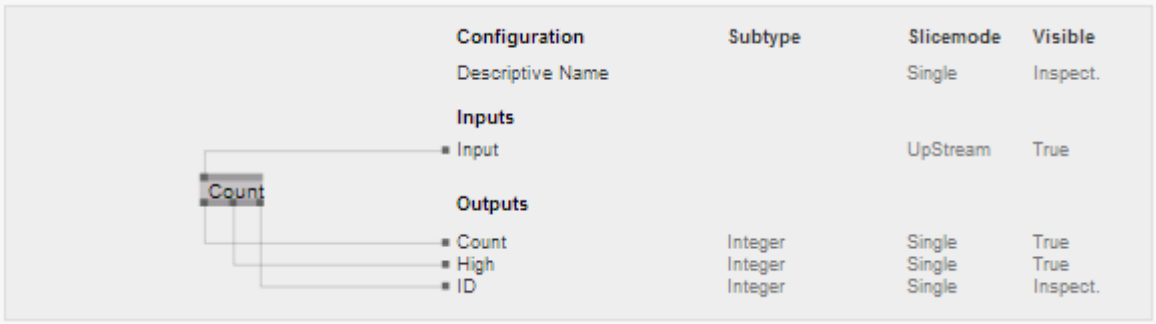
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Input Count	Integer	Manual	True
Inputs			
Input 1	Real	UpStream	True
Input 2	Real	UpStream	True
Outputs			
Window Handle	Integer	Single	False
Output	Real	Manual	True
Output Bin Size	Integer	Manual	True
ID	Integer	Single	Inspect.

Contour (FreeFrame DShow9)



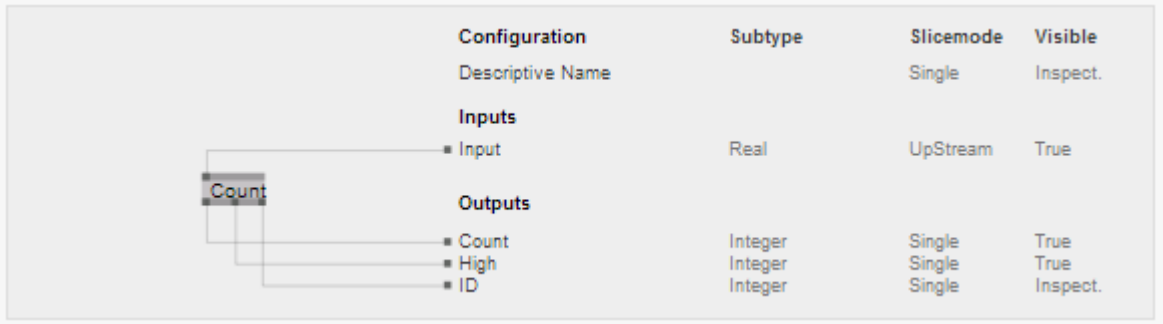
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Video Input	-> DirectShow	Single	True
Show Filtered	Boolean	Single	True
Threshold	Real	Single	True
Invert	Boolean	Single	True
Cleanse	Boolean	Single	True
Show Contours	Boolean	Single	True
Thickness	Real	Single	True
Scaled Values	Boolean	Single	True
Unique ID	Boolean	Single	True
Max Countours	Real	Single	True
Enabled	Boolean	Single	True
Outputs			
Video Output	DirectShow ->	Single	True
Contours X	Normalized Real	Manual	True
Contours Y	Normalized Real	Manual	True
Contours BinSize	Boolean	Manual	True
X	Normalized Real	Manual	True
Y	Normalized Real	Manual	True
Width	Normalized Real	Manual	True
Height	Normalized Real	Manual	True
Orientation	Normalized Real	Manual	True
Area	Normalized Real	Manual	True
Contours ID	Boolean	Manual	True
ID	Integer	Single	Inspect.

Count (String)



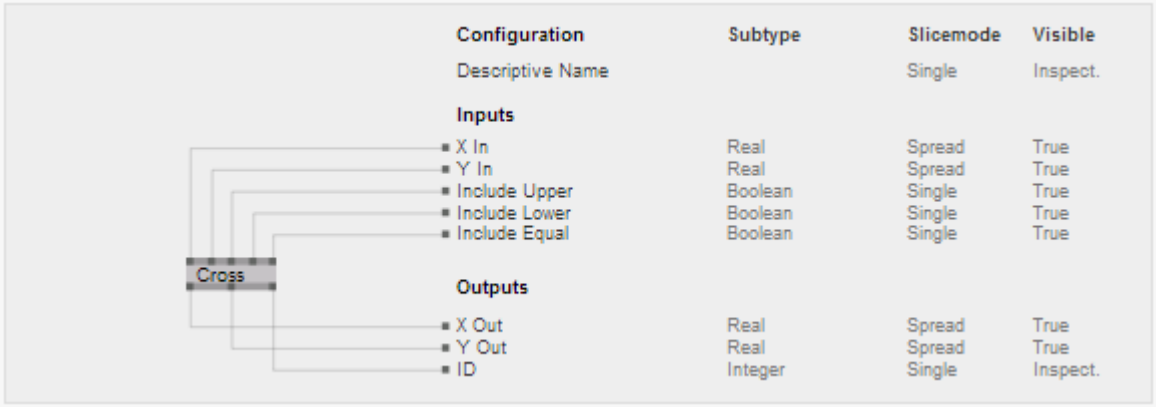
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Input		UpStream	True
Outputs			
Count	Integer	Single	True
High	Integer	Single	True
ID	Integer	Single	Inspect.

Count (Value)



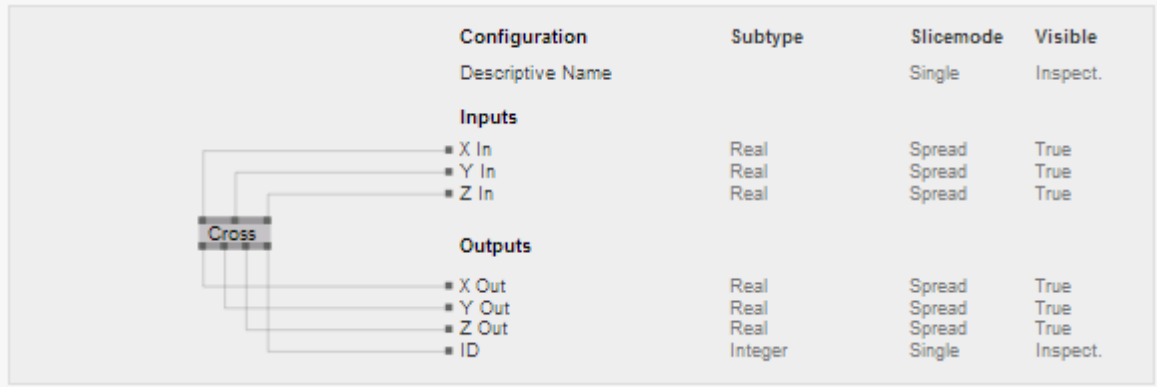
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Input	Real	UpStream	True
Outputs			
Count	Integer	Single	True
High	Integer	Single	True
ID	Integer	Single	Inspect.

Cross (2d)



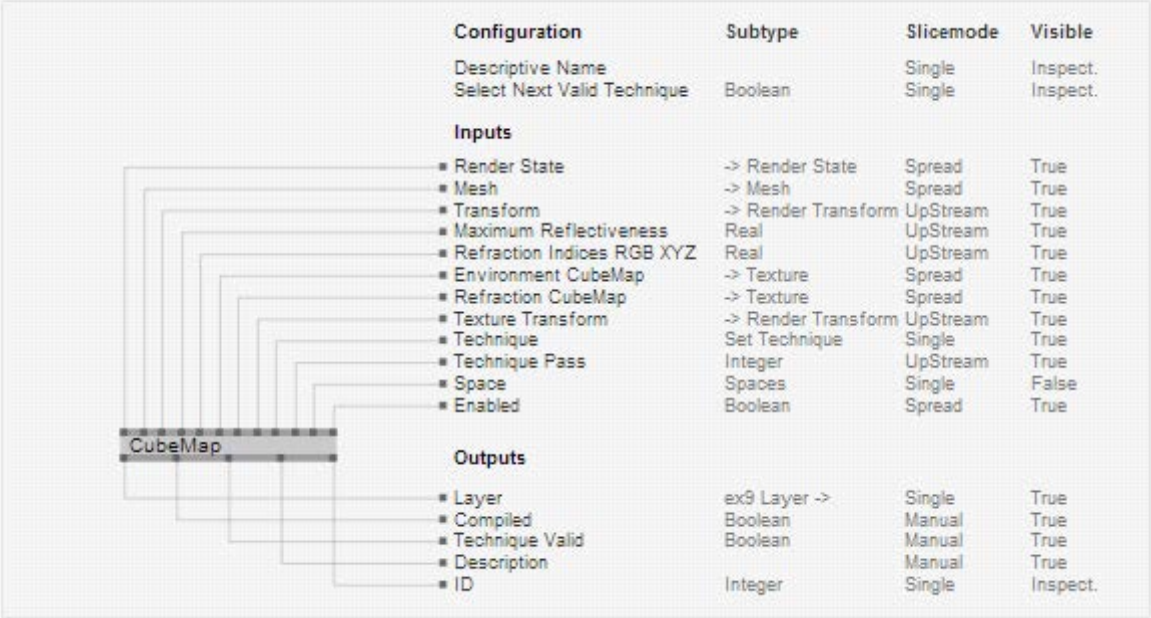
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
X In	Real	Spread	True
Y In	Real	Spread	True
Include Upper	Boolean	Single	True
Include Lower	Boolean	Single	True
Include Equal	Boolean	Single	True
Outputs			
X Out	Real	Spread	True
Y Out	Real	Spread	True
ID	Integer	Single	Inspect.

Cross (3d)



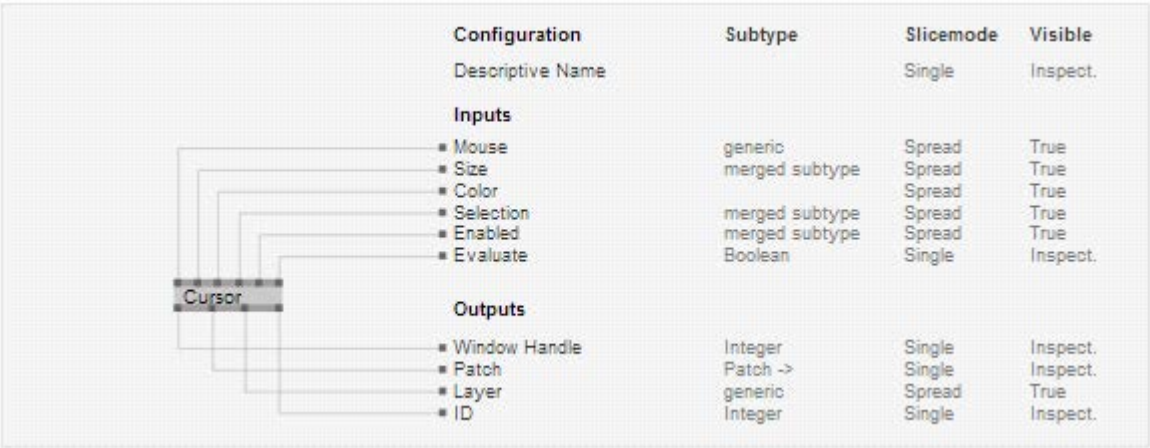
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
X In	Real	Spread	True
Y In	Real	Spread	True
Z In	Real	Spread	True
Outputs			
X Out	Real	Spread	True
Y Out	Real	Spread	True
Z Out	Real	Spread	True
ID	Integer	Single	Inspect.

CubeMap (EX9.Effect)



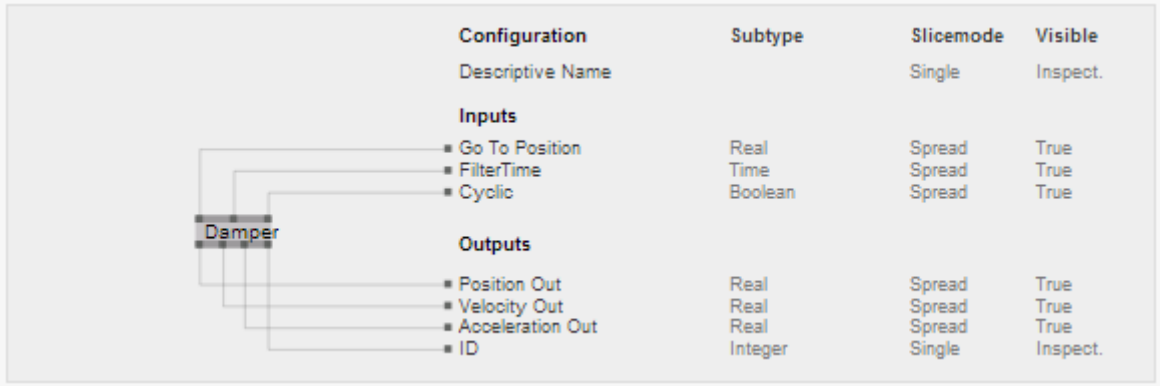
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Select Next Valid Technique	Boolean	Single	Inspect.
Inputs			
Render State	-> Render State	Spread	True
Mesh	-> Mesh	Spread	True
Transform	-> Render Transform	UpStream	True
Maximum Reflectiveness	Real	UpStream	True
Refraction Indices RGB XYZ	Real	UpStream	True
Environment CubeMap	-> Texture	Spread	True
Refraction CubeMap	-> Texture	Spread	True
Texture Transform	-> Render Transform	UpStream	True
Technique	Set Technique	Single	True
Technique Pass	Integer	UpStream	True
Space	Spaces	Single	False
Enabled	Boolean	Spread	True
Outputs			
Layer	ex9 Layer ->	Single	True
Compiled	Boolean	Manual	True
Technique Valid	Boolean	Manual	True
Description		Manual	True
ID	Integer	Single	Inspect.

Cursor (DX9)



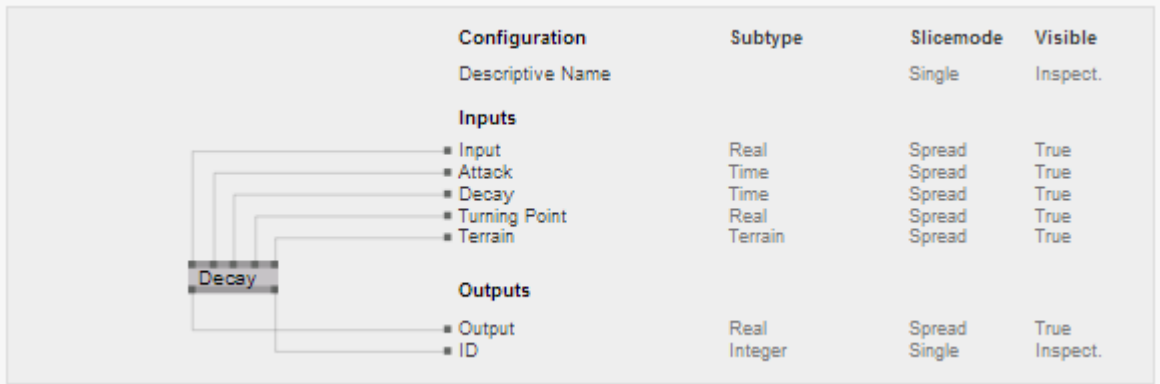
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Mouse	generic	Spread	True
Size	merged subtype	Spread	True
Color		Spread	True
Selection	merged subtype	Spread	True
Enabled	merged subtype	Spread	True
Evaluate	Boolean	Single	Inspect.
Outputs			
Window Handle	Integer	Single	Inspect.
Patch	Patch ->	Single	Inspect.
Layer	generic	Spread	True
ID	Integer	Single	Inspect.

Damper (Animation)



Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Go To Position	Real	Spread	True
FilterTime	Time	Spread	True
Cyclic	Boolean	Spread	True
Outputs			
Position Out	Real	Spread	True
Velocity Out	Real	Spread	True
Acceleration Out	Real	Spread	True
ID	Integer	Single	Inspect.

Decay (Animation)



Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Input	Real	Spread	True
Attack	Time	Spread	True
Decay	Time	Spread	True
Turning Point	Real	Spread	True
Terrain	Terrain	Spread	True
Outputs			
Output	Real	Spread	True
ID	Integer	Single	Inspect.

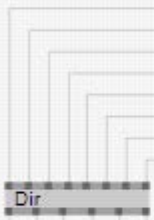
Delay (Animation)

Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Input	Real	Spread	True
Time	Time	Spread	True
Insert	Boolean	Spread	True
Mode	DelayInterpolationMode	Spread	True
Reset	Boolean	Spread	True
Reset Value	Real	Spread	True
Outputs			
Output	Real	Spread	True
Debug		Spread	True
ID	Integer	Single	Inspect.

DeNiro (Animation)

Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Automatic Update	UpdateFilterGraph	Single	Inspect.
Inputs			
Go To Position	Real	Spread	True
Go	Boolean	Spread	True
Position In	Real	Spread	True
Velocity In	Real	Spread	True
Force Update	Bang	Spread	True
Pause	Boolean	Spread	True
Select Input	DeNiroInputSelect	Single	True
Go To Velocity	Real	Spread	True
Acceleration	Real	Spread	True
Cyclic	Boolean	Single	True
Constant Drive	Normalized Real	Spread	True
Max Velocity	Real	Spread	True
Outputs			
Position Out	Real	Spread	True
Velocity Out	Real	Spread	True
Acceleration Out	Real	Spread	True
ID	Integer	Single	Inspect.

Dir (File Advanced)



Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Directory		Single	True
Include Subdirectories	Boolean	Single	True
Show Short Filename	Boolean	Single	True
Mask		UpStream	True
Sort Order	Sort Order	Single	True
Count	Integer	Single	True
Count Selection	Count Selection	Single	True
Update	Trigger	Single	True
Outputs			
Window Handle	Integer	Single	False
Filenames		Manual	True
Short Filenames		Manual	True
File Count	Integer	Single	True
Message		Single	Hidden
ID	Integer	Single	Inspect.

DMX (Network Artnet Receiver)

Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
SubnetID	Integer	Single	Inspect.
UniverseID	Integer	Single	Inspect.
Inputs			
Outputs			
Input	Normalized Real	Manual	True
TimeOut	Time	Single	True
ID	Integer	Single	Inspect.

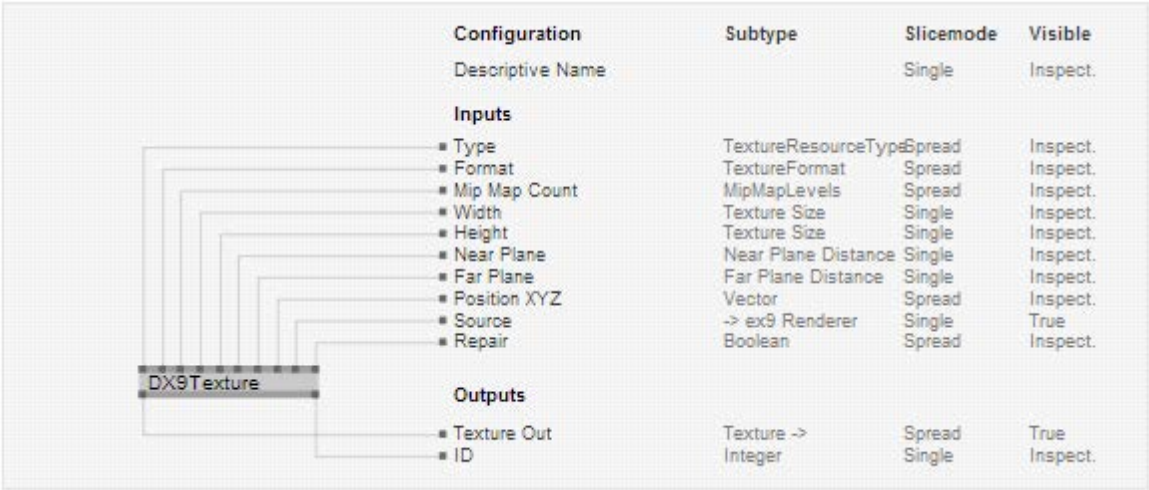
DMX (Network Artnet Sender)

Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
SubnetID	Integer	Single	Inspect.
UniverseID	Integer	Single	Inspect.
Inputs			
Input	Normalized Real	Manual	True
Do Send	Bang	Spread	True
Destination Address		Single	True
Outputs			
ID	Integer	Single	Inspect.

DrawFixed (EX9.Effect)

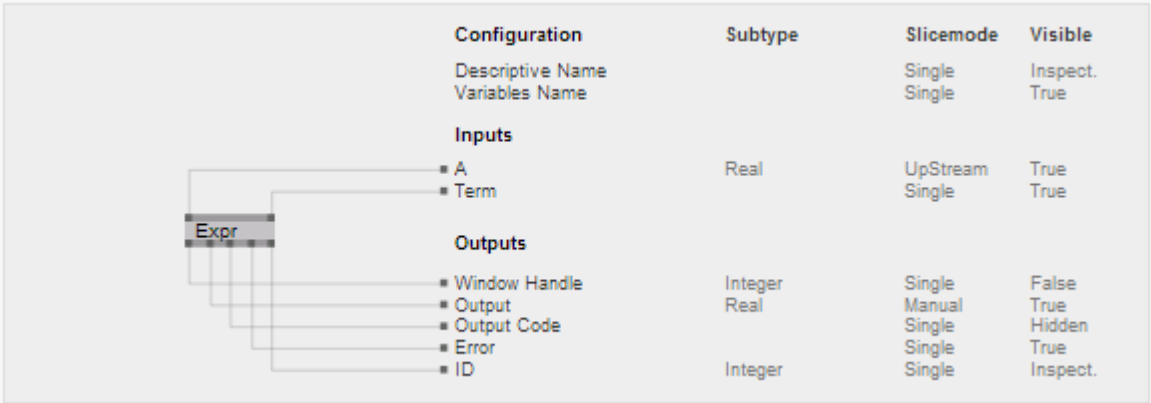
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Render State	-> Render State	Spread	True
Mesh	-> X Mesh	Spread	True
Transform	-> Render Transform	Spread	True
Texture	-> Texture	Spread	True
Texture Transform	-> Matrix Transform	Spread	True
Sampler State	-> Sampler State	Spread	True
Symmetric Texture Transform	Boolean	Single	Inspect.
Space	Spaces	Single	False
Enabled	Boolean	Single	Hidden
Outputs			
Layer	ex9 Layer ->	Single	True
ID	Integer	Single	Inspect.

DX9Texture (EX9.Texture)



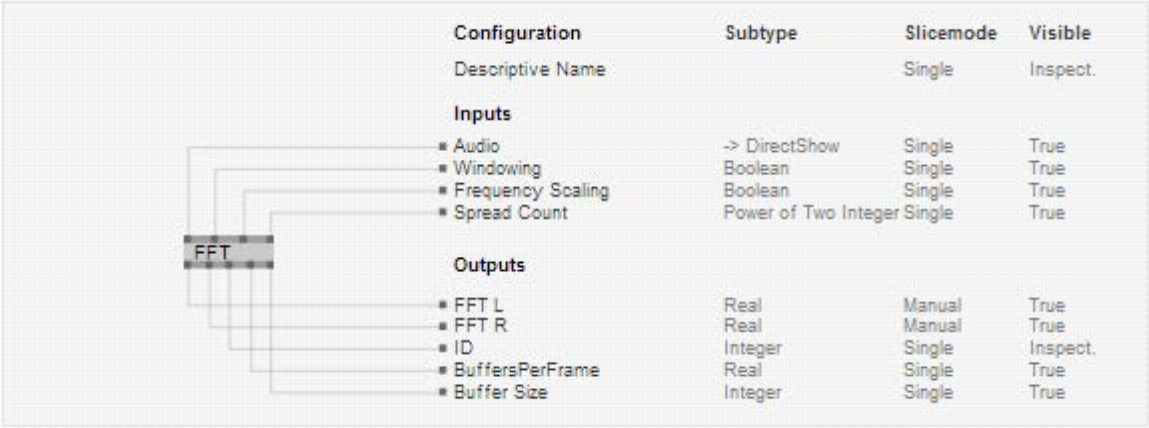
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Type	TextureResourceTyp	Spread	Inspect.
Format	TextureFormat	Spread	Inspect.
Mip Map Count	MipMapLevels	Spread	Inspect.
Width	Texture Size	Single	Inspect.
Height	Texture Size	Single	Inspect.
Near Plane	Near Plane Distance	Single	Inspect.
Far Plane	Far Plane Distance	Single	Inspect.
Position XYZ	Vector	Spread	Inspect.
Source	-> ex9 Renderer	Single	True
Repair	Boolean	Spread	Inspect.
Outputs			
Texture Out	Texture ->	Spread	True
ID	Integer	Single	Inspect.

Expr (Value Advanced)



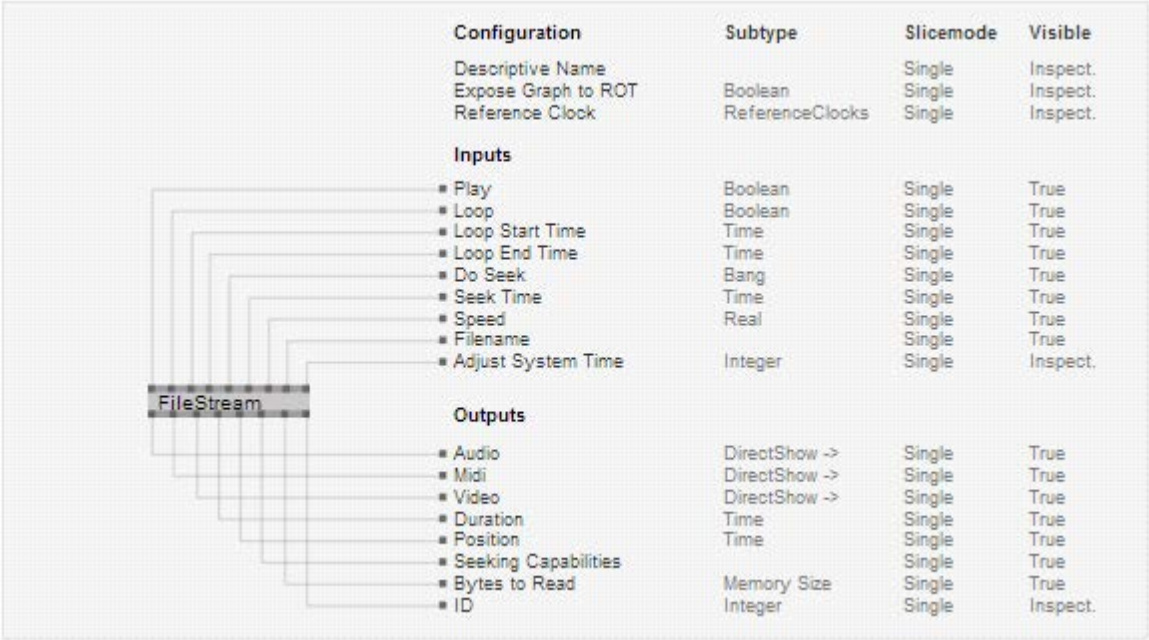
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Variables Name		Single	True
Inputs			
A	Real	UpStream	True
Term		Single	True
Outputs			
Window Handle	Integer	Single	False
Output	Real	Manual	True
Output Code		Single	Hidden
Error		Single	True
ID	Integer	Single	Inspect.

FFT (DShow9)



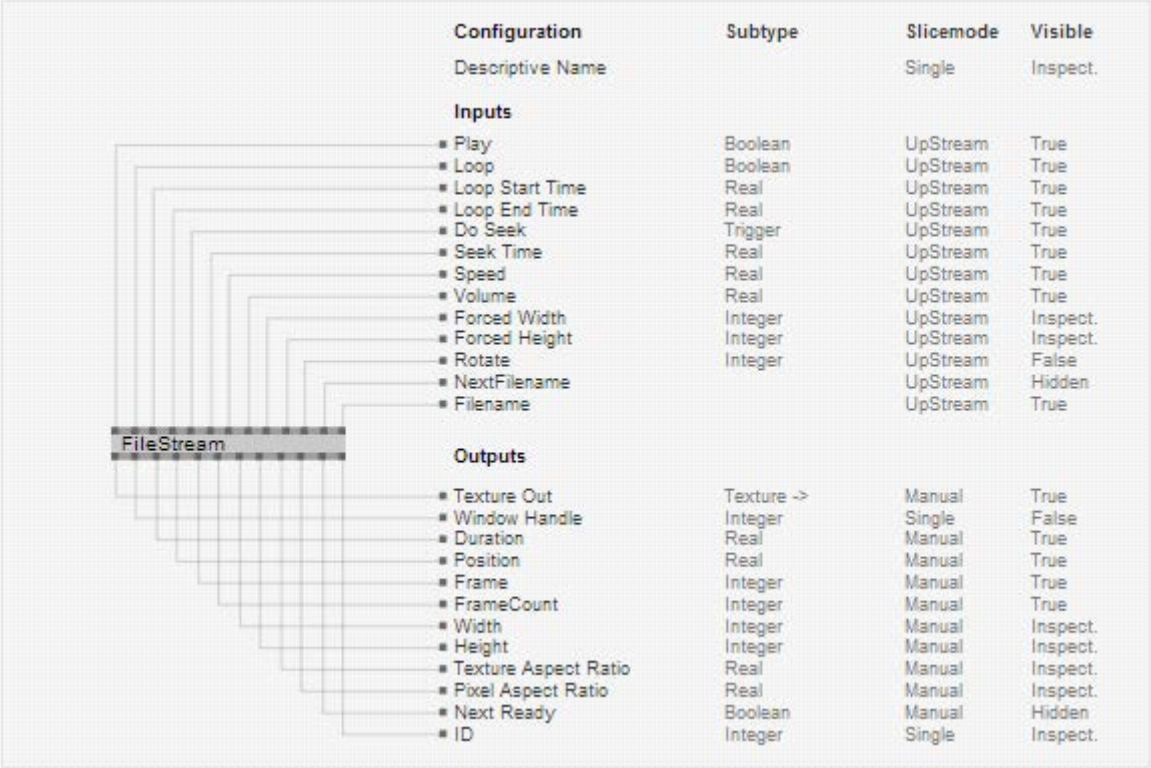
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Audio	-> DirectShow	Single	True
Windowing	Boolean	Single	True
Frequency Scaling	Boolean	Single	True
Spread Count	Power of Two Integer	Single	True
Outputs			
FFT L	Real	Manual	True
FFT R	Real	Manual	True
ID	Integer	Single	Inspect.
BuffersPerFrame	Real	Single	True
Buffer Size	Integer	Single	True

FileStream (DShow9)



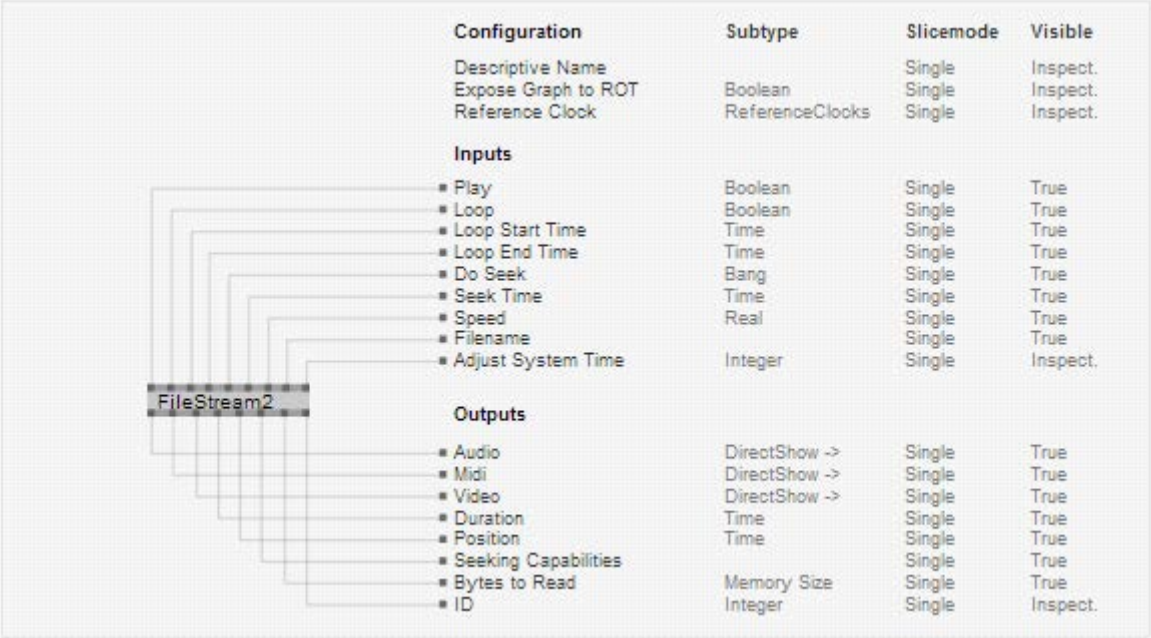
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Expose Graph to ROT	Boolean	Single	Inspect.
Reference Clock	ReferenceClocks	Single	Inspect.
Inputs			
Play	Boolean	Single	True
Loop	Boolean	Single	True
Loop Start Time	Time	Single	True
Loop End Time	Time	Single	True
Do Seek	Bang	Single	True
Seek Time	Time	Single	True
Speed	Real	Single	True
Filename		Single	True
Adjust System Time	Integer	Single	Inspect.
Outputs			
Audio	DirectShow ->	Single	True
Midi	DirectShow ->	Single	True
Video	DirectShow ->	Single	True
Duration	Time	Single	True
Position	Time	Single	True
Seeking Capabilities		Single	True
Bytes to Read	Memory Size	Single	True
ID	Integer	Single	Inspect.

FileStream (EX9.Texture VLC)



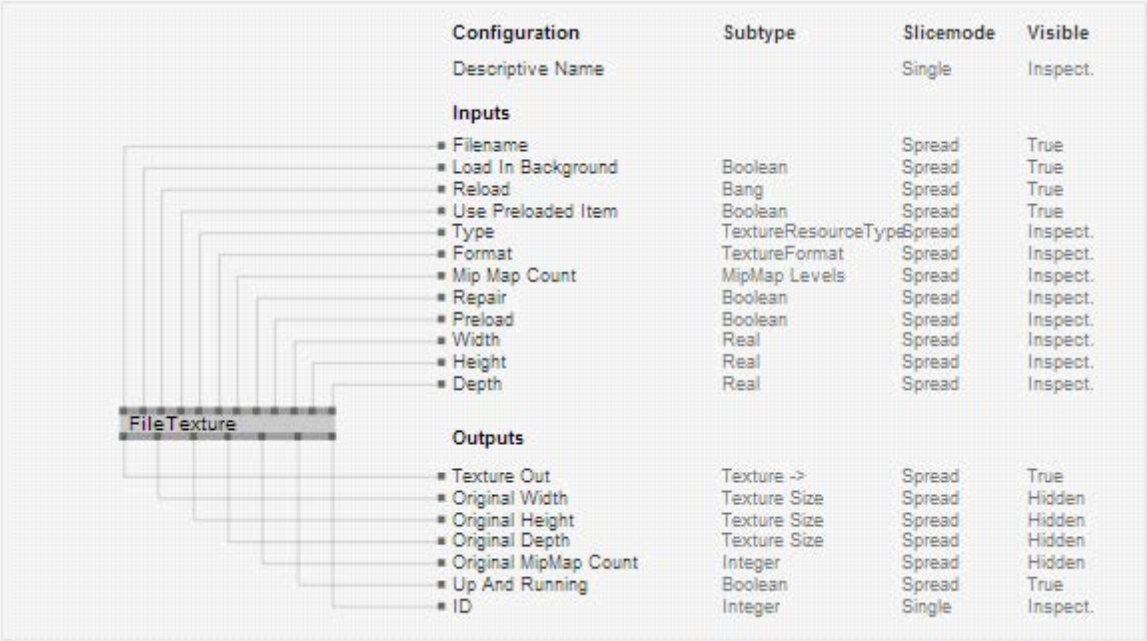
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Play	Boolean	UpStream	True
Loop	Boolean	UpStream	True
Loop Start Time	Real	UpStream	True
Loop End Time	Real	UpStream	True
Do Seek	Trigger	UpStream	True
Seek Time	Real	UpStream	True
Speed	Real	UpStream	True
Volume	Real	UpStream	True
Forced Width	Integer	UpStream	Inspect.
Forced Height	Integer	UpStream	Inspect.
Rotate	Integer	UpStream	False
NextFilename		UpStream	Hidden
Filename		UpStream	True
Outputs			
Texture Out	Texture ->	Manual	True
Window Handle	Integer	Single	False
Duration	Real	Manual	True
Position	Real	Manual	True
Frame	Integer	Manual	True
FrameCount	Integer	Manual	True
Width	Integer	Manual	Inspect.
Height	Integer	Manual	Inspect.
Texture Aspect Ratio	Real	Manual	Inspect.
Pixel Aspect Ratio	Real	Manual	Inspect.
Next Ready	Boolean	Manual	Hidden
ID	Integer	Single	Inspect.

FileStream2 (DShow9)



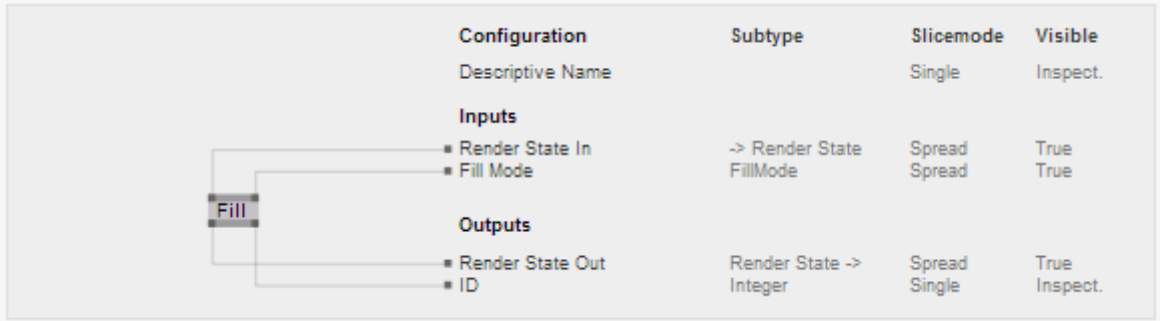
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Expose Graph to ROT	Boolean	Single	Inspect.
Reference Clock	ReferenceClocks	Single	Inspect.
Inputs			
Play	Boolean	Single	True
Loop	Boolean	Single	True
Loop Start Time	Time	Single	True
Loop End Time	Time	Single	True
Do Seek	Bang	Single	True
Seek Time	Time	Single	True
Speed	Real	Single	True
Filename		Single	True
Adjust System Time	Integer	Single	Inspect.
Outputs			
Audio	DirectShow ->	Single	True
Midi	DirectShow ->	Single	True
Video	DirectShow ->	Single	True
Duration	Time	Single	True
Position	Time	Single	True
Seeking Capabilities		Single	True
Bytes to Read	Memory Size	Single	True
ID	Integer	Single	Inspect.

FileTexture (EX9.Texture)



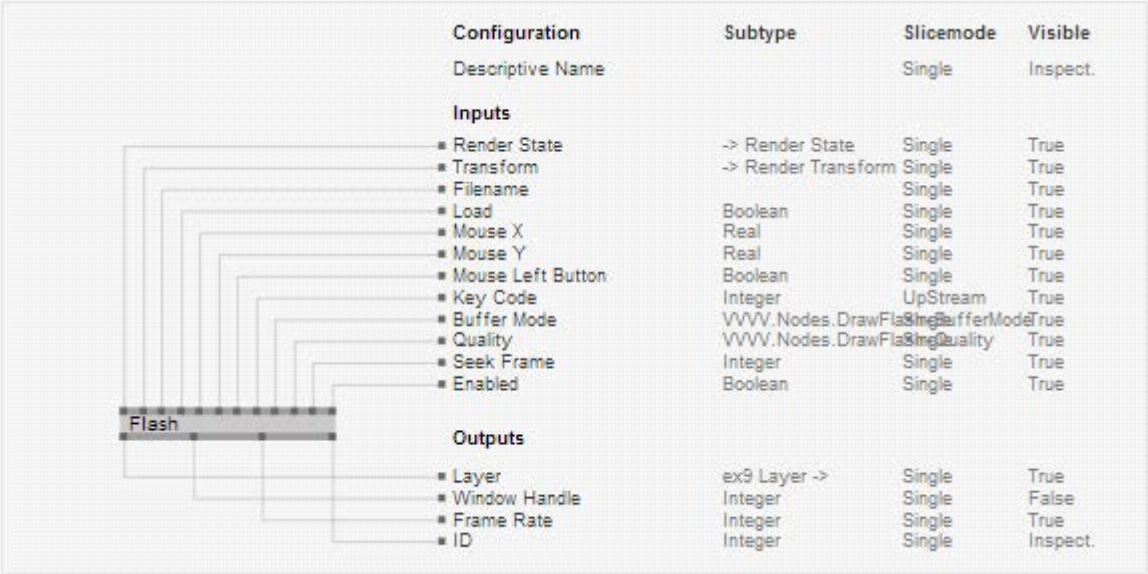
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Filename		Spread	True
Load In Background	Boolean	Spread	True
Reload	Bang	Spread	True
Use Preloaded Item	Boolean	Spread	True
Type	TextureResourceType	Spread	Inspect.
Format	TextureFormat	Spread	Inspect.
Mip Map Count	MipMap Levels	Spread	Inspect.
Repair	Boolean	Spread	Inspect.
Preload	Boolean	Spread	Inspect.
Width	Real	Spread	Inspect.
Height	Real	Spread	Inspect.
Depth	Real	Spread	Inspect.
Outputs			
Texture Out	Texture ->	Spread	True
Original Width	Texture Size	Spread	Hidden
Original Height	Texture Size	Spread	Hidden
Original Depth	Texture Size	Spread	Hidden
Original MipMap Count	Integer	Spread	Hidden
Up And Running	Boolean	Spread	True
ID	Integer	Single	Inspect.

Fill (EX9.RenderState)



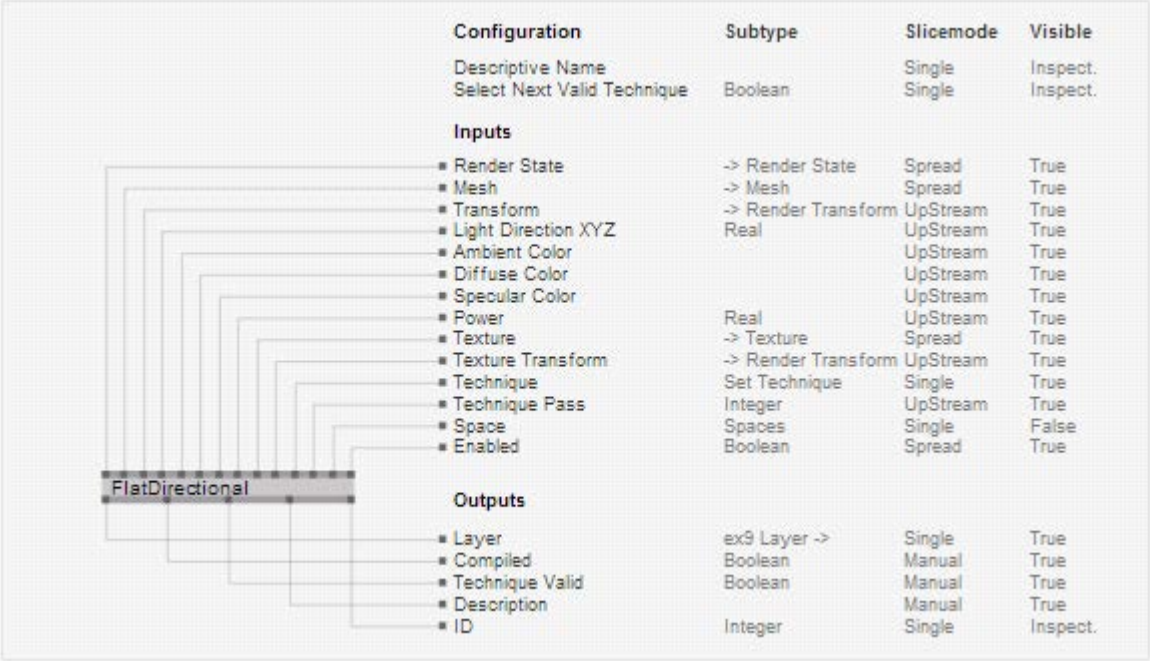
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Render State In	-> Render State	Spread	True
Fill Mode	FillMode	Spread	True
Outputs			
Render State Out	Render State ->	Spread	True
ID	Integer	Single	Inspect.

Flash (EX9)



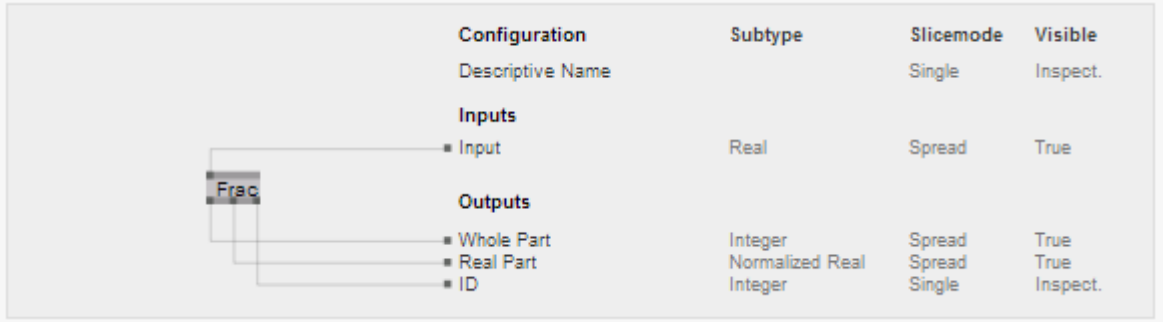
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Render State	-> Render State	Single	True
Transform	-> Render Transform	Single	True
Filename		Single	True
Load	Boolean	Single	True
Mouse X	Real	Single	True
Mouse Y	Real	Single	True
Mouse Left Button	Boolean	Single	True
Key Code	Integer	UpStream	True
Buffer Mode	VVVV.Nodes.DrawFlashBufferMode	Single	True
Quality	VVVV.Nodes.DrawFlashQuality	Single	True
Seek Frame	Integer	Single	True
Enabled	Boolean	Single	True
Outputs			
Layer	ex9 Layer ->	Single	True
Window Handle	Integer	Single	False
Frame Rate	Integer	Single	True
ID	Integer	Single	Inspect.

FlatDirectional (EX9.Effect)



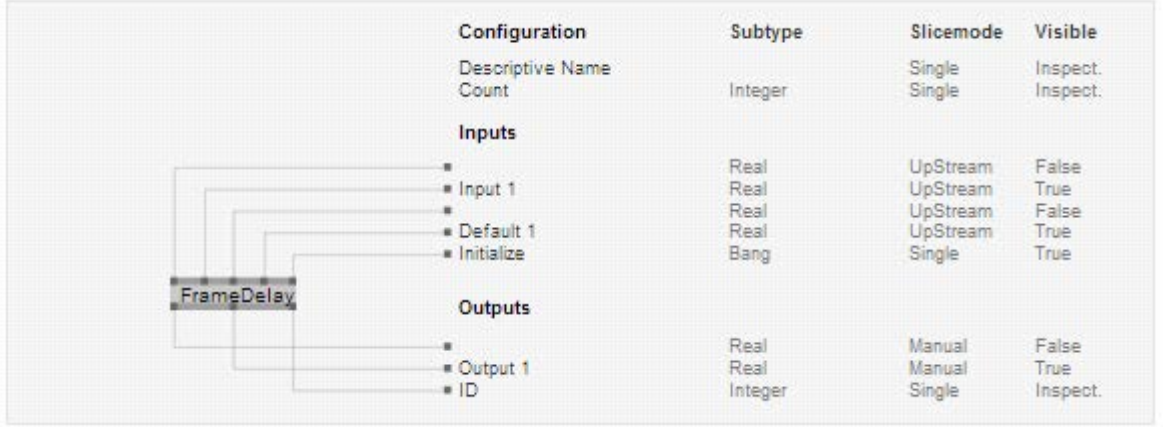
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Select Next Valid Technique	Boolean	Single	Inspect.
Inputs			
Render State	-> Render State	Spread	True
Mesh	-> Mesh	Spread	True
Transform	-> Render Transform	UpStream	True
Light Direction XYZ	Real	UpStream	True
Ambient Color		UpStream	True
Diffuse Color		UpStream	True
Specular Color		UpStream	True
Power	Real	UpStream	True
Texture	-> Texture	Spread	True
Texture Transform	-> Render Transform	UpStream	True
Technique	Set Technique	Single	True
Technique Pass	Integer	UpStream	True
Space	Spaces	Single	False
Enabled	Boolean	Spread	True
Outputs			
Layer	ex9 Layer ->	Single	True
Compiled	Boolean	Manual	True
Technique Valid	Boolean	Manual	True
Description		Manual	True
ID	Integer	Single	Inspect.

Frac (Value)



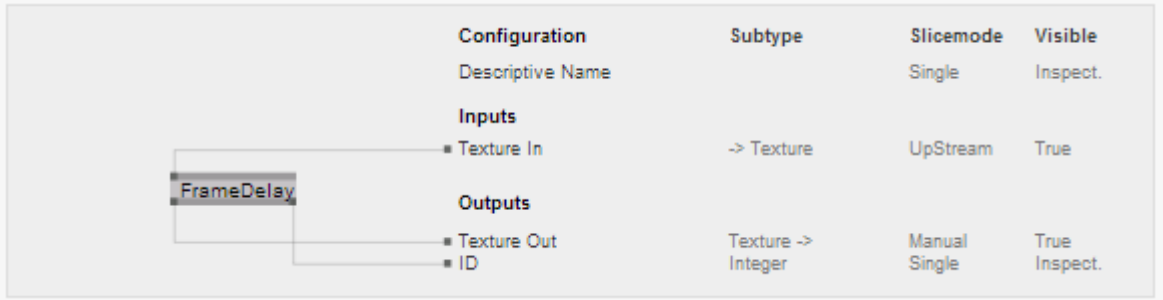
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Input	Real	Spread	True
Outputs			
Whole Part	Integer	Spread	True
Real Part	Normalized Real	Spread	True
ID	Integer	Single	Inspect.

FrameDelay (Animation)



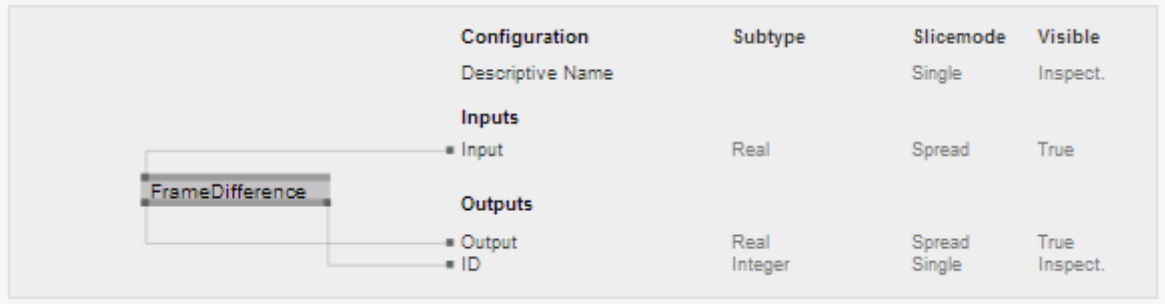
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Count	Integer	Single	Inspect.
Inputs			
	Real	UpStream	False
Input 1	Real	UpStream	True
	Real	UpStream	False
Default 1	Real	UpStream	True
Initialize	Bang	Single	True
Outputs			
	Real	Manual	False
Output 1	Real	Manual	True
ID	Integer	Single	Inspect.

FrameDelay (EX9.Texture)

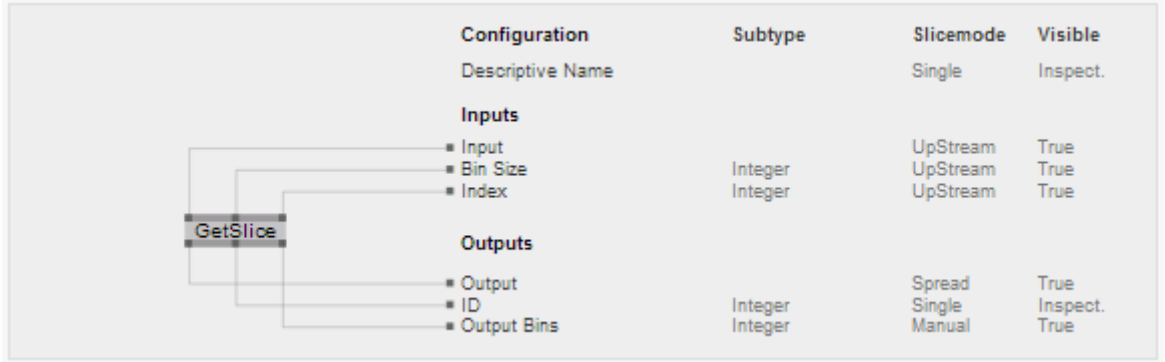


Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Texture In	-> Texture	UpStream	True
Outputs			
Texture Out	Texture ->	Manual	True
ID	Integer	Single	Inspect.

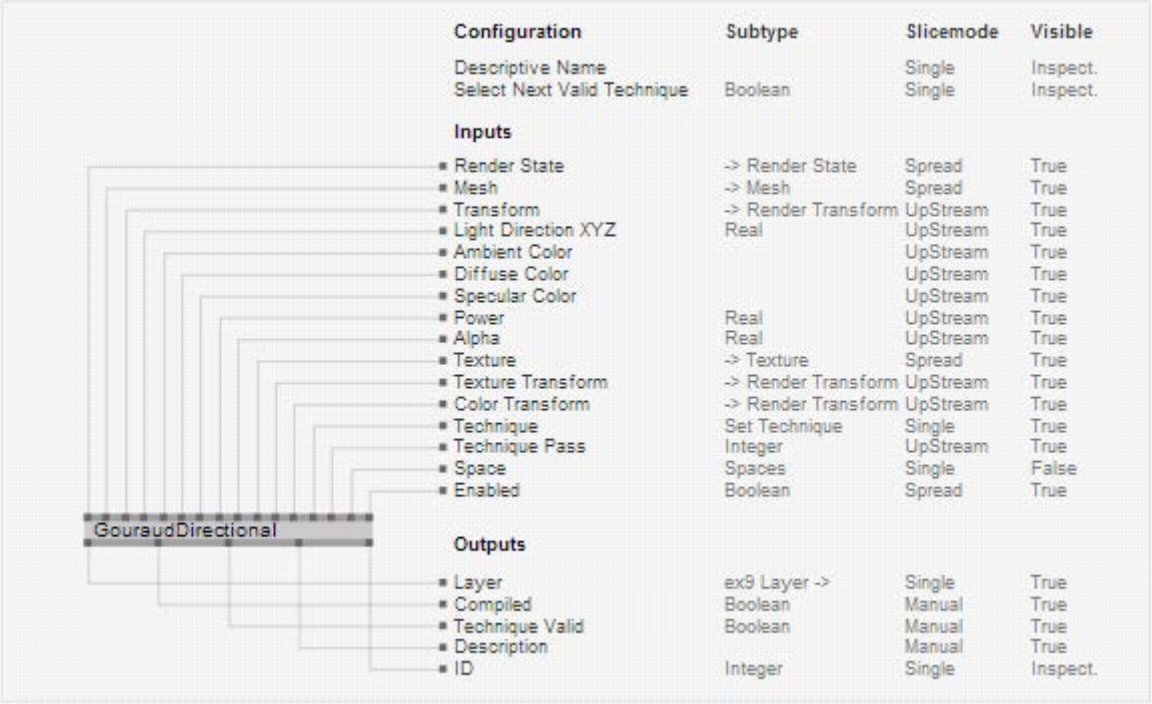
FrameDifference (Animation)



GetSlice (String)

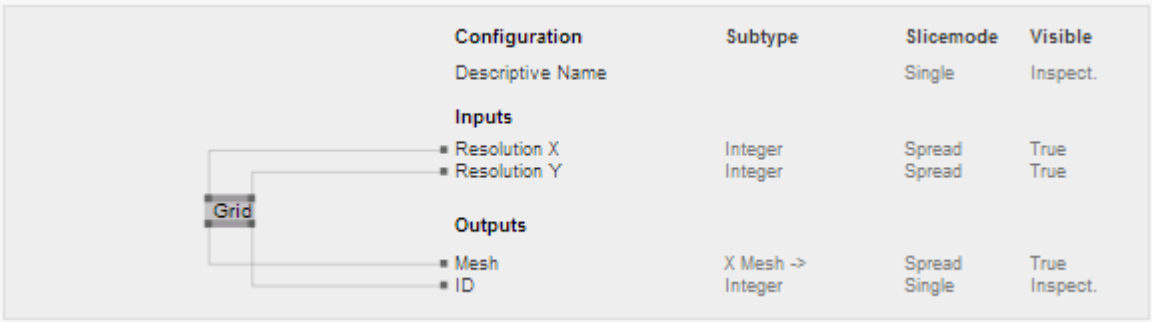


GouraudDirectional (EX9.Effect) basic vertex based lightning with directional light



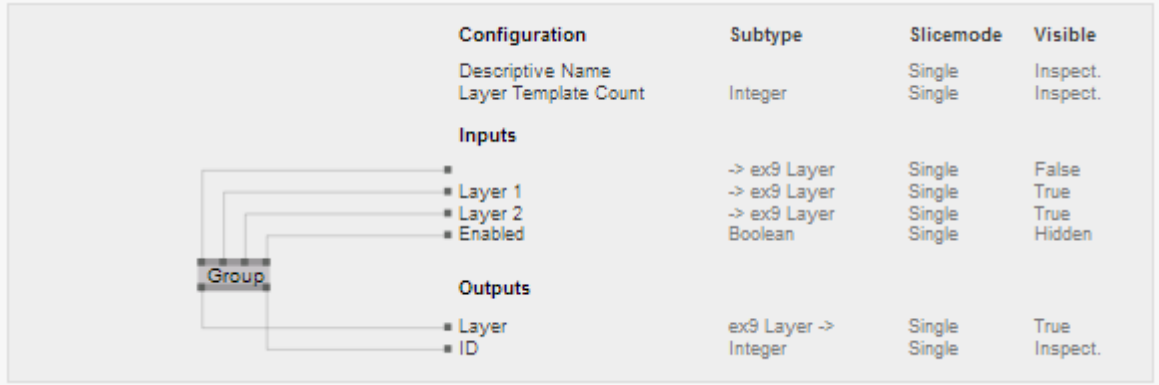
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Select Next Valid Technique	Boolean	Single	Inspect.
Inputs			
Render State	-> Render State	Spread	True
Mesh	-> Mesh	Spread	True
Transform	-> Render Transform	UpStream	True
Light Direction XYZ	Real	UpStream	True
Ambient Color		UpStream	True
Diffuse Color		UpStream	True
Specular Color		UpStream	True
Power	Real	UpStream	True
Alpha	Real	UpStream	True
Texture	-> Texture	Spread	True
Texture Transform	-> Render Transform	UpStream	True
Color Transform	-> Render Transform	UpStream	True
Technique	Set Technique	Single	True
Technique Pass	Integer	UpStream	True
Space	Spaces	Single	False
Enabled	Boolean	Spread	True
Outputs			
Layer	ex9 Layer ->	Single	True
Compiled	Boolean	Manual	True
Technique Valid	Boolean	Manual	True
Description		Manual	True
ID	Integer	Single	Inspect.

Grid (EX9.Geometry)



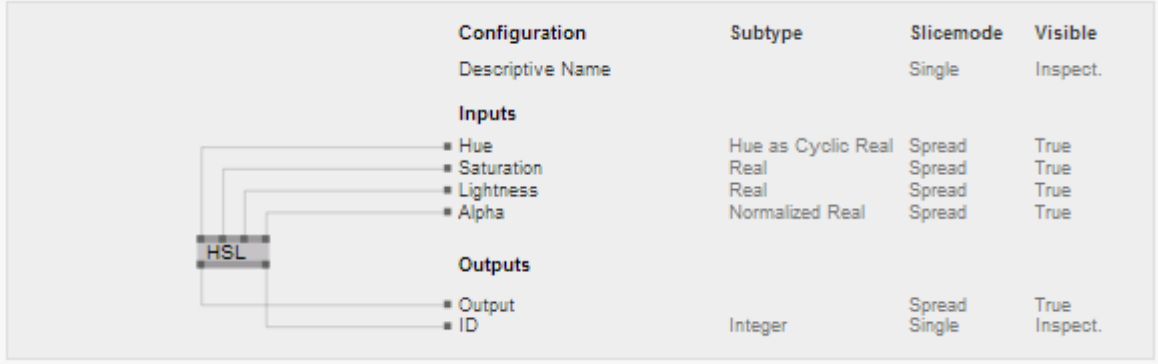
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Resolution X	Integer	Spread	True
Resolution Y	Integer	Spread	True
Outputs			
Mesh	X Mesh ->	Spread	True
ID	Integer	Single	Inspect.

Group (EX9)



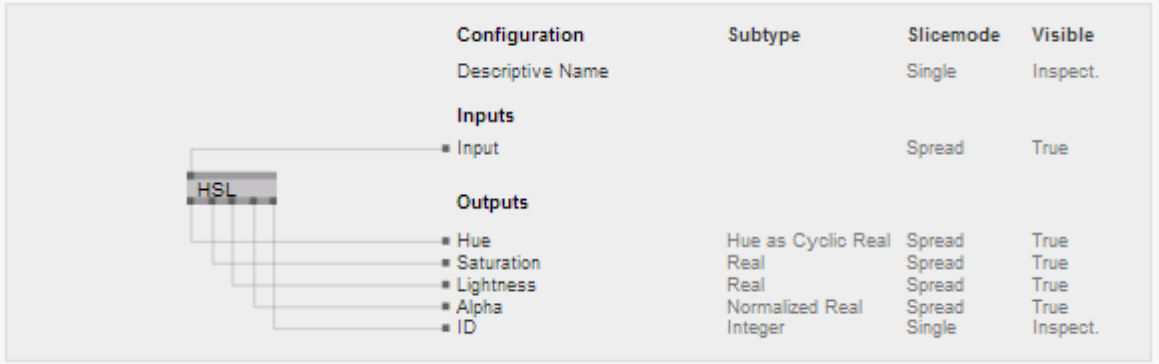
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Layer Template Count	Integer	Single	Inspect.
Inputs			
	-> ex9 Layer	Single	False
Layer 1	-> ex9 Layer	Single	True
Layer 2	-> ex9 Layer	Single	True
Enabled	Boolean	Single	Hidden
Outputs			
Layer	ex9 Layer ->	Single	True
ID	Integer	Single	Inspect.

HSL (Color Join)



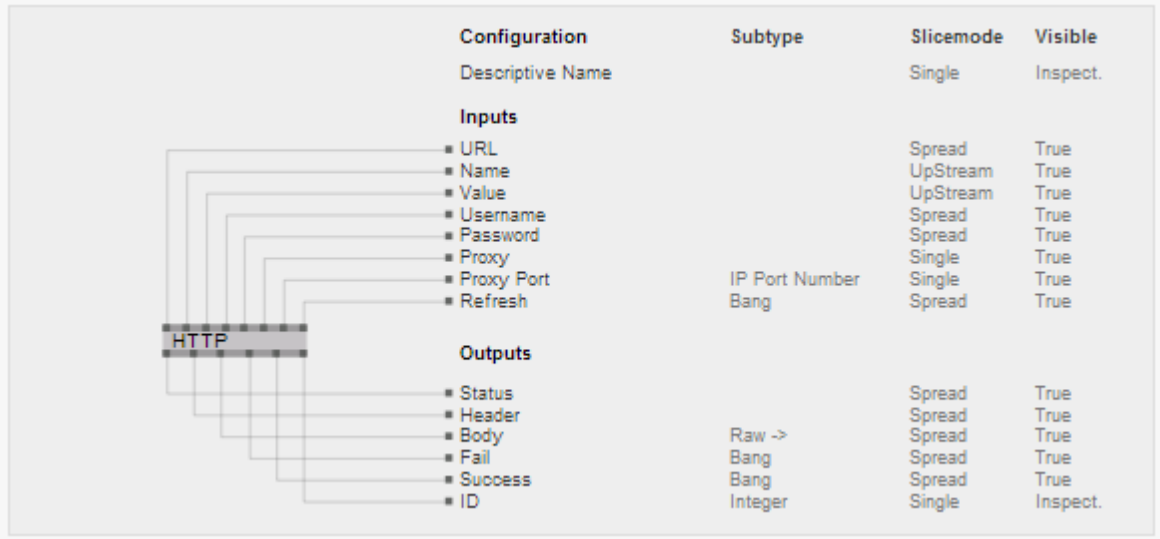
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Hue	Hue as Cyclic Real	Spread	True
Saturation	Real	Spread	True
Lightness	Real	Spread	True
Alpha	Normalized Real	Spread	True
Outputs			
Output		Spread	True
ID	Integer	Single	Inspect.

HSL (Color Split)

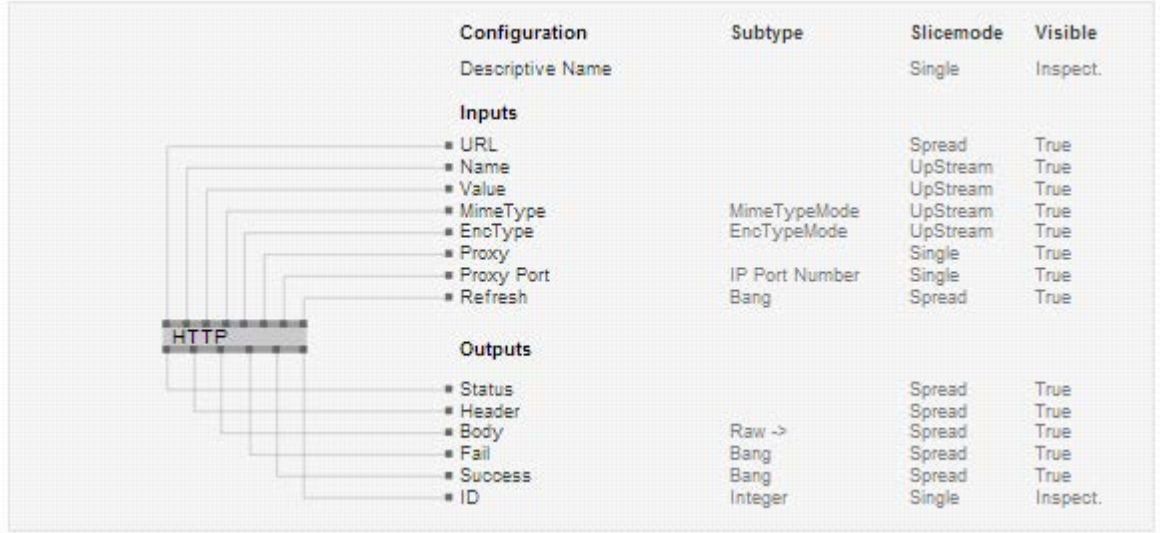


Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Input		Spread	True
Outputs			
Hue	Hue as Cyclic Real	Spread	True
Saturation	Real	Spread	True
Lightness	Real	Spread	True
Alpha	Normalized Real	Spread	True
ID	Integer	Single	Inspect.

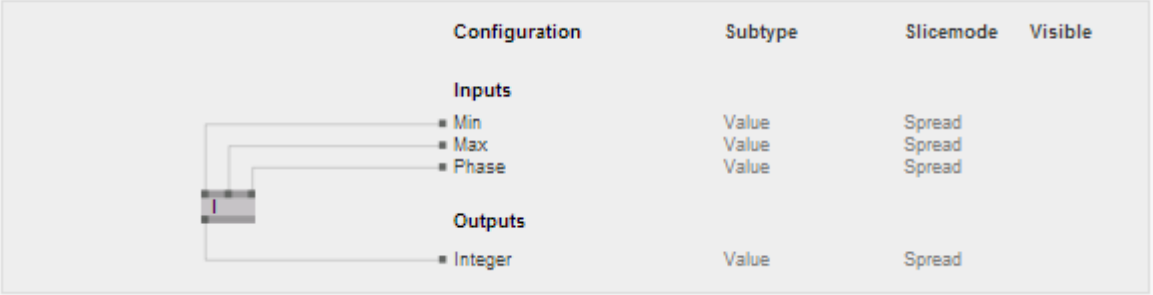
HTTP (Network Get)



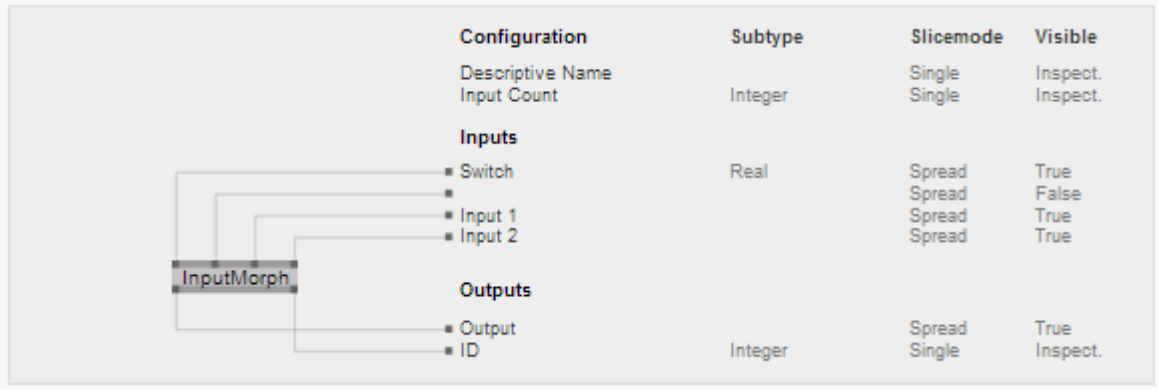
HTTP (Network Post)



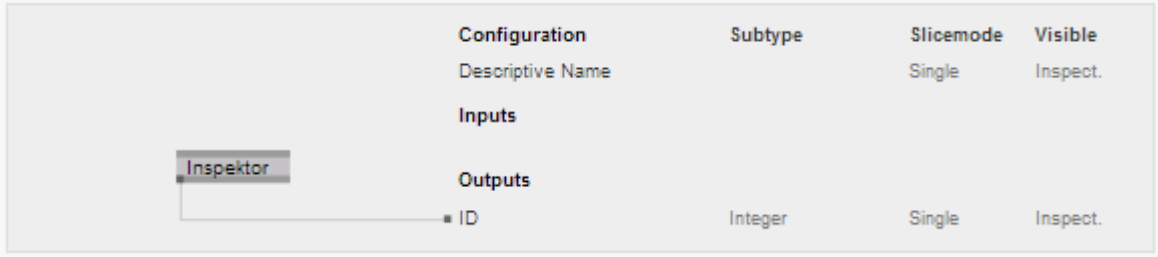
I (Spreads)



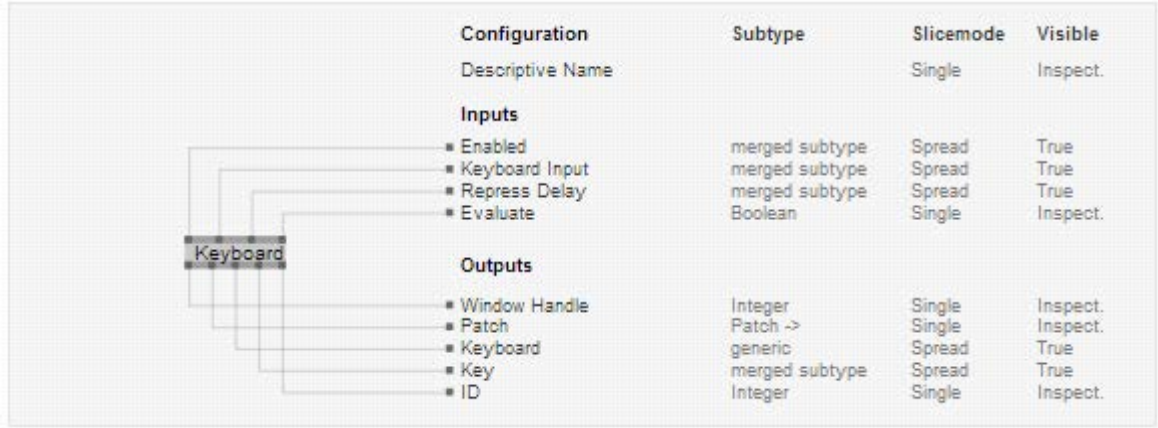
InputMorph (Value)



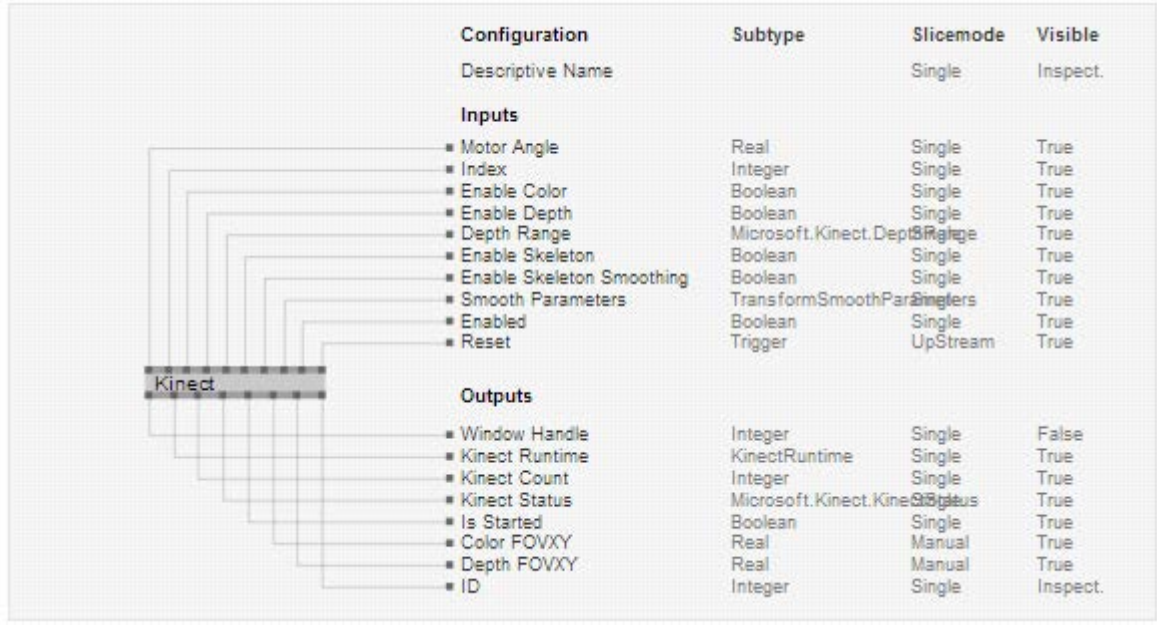
Inspektor (VVVV)



Keyboard (System Global)

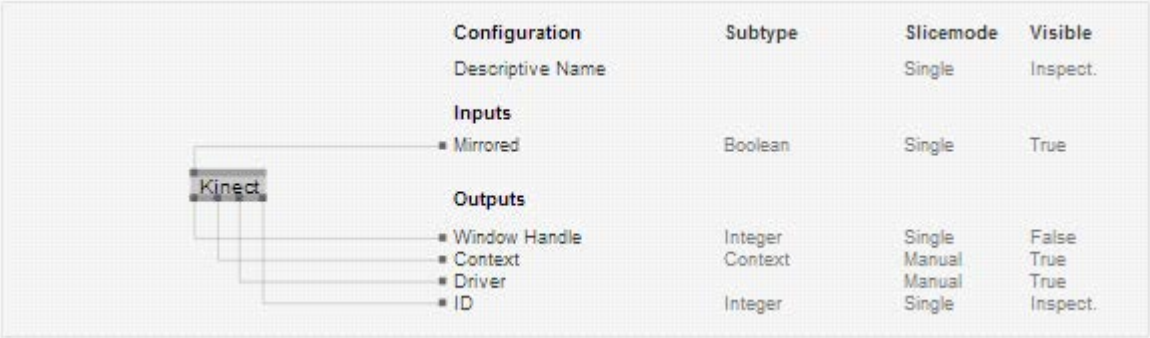


Kinect (Devices Microsoft)



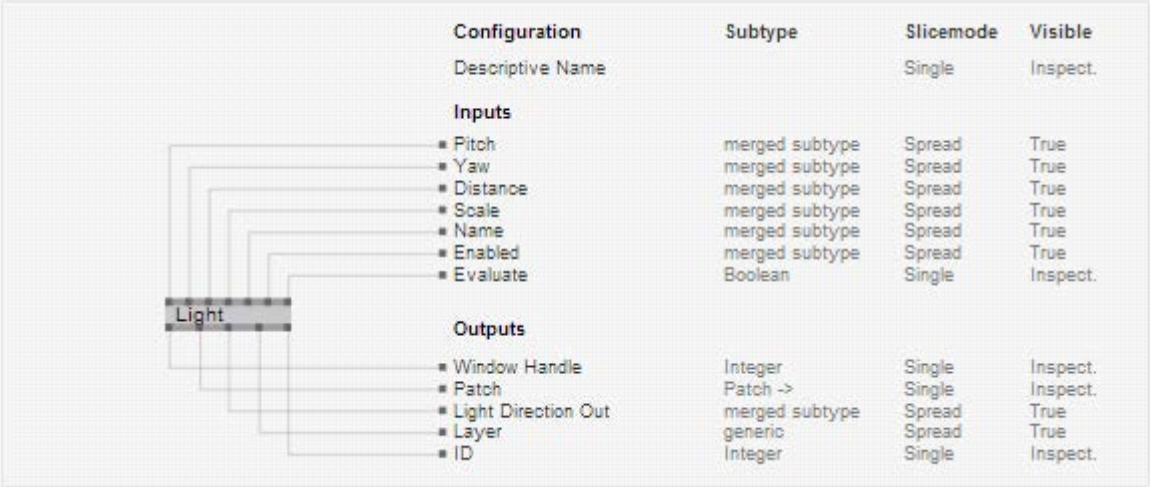
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Motor Angle	Real	Single	True
Index	Integer	Single	True
Enable Color	Boolean	Single	True
Enable Depth	Boolean	Single	True
Depth Range	Microsoft.Kinect.DepthRange	Single	True
Enable Skeleton	Boolean	Single	True
Enable Skeleton Smoothing	Boolean	Single	True
Smooth Parameters	TransformSmoothParameters	Single	True
Enabled	Boolean	Single	True
Reset	Trigger	UpStream	True
Outputs			
Window Handle	Integer	Single	False
Kinect Runtime	KinectRuntime	Single	True
Kinect Count	Integer	Single	True
Kinect Status	Microsoft.Kinect.KinectStatus	Single	True
Is Started	Boolean	Single	True
Color FOVXY	Real	Manual	True
Depth FOVXY	Real	Manual	True
ID	Integer	Single	Inspect.

Kinect (Devices OpenNI)



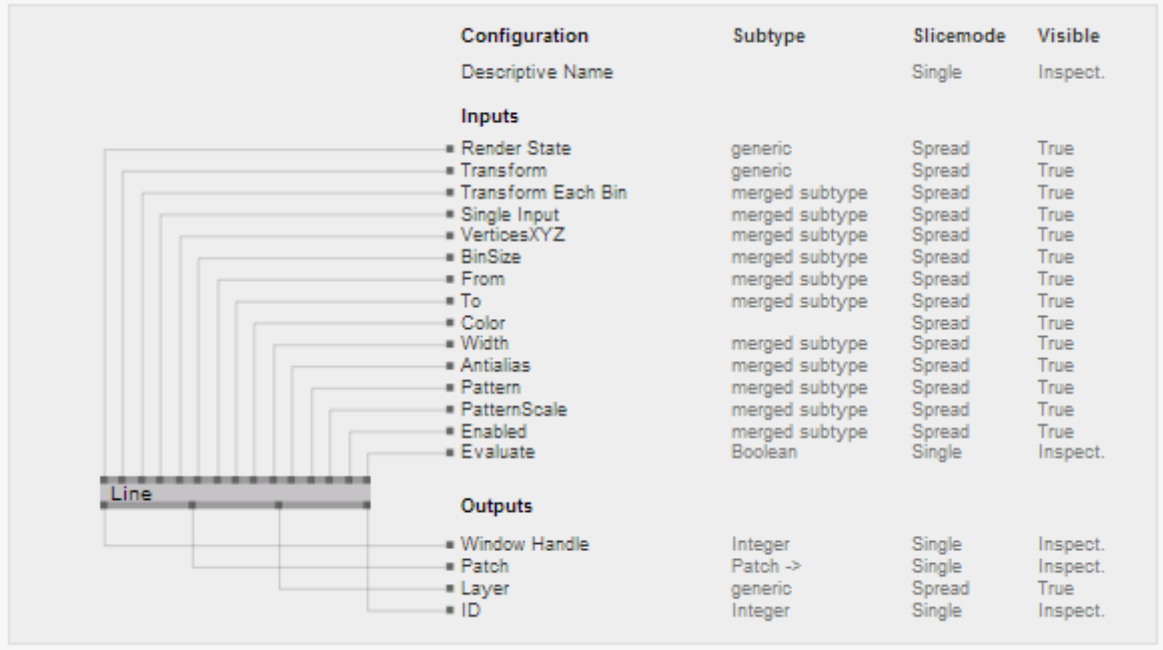
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Mirrored	Boolean	Single	True
Outputs			
Window Handle	Integer	Single	False
Context	Context	Manual	True
Driver		Manual	True
ID	Integer	Single	Inspect.

Light (EX9 Direction)



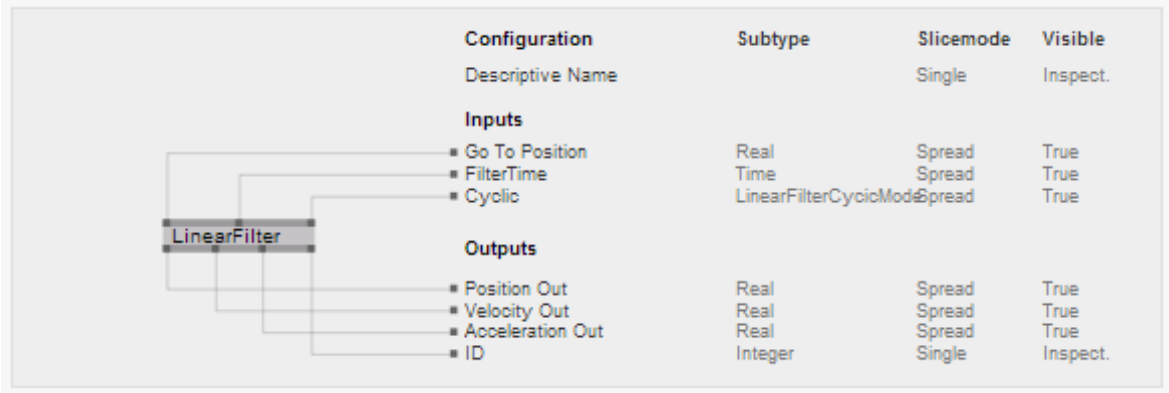
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Pitch	merged subtype	Spread	True
Yaw	merged subtype	Spread	True
Distance	merged subtype	Spread	True
Scale	merged subtype	Spread	True
Name	merged subtype	Spread	True
Enabled	merged subtype	Spread	True
Evaluate	Boolean	Single	Inspect.
Outputs			
Window Handle	Integer	Single	Inspect.
Patch	Patch ->	Single	Inspect.
Light Direction Out	merged subtype	Spread	True
Layer	generic	Spread	True
ID	Integer	Single	Inspect.

Line (EX9)



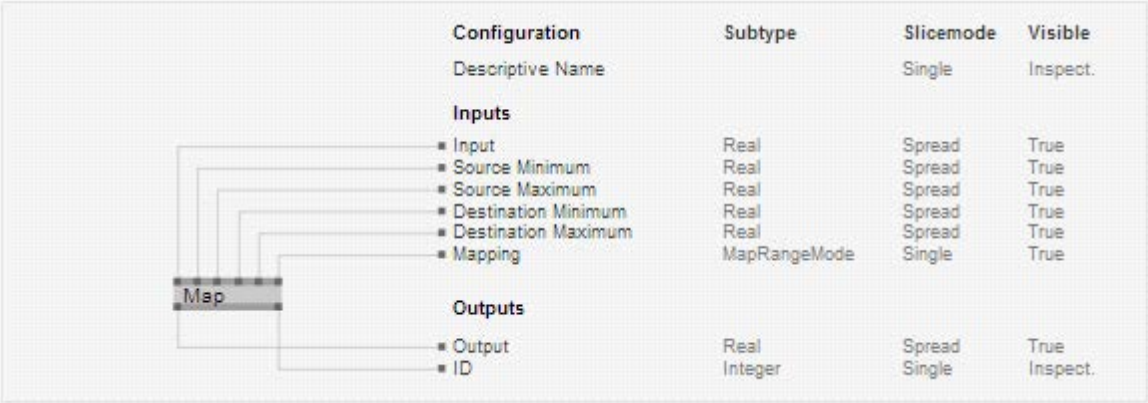
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Render State	generic	Spread	True
Transform	generic	Spread	True
Transform Each Bin	merged subtype	Spread	True
Single Input	merged subtype	Spread	True
VerticesXYZ	merged subtype	Spread	True
BinSize	merged subtype	Spread	True
From	merged subtype	Spread	True
To	merged subtype	Spread	True
Color		Spread	True
Width	merged subtype	Spread	True
Antialias	merged subtype	Spread	True
Pattern	merged subtype	Spread	True
PatternScale	merged subtype	Spread	True
Enabled	merged subtype	Spread	True
Evaluate	Boolean	Single	Inspect.
Outputs			
Window Handle	Integer	Single	Inspect.
Patch	Patch ->	Single	Inspect.
Layer	generic	Spread	True
ID	Integer	Single	Inspect.

LinearFilter (Animation)



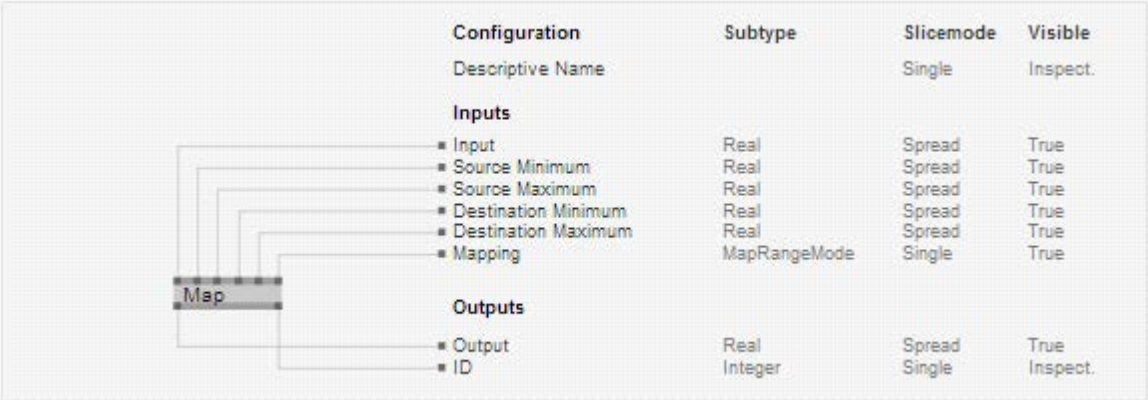
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Go To Position	Real	Spread	True
FilterTime	Time	Spread	True
Cyclic	LinearFilterCycicMode	Spread	True
Outputs			
Position Out	Real	Spread	True
Velocity Out	Real	Spread	True
Acceleration Out	Real	Spread	True
ID	Integer	Single	Inspect.

Map (Value)



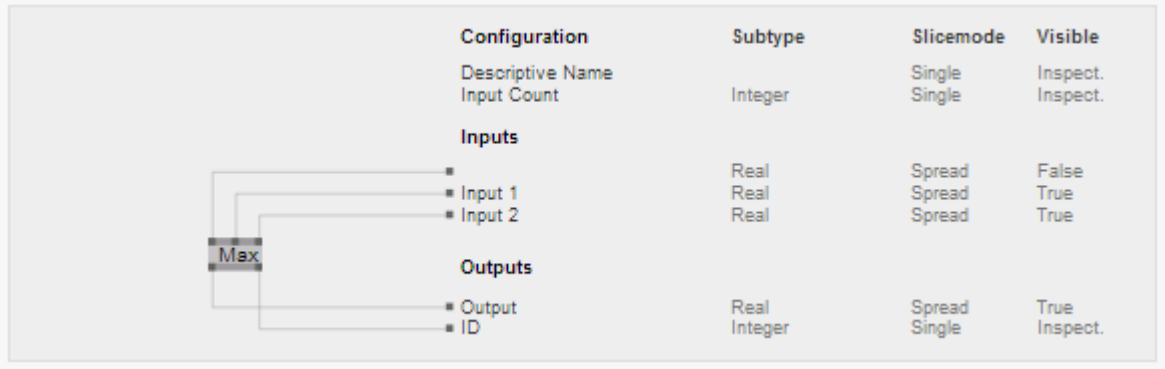
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Input	Real	Spread	True
Source Minimum	Real	Spread	True
Source Maximum	Real	Spread	True
Destination Minimum	Real	Spread	True
Destination Maximum	Real	Spread	True
Mapping	MapRangeMode	Single	True
Outputs			
Output	Real	Spread	True
ID	Integer	Single	Inspect.

Max (Value)



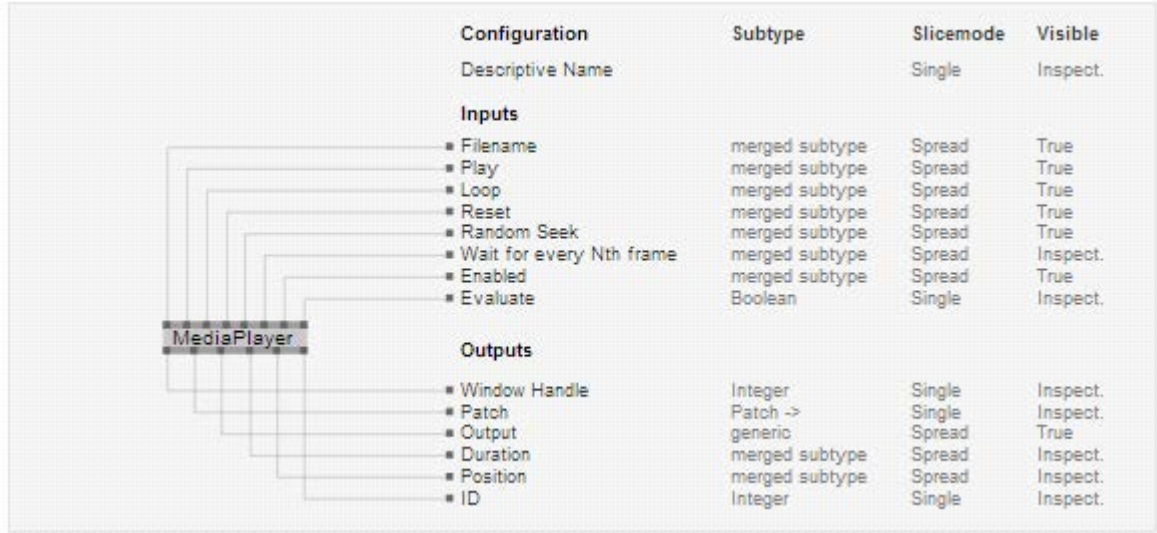
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Input	Real	Spread	True
Source Minimum	Real	Spread	True
Source Maximum	Real	Spread	True
Destination Minimum	Real	Spread	True
Destination Maximum	Real	Spread	True
Mapping	MapRangeMode	Single	True
Outputs			
Output	Real	Spread	True
ID	Integer	Single	Inspect.

MediaPlayer (EX9.Texture Source)



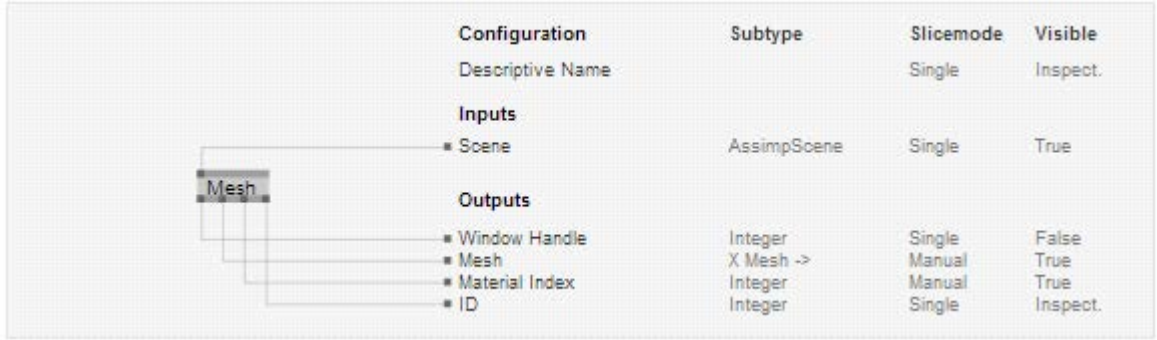
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Input Count	Integer	Single	Inspect.
Inputs			
	Real	Spread	False
Input 1	Real	Spread	True
Input 2	Real	Spread	True
Outputs			
Output	Real	Spread	True
ID	Integer	Single	Inspect.

Mesh (EX9.Geometry Assimp)



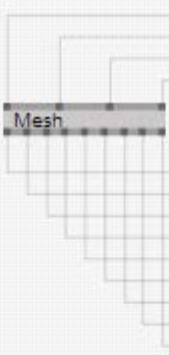
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Filename	merged subtype	Spread	True
Play	merged subtype	Spread	True
Loop	merged subtype	Spread	True
Reset	merged subtype	Spread	True
Random Seek	merged subtype	Spread	True
Wait for every Nth frame	merged subtype	Spread	Inspect.
Enabled	merged subtype	Spread	True
Evaluate	Boolean	Single	Inspect.
Outputs			
Window Handle	Integer	Single	Inspect.
Patch	Patch ->	Single	Inspect.
Output	generic	Spread	True
Duration	merged subtype	Spread	Inspect.
Position	merged subtype	Spread	Inspect.
ID	Integer	Single	Inspect.

Mesh (EX9.Geometry Collada)



Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Scene	AssimpScene	Single	True
Outputs			
Window Handle	Integer	Single	False
Mesh	X Mesh ->	Manual	True
Material Index	Integer	Manual	True
ID	Integer	Single	Inspect.

Mesh (EX9.Geometry Join)



Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Opaque=1?	Boolean	Single	True
Inputs			
COLLADA Model	Model	UpStream	True
Time	Real	UpStream	True
Bin Size	Integer	Single	True
Index	Integer	UpStream	True
Outputs			
Mesh	X Mesh ->	Manual	True
Window Handle	Integer	Single	False
TextureFileName		Manual	True
Emissive Color		Manual	True
Diffuse Color		Manual	True
Specular Color		Manual	True
Power	Real	Manual	True
Opaque	Real	Manual	True
ID	Integer	Single	Inspect.

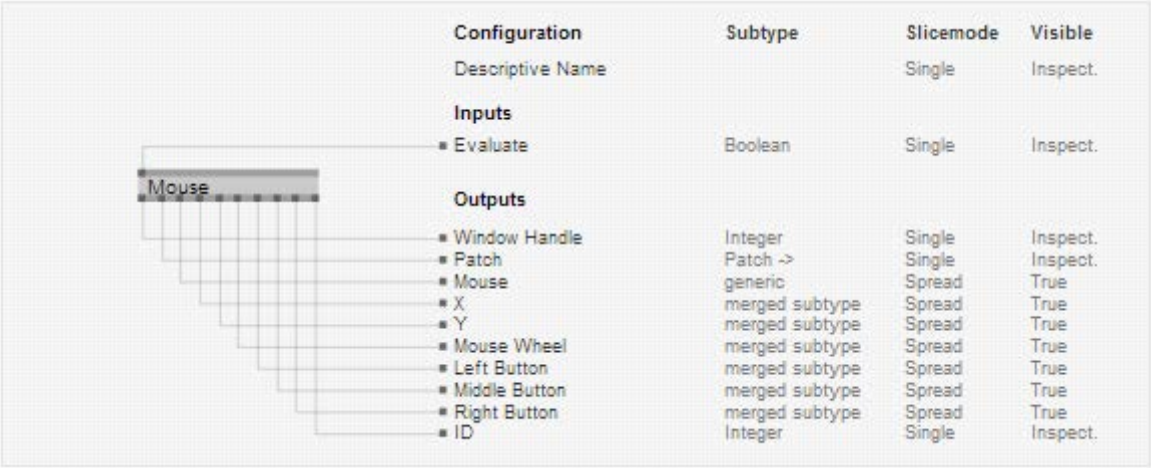
MidiNote (Devices)

Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Channel	Integer	UpStream	True
Note	Integer	UpStream	True
Buffer Length	Integer	Single	True
Enabled	Boolean	Single	True
Midi Input Port	MidiInputPorts	Single	True
Outputs			
Output	Normalized Real	Spread	True
On Data	Bang	Spread	True
Buffer Overflow	Boolean	Spread	True
ID	Integer	Single	Inspect.

Min (Value)

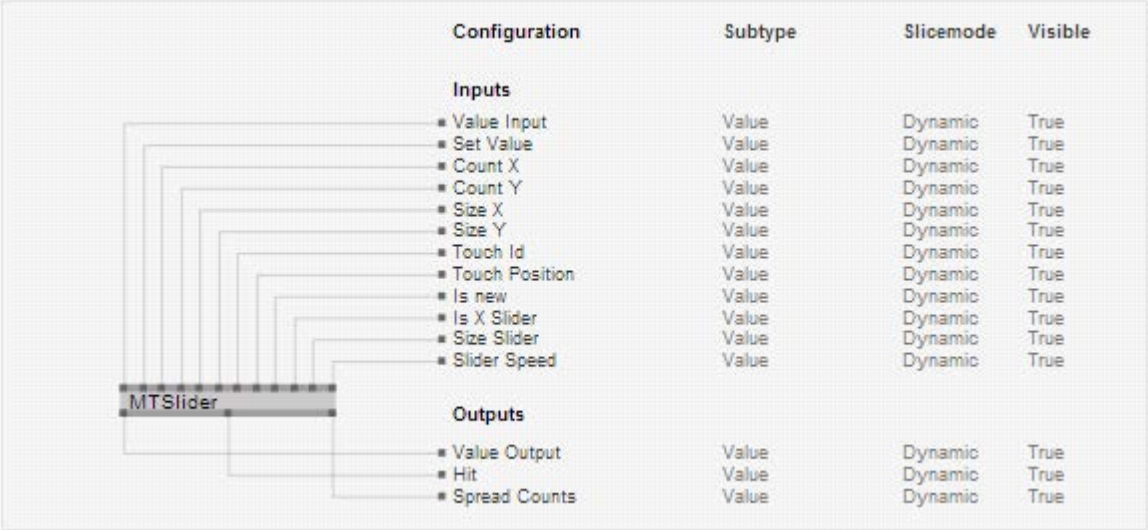
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Input Count	Integer	Single	Inspect.
Inputs			
	Real	Spread	False
Input 1	Real	Spread	True
Input 2	Real	Spread	True
Outputs			
Output	Real	Spread	True
ID	Integer	Single	Inspect.

Mouse (System Window)



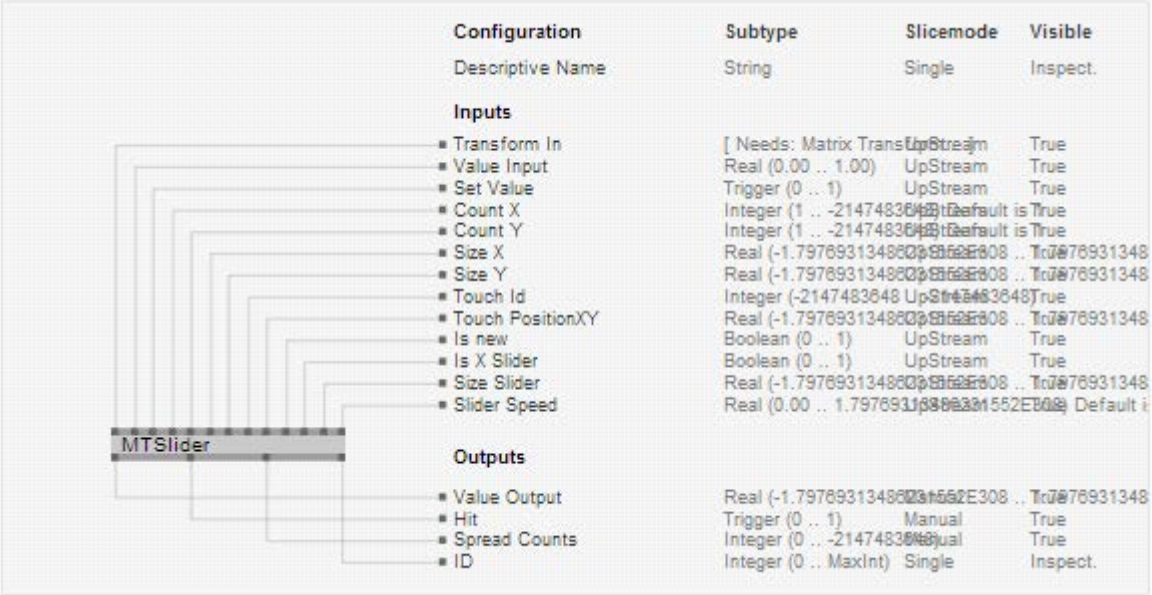
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Evaluate	Boolean	Single	Inspect.
Outputs			
Window Handle	Integer	Single	Inspect.
Patch	Patch ->	Single	Inspect.
Mouse	generic	Spread	True
X	merged subtype	Spread	True
Y	merged subtype	Spread	True
Mouse Wheel	merged subtype	Spread	True
Left Button	merged subtype	Spread	True
Middle Button	merged subtype	Spread	True
Right Button	merged subtype	Spread	True
ID	Integer	Single	Inspect.

MTSlider (2d GUI)

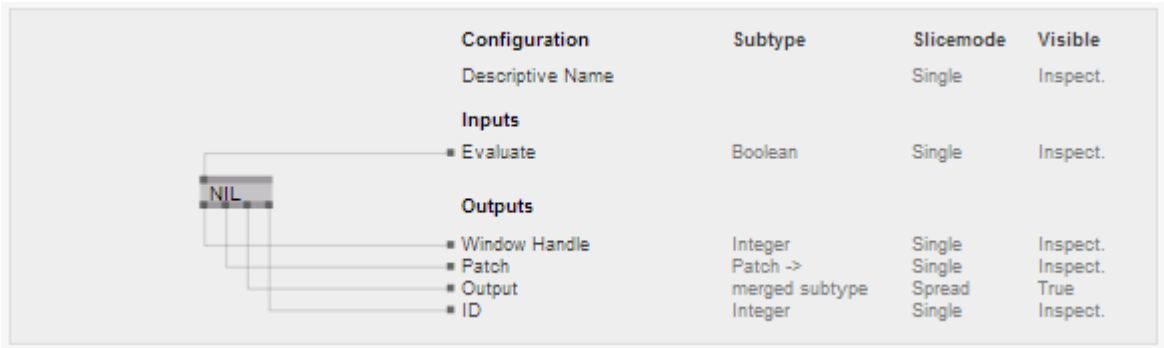


Configuration	Subtype	Slicemode	Visible
Inputs			
Value Input	Value	Dynamic	True
Set Value	Value	Dynamic	True
Count X	Value	Dynamic	True
Count Y	Value	Dynamic	True
Size X	Value	Dynamic	True
Size Y	Value	Dynamic	True
Touch Id	Value	Dynamic	True
Touch Position	Value	Dynamic	True
Is new	Value	Dynamic	True
Is X Slider	Value	Dynamic	True
Size Slider	Value	Dynamic	True
Slider Speed	Value	Dynamic	True
Outputs			
Value Output	Value	Dynamic	True
Hit	Value	Dynamic	True
Spread Counts	Value	Dynamic	True

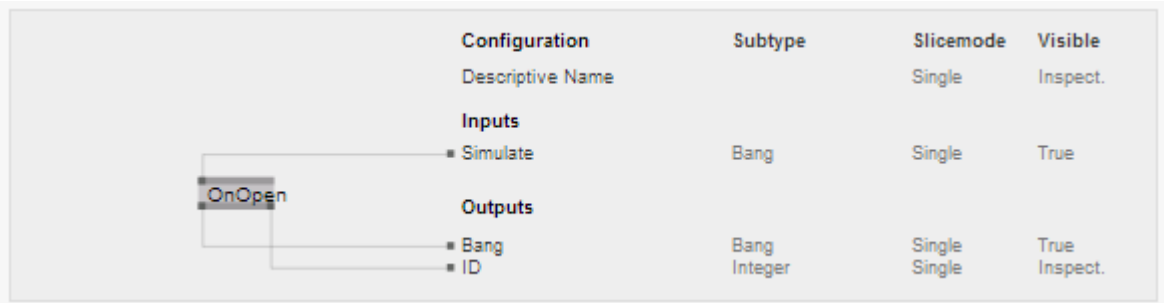
MTSlider (GUI Multitouch)



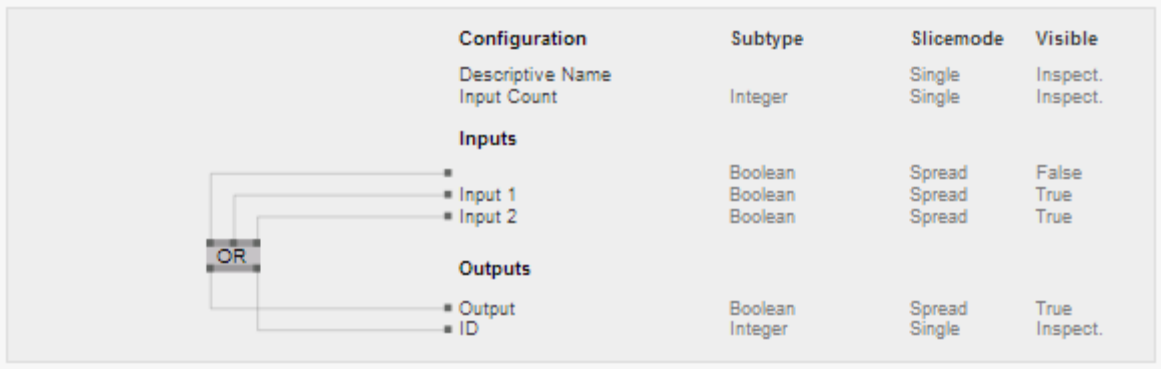
NIL (Spreads)



OnOpen (VVVV)

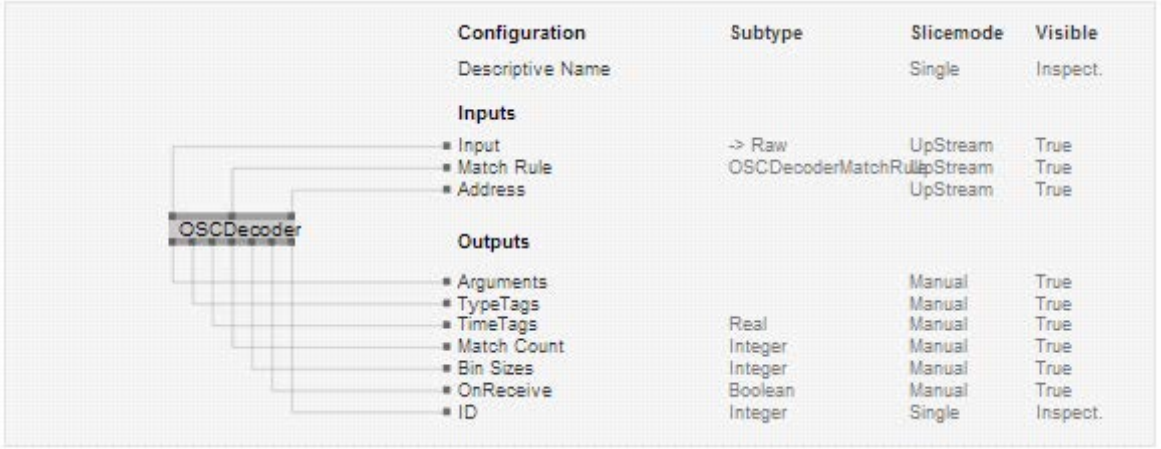


OR (Boolean)



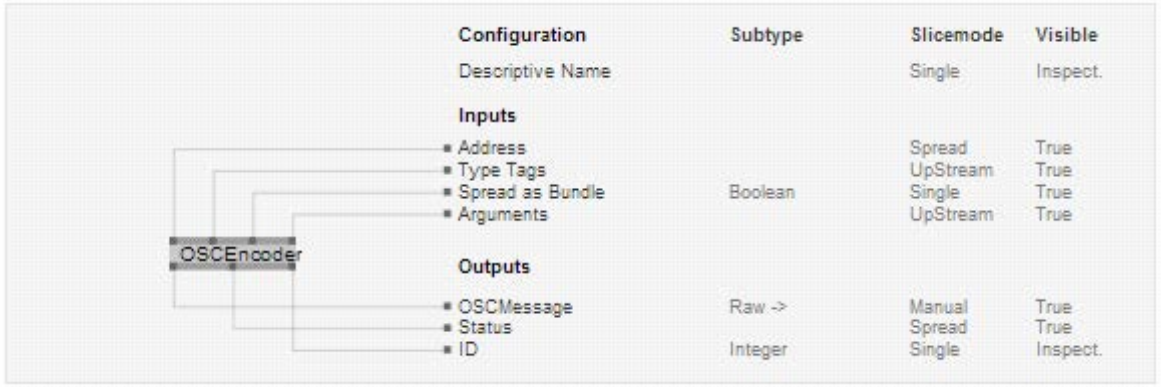
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Input Count	Integer	Single	Inspect.
Inputs			
	Boolean	Spread	False
Input 1	Boolean	Spread	True
Input 2	Boolean	Spread	True
Outputs			
Output	Boolean	Spread	True
ID	Integer	Single	Inspect.

OSCDecoder (Network)



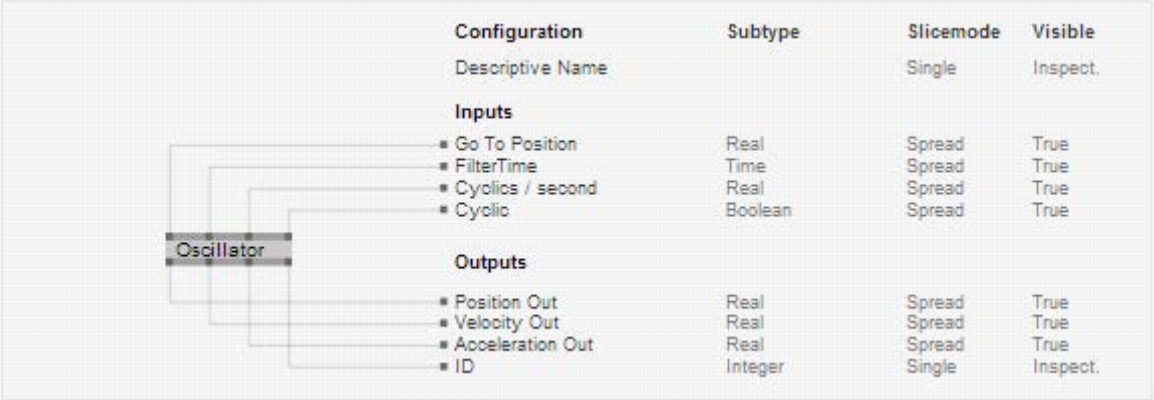
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Input	-> Raw	UpStream	True
Match Rule	OSCDecoderMatchRule	UpStream	True
Address		UpStream	True
Outputs			
Arguments		Manual	True
TypeTags		Manual	True
TimeTags	Real	Manual	True
Match Count	Integer	Manual	True
Bin Sizes	Integer	Manual	True
OnReceive	Boolean	Manual	True
ID	Integer	Single	Inspect.

OSCEncoder (Network)



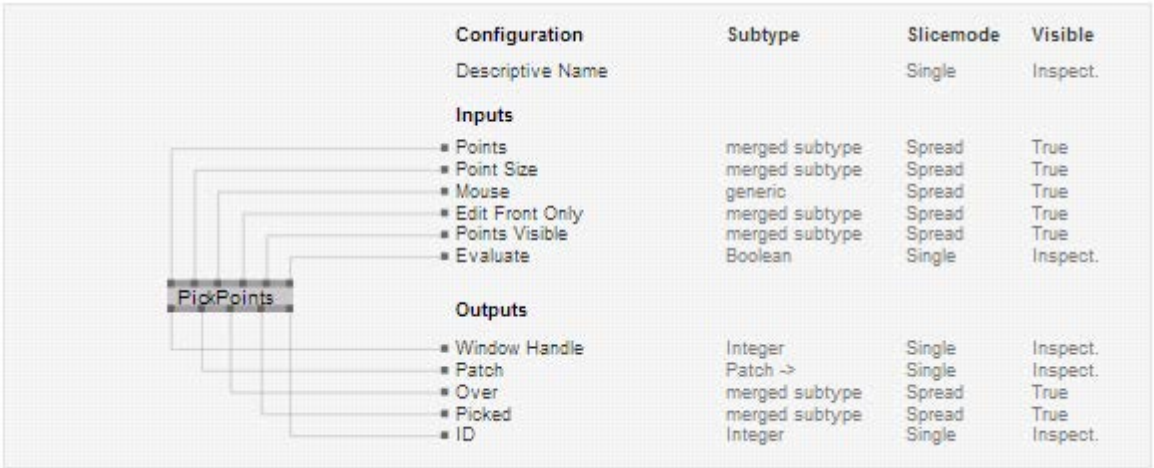
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Address		Spread	True
Type Tags		UpStream	True
Spread as Bundle	Boolean	Single	True
Arguments		UpStream	True
Outputs			
OSCMessage	Raw ->	Manual	True
Status		Spread	True
ID	Integer	Single	Inspect.

Oscillator (Animation)



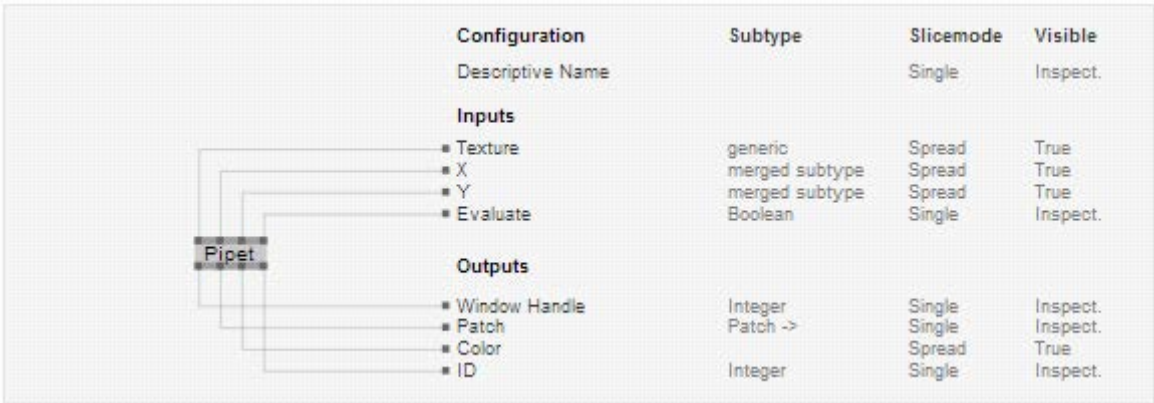
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Go To Position	Real	Spread	True
FilterTime	Time	Spread	True
Cyclics / second	Real	Spread	True
Cyclic	Boolean	Spread	True
Outputs			
Position Out	Real	Spread	True
Velocity Out	Real	Spread	True
Acceleration Out	Real	Spread	True
ID	Integer	Single	Inspect.

PickPoints (3D Vector)



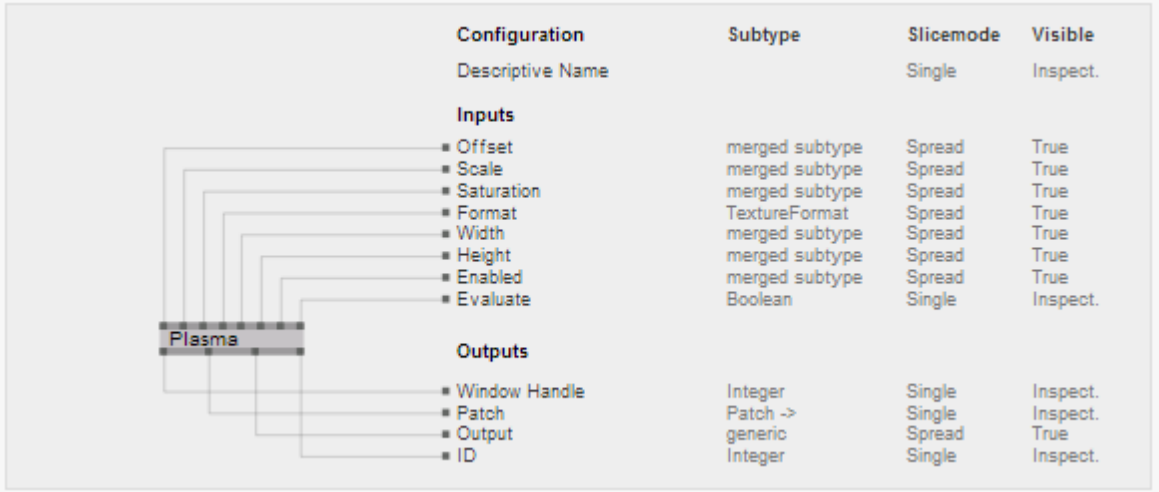
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Points	merged subtype	Spread	True
Point Size	merged subtype	Spread	True
Mouse	generic	Spread	True
Edit Front Only	merged subtype	Spread	True
Points Visible	merged subtype	Spread	True
Evaluate	Boolean	Single	Inspect.
Outputs			
Window Handle	Integer	Single	Inspect.
Patch	Patch ->	Single	Inspect.
Over	merged subtype	Spread	True
Picked	merged subtype	Spread	True
ID	Integer	Single	Inspect.

Pipet (EX9.Texture Simple)



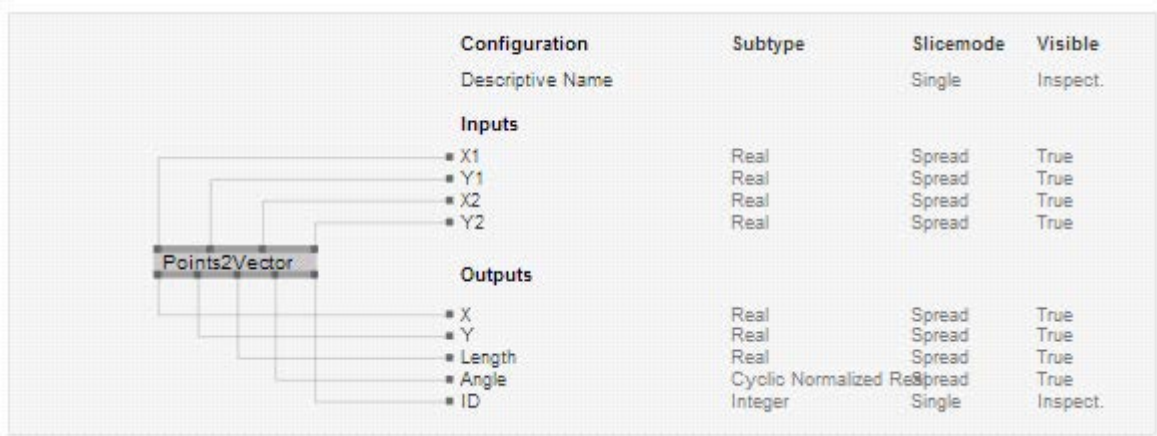
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Texture	generic	Spread	True
X	merged subtype	Spread	True
Y	merged subtype	Spread	True
Evaluate	Boolean	Single	Inspect.
Outputs			
Window Handle	Integer	Single	Inspect.
Patch	Patch ->	Single	Inspect.
Color		Spread	True
ID	Integer	Single	Inspect.

Plasma (EX9.Texture Source)



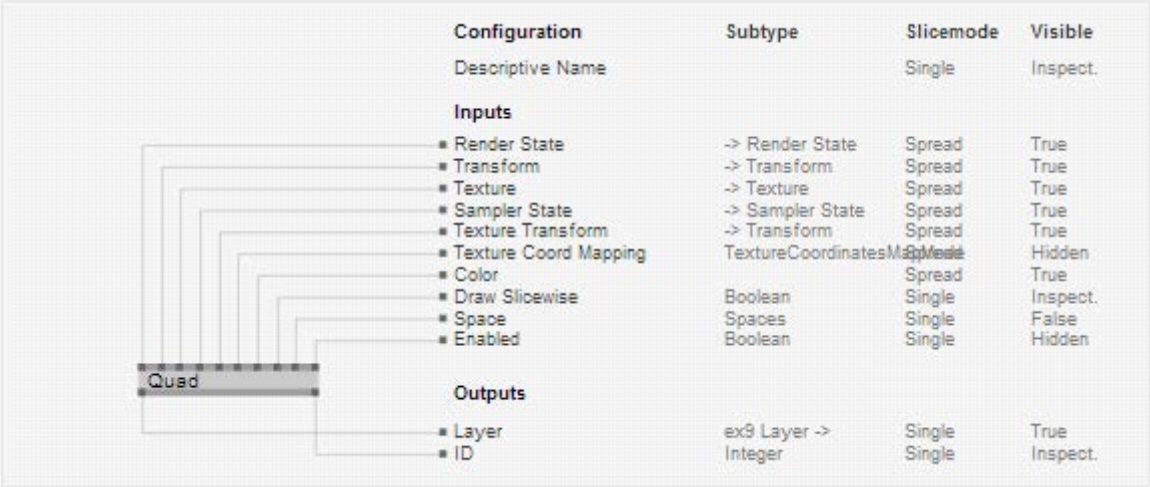
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Offset	merged subtype	Spread	True
Scale	merged subtype	Spread	True
Saturation	merged subtype	Spread	True
Format	TextureFormat	Spread	True
Width	merged subtype	Spread	True
Height	merged subtype	Spread	True
Enabled	merged subtype	Spread	True
Evaluate	Boolean	Single	Inspect.
Outputs			
Window Handle	Integer	Single	Inspect.
Patch	Patch ->	Single	Inspect.
Output	generic	Spread	True
ID	Integer	Single	Inspect.

Points2Vector (2d)



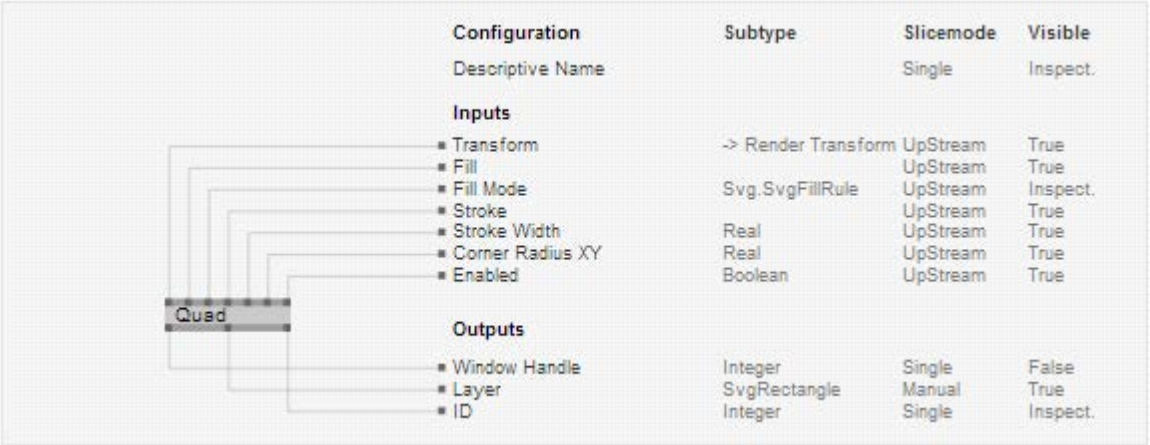
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
X1	Real	Spread	True
Y1	Real	Spread	True
X2	Real	Spread	True
Y2	Real	Spread	True
Outputs			
X	Real	Spread	True
Y	Real	Spread	True
Length	Real	Spread	True
Angle	Cyclic Normalized Re	Spread	True
ID	Integer	Single	Inspect.

Quad (DX9)



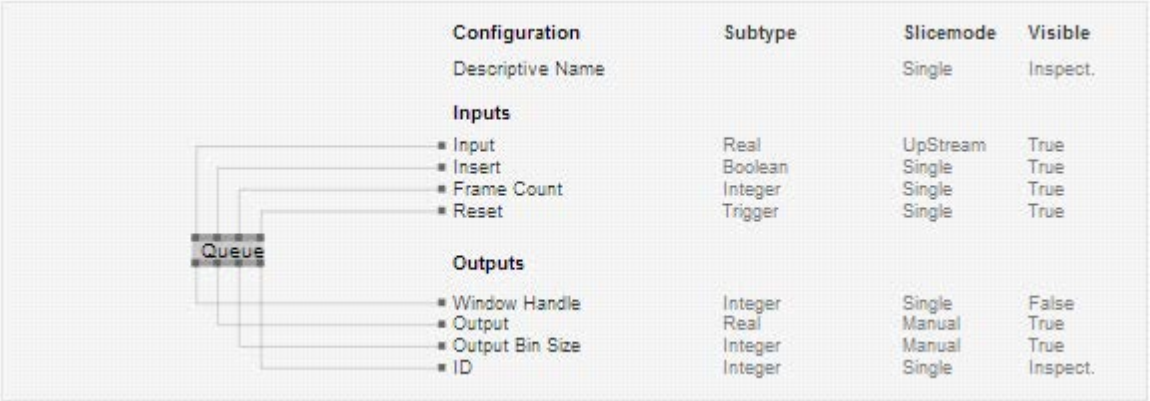
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Render State	-> Render State	Spread	True
Transform	-> Transform	Spread	True
Texture	-> Texture	Spread	True
Sampler State	-> Sampler State	Spread	True
Texture Transform	-> Transform	Spread	True
Texture Coord Mapping	TextureCoordinatesMapMode	Spread	Hidden
Color		Spread	True
Draw Slicewise	Boolean	Single	Inspect.
Space	Spaces	Single	False
Enabled	Boolean	Single	Hidden
Outputs			
Layer	ex9 Layer ->	Single	True
ID	Integer	Single	Inspect.

Quad (SVG)



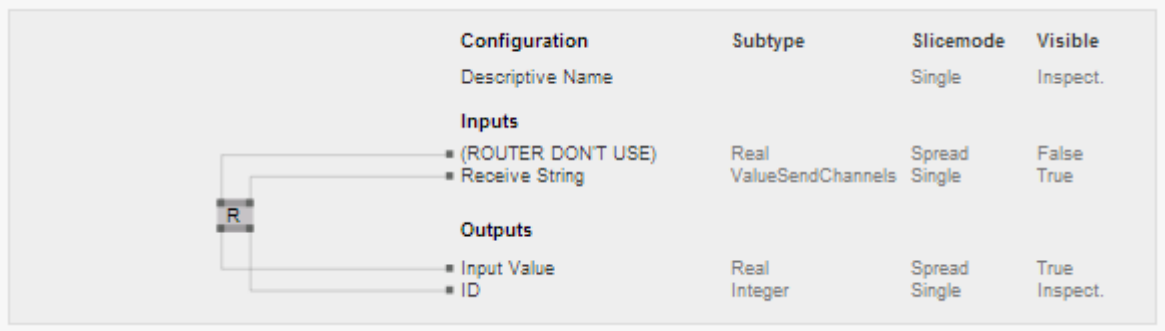
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Transform	-> Render Transform	UpStream	True
Fill		UpStream	True
Fill Mode	Svg.SvgFillRule	UpStream	Inspect.
Stroke		UpStream	True
Stroke Width	Real	UpStream	True
Corner Radius XY	Real	UpStream	True
Enabled	Boolean	UpStream	True
Outputs			
Window Handle	Integer	Single	False
Layer	SvgRectangle	Manual	True
ID	Integer	Single	Inspect.

Queue (Spreads)

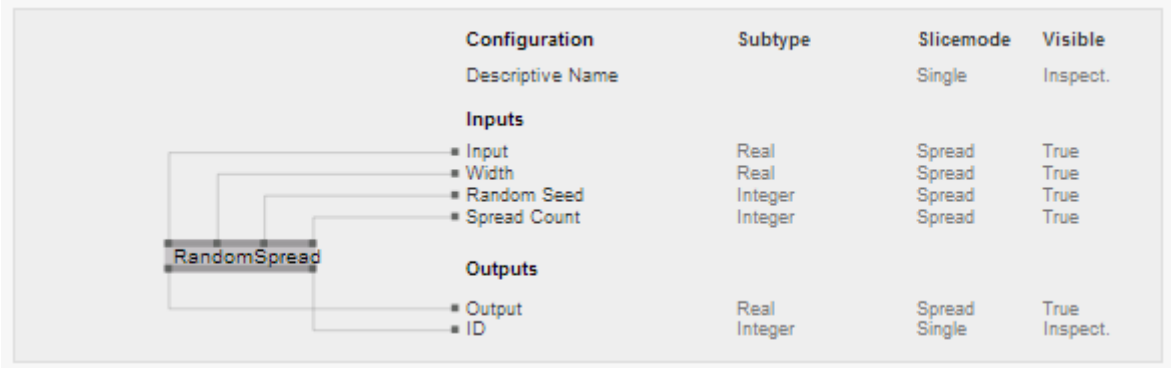


Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Input	Real	UpStream	True
Insert	Boolean	Single	True
Frame Count	Integer	Single	True
Reset	Trigger	Single	True
Outputs			
Window Handle	Integer	Single	False
Output	Real	Manual	True
Output Bin Size	Integer	Manual	True
ID	Integer	Single	Inspect.

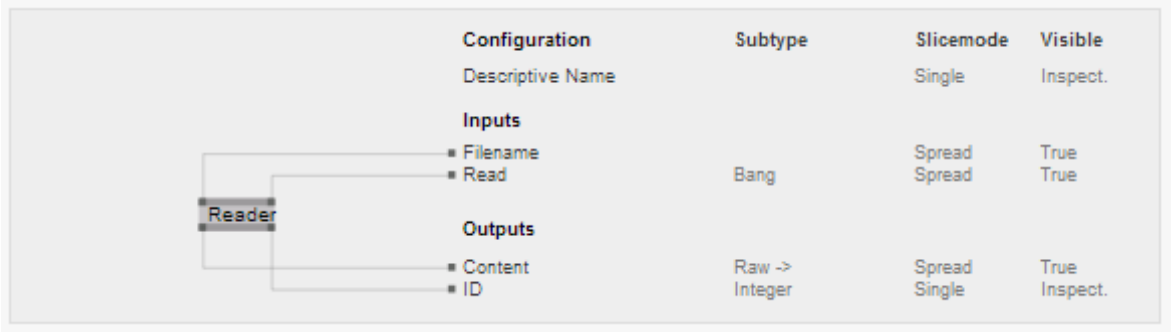
R (Value)



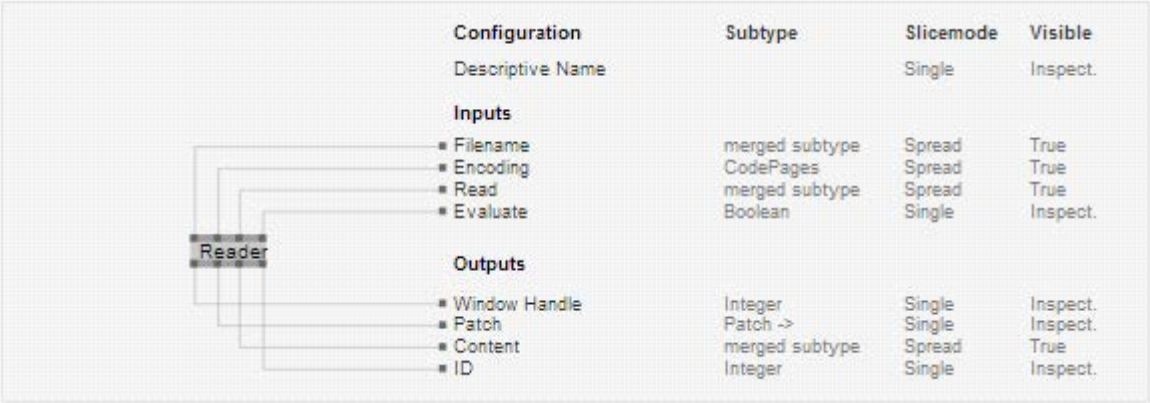
RandomSpread (Spreads)



Reader (Raw)

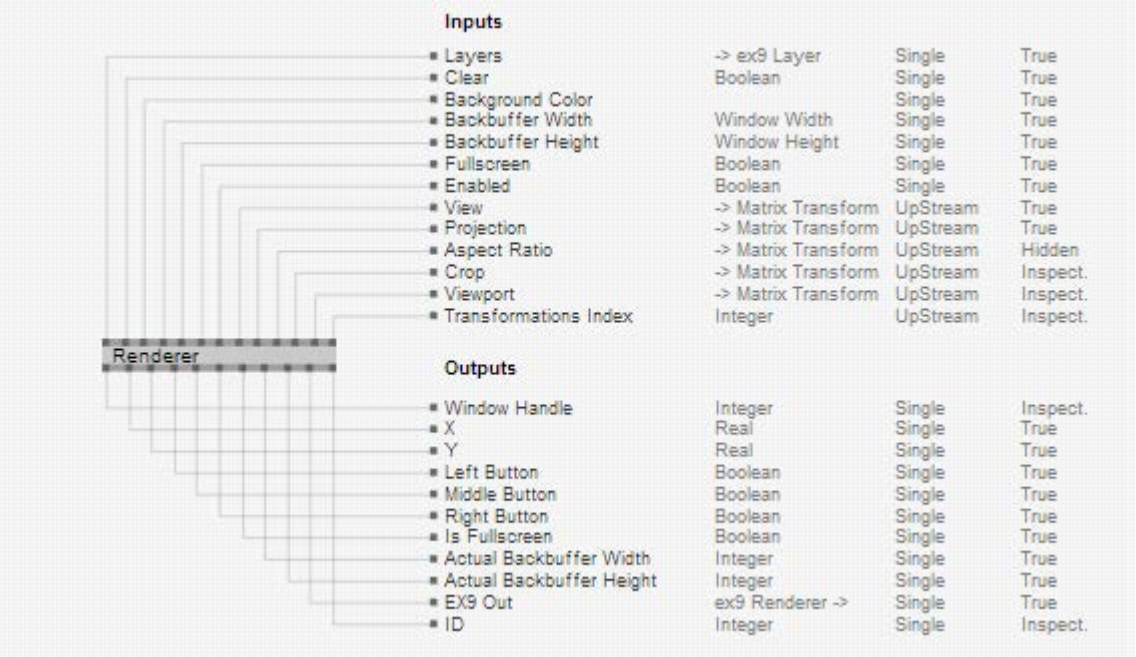


Reader (String)



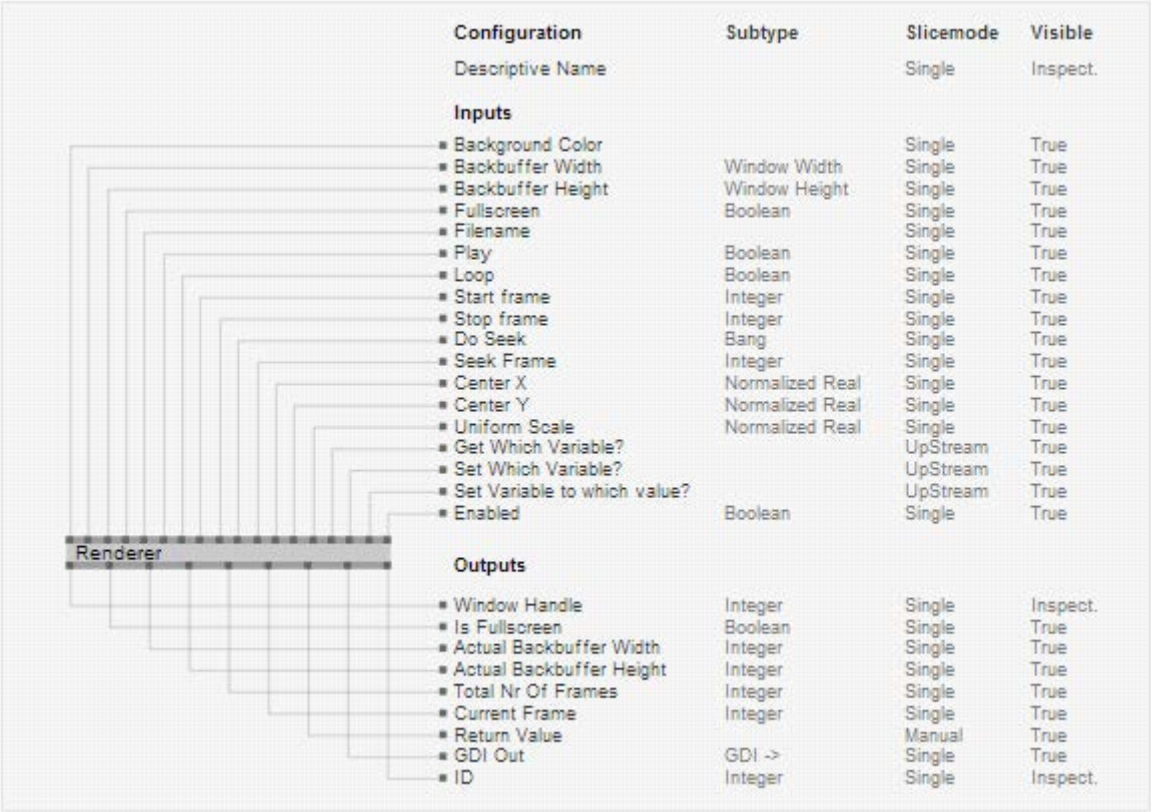
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Filename	merged subtype	Spread	True
Encoding	CodePages	Spread	True
Read	merged subtype	Spread	True
Evaluate	Boolean	Single	Inspect.
Outputs			
Window Handle	Integer	Single	Inspect.
Patch	Patch ->	Single	Inspect.
Content	merged subtype	Spread	True
ID	Integer	Single	Inspect.

Renderer (EX9)



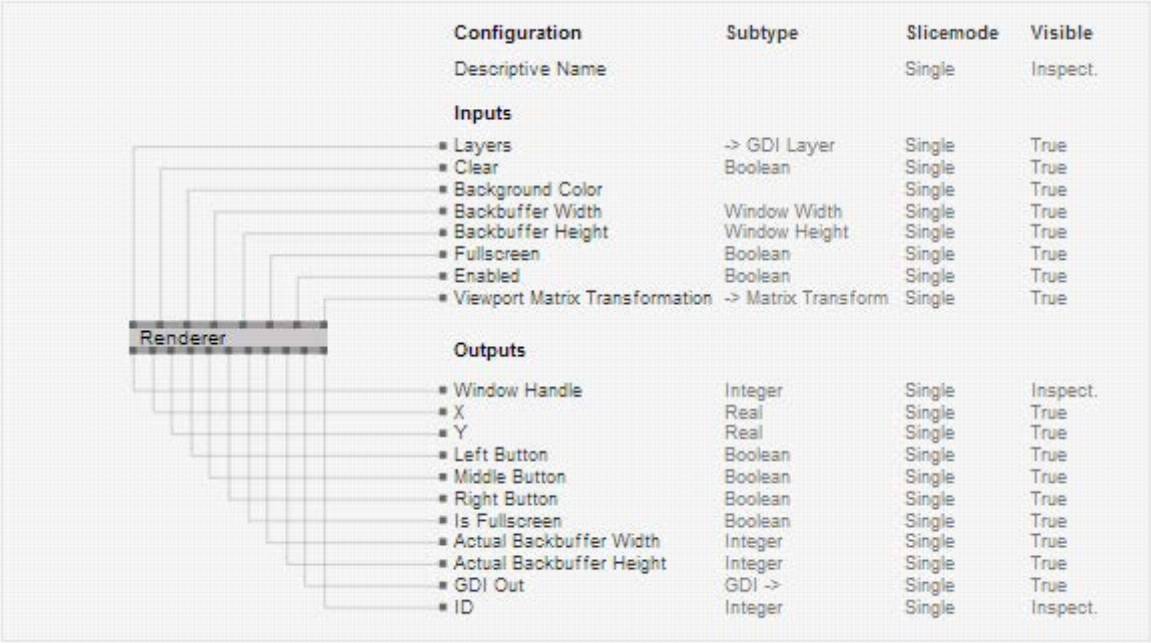
Inputs			
Layers	-> ex9 Layer	Single	True
Clear	Boolean	Single	True
Background Color		Single	True
Backbuffer Width	Window Width	Single	True
Backbuffer Height	Window Height	Single	True
Fullscreen	Boolean	Single	True
Enabled	Boolean	Single	True
View	-> Matrix Transform	UpStream	True
Projection	-> Matrix Transform	UpStream	True
Aspect Ratio	-> Matrix Transform	UpStream	Hidden
Crop	-> Matrix Transform	UpStream	Inspect.
Viewport	-> Matrix Transform	UpStream	Inspect.
Transformations Index	Integer	UpStream	Inspect.
Outputs			
Window Handle	Integer	Single	Inspect.
X	Real	Single	True
Y	Real	Single	True
Left Button	Boolean	Single	True
Middle Button	Boolean	Single	True
Right Button	Boolean	Single	True
Is Fullscreen	Boolean	Single	True
Actual Backbuffer Width	Integer	Single	True
Actual Backbuffer Height	Integer	Single	True
EX9 Out	ex9 Renderer ->	Single	True
ID	Integer	Single	Inspect.

Renderer (Flash)



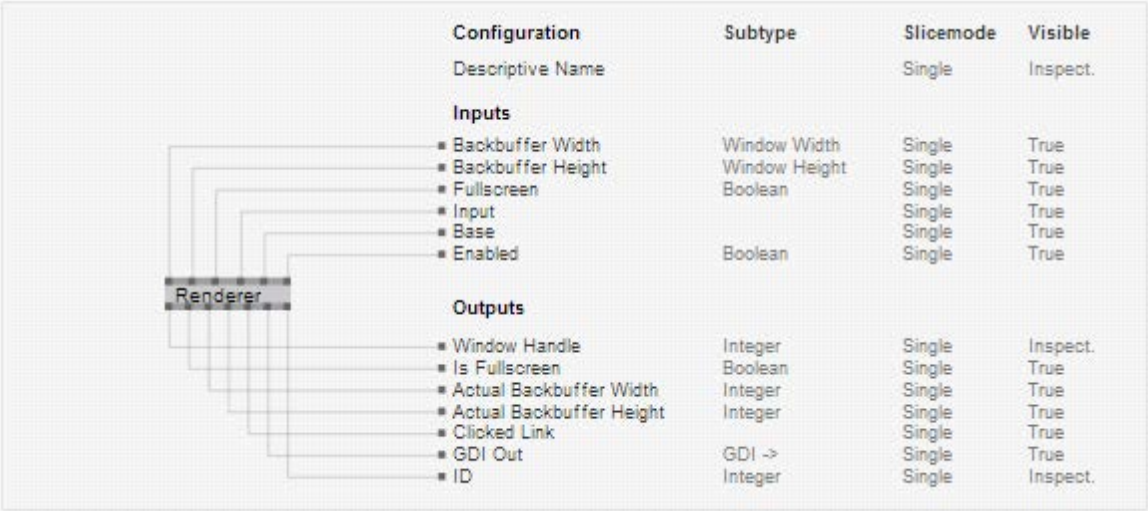
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Background Color		Single	True
Backbuffer Width	Window Width	Single	True
Backbuffer Height	Window Height	Single	True
Fullscreen	Boolean	Single	True
Filename		Single	True
Play	Boolean	Single	True
Loop	Boolean	Single	True
Start frame	Integer	Single	True
Stop frame	Integer	Single	True
Do Seek	Bang	Single	True
Seek Frame	Integer	Single	True
Center X	Normalized Real	Single	True
Center Y	Normalized Real	Single	True
Uniform Scale	Normalized Real	Single	True
Get Which Variable?		UpStream	True
Set Which Variable?		UpStream	True
Set Variable to which value?		UpStream	True
Enabled	Boolean	Single	True
Outputs			
Window Handle	Integer	Single	Inspect.
Is Fullscreen	Boolean	Single	True
Actual Backbuffer Width	Integer	Single	True
Actual Backbuffer Height	Integer	Single	True
Total Nr Of Frames	Integer	Single	True
Current Frame	Integer	Single	True
Return Value		Manual	True
GDI Out	GDI ->	Single	True
ID	Integer	Single	Inspect.

Renderer (GDI)



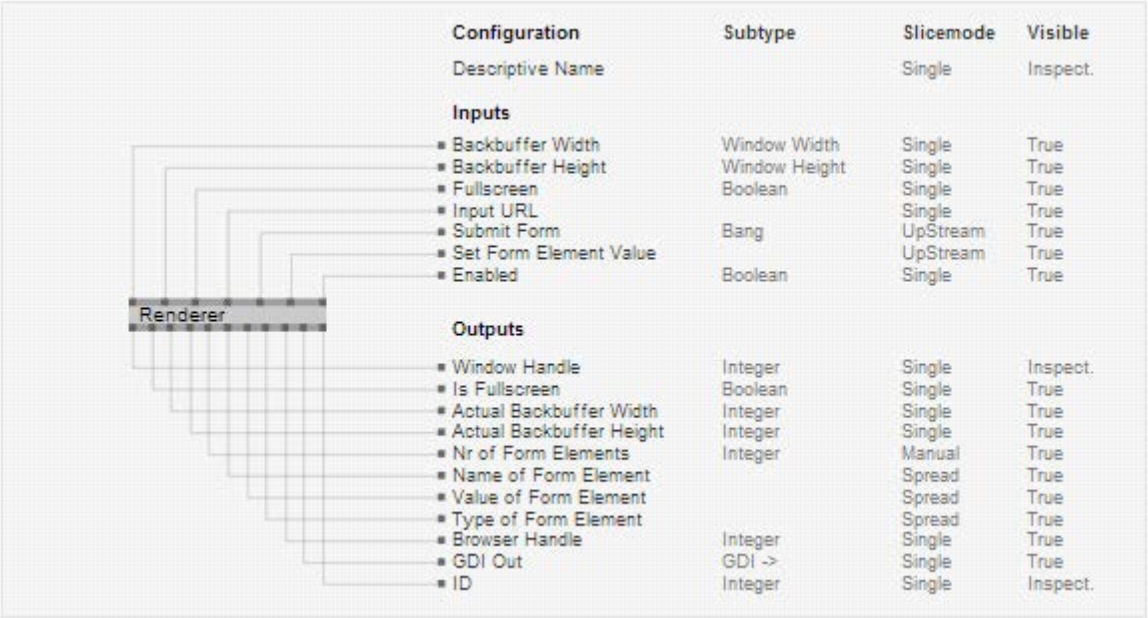
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Layers	-> GDI Layer	Single	True
Clear	Boolean	Single	True
Background Color		Single	True
Backbuffer Width	Window Width	Single	True
Backbuffer Height	Window Height	Single	True
Fullscreen	Boolean	Single	True
Enabled	Boolean	Single	True
Viewport Matrix Transformation	-> Matrix Transform	Single	True
Outputs			
Window Handle	Integer	Single	Inspect.
X	Real	Single	True
Y	Real	Single	True
Left Button	Boolean	Single	True
Middle Button	Boolean	Single	True
Right Button	Boolean	Single	True
Is Fullscreen	Boolean	Single	True
Actual Backbuffer Width	Integer	Single	True
Actual Backbuffer Height	Integer	Single	True
GDI Out	GDI ->	Single	True
ID	Integer	Single	Inspect.

Renderer (HTML String)



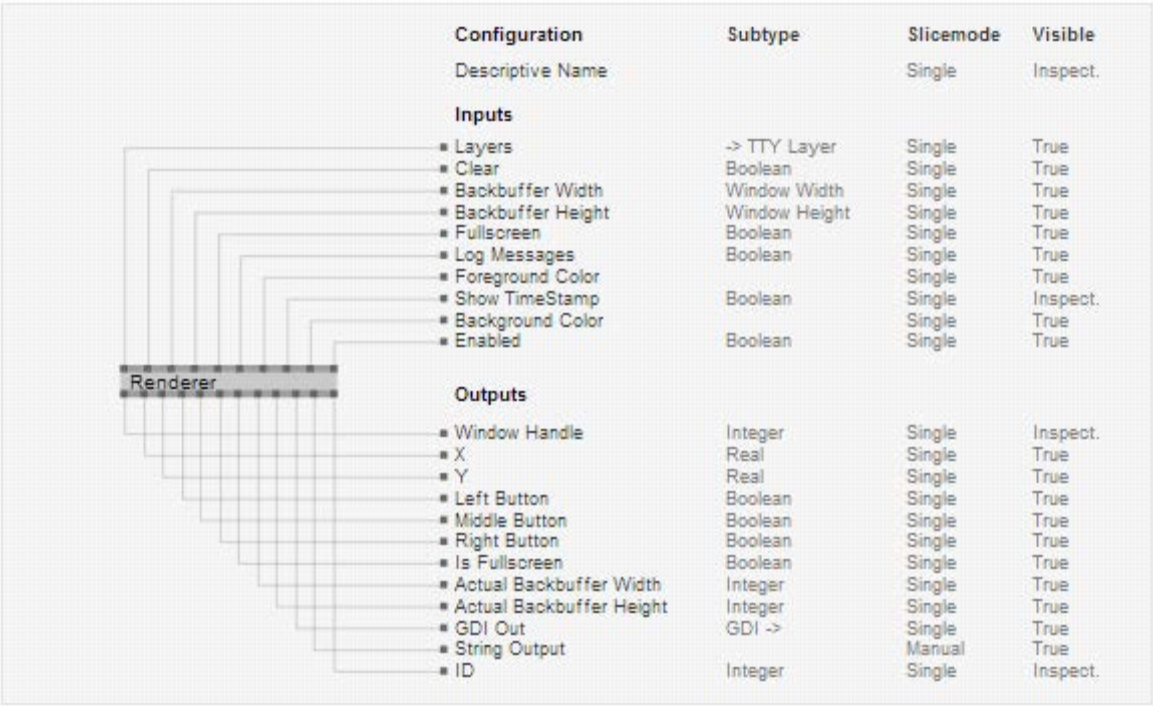
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Backbuffer Width	Window Width	Single	True
Backbuffer Height	Window Height	Single	True
Fullscreen	Boolean	Single	True
Input		Single	True
Base		Single	True
Enabled	Boolean	Single	True
Outputs			
Window Handle	Integer	Single	Inspect.
Is Fullscreen	Boolean	Single	True
Actual Backbuffer Width	Integer	Single	True
Actual Backbuffer Height	Integer	Single	True
Clicked Link		Single	True
GDI Out	GDI ->	Single	True
ID	Integer	Single	Inspect.

Renderer (HTML URL)



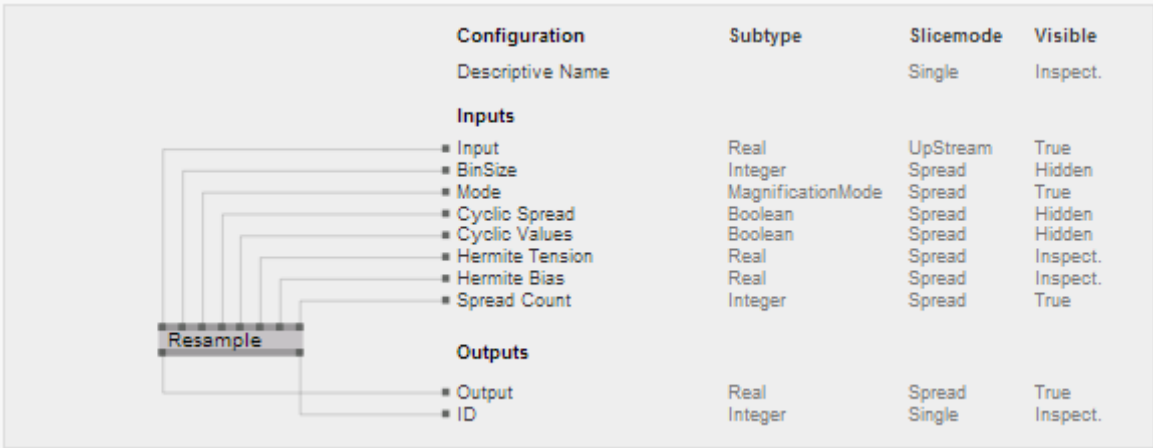
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Backbuffer Width	Window Width	Single	True
Backbuffer Height	Window Height	Single	True
Fullscreen	Boolean	Single	True
Input URL		Single	True
Submit Form	Bang	UpStream	True
Set Form Element Value		UpStream	True
Enabled	Boolean	Single	True
Outputs			
Window Handle	Integer	Single	Inspect.
Is Fullscreen	Boolean	Single	True
Actual Backbuffer Width	Integer	Single	True
Actual Backbuffer Height	Integer	Single	True
Nr of Form Elements	Integer	Manual	True
Name of Form Element		Spread	True
Value of Form Element		Spread	True
Type of Form Element		Spread	True
Browser Handle	Integer	Single	True
GDI Out	GDI ->	Single	True
ID	Integer	Single	Inspect.

Renderer (TTY)



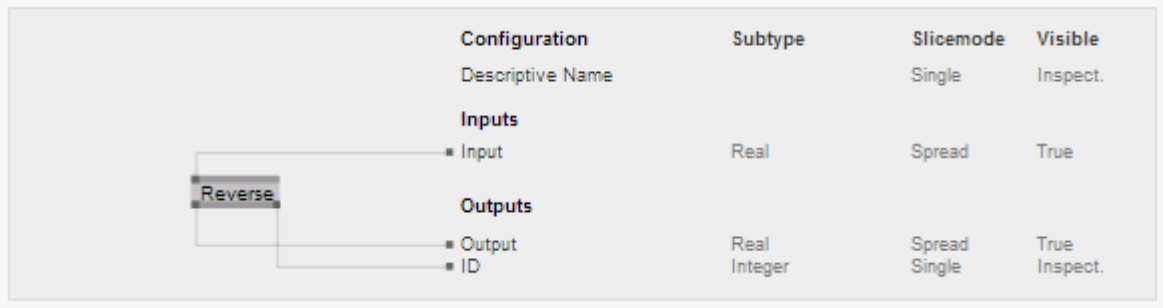
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Layers	-> TTY Layer	Single	True
Clear	Boolean	Single	True
Backbuffer Width	Window Width	Single	True
Backbuffer Height	Window Height	Single	True
Fullscreen	Boolean	Single	True
Log Messages	Boolean	Single	True
Foreground Color		Single	True
Show TimeStamp	Boolean	Single	Inspect.
Background Color		Single	True
Enabled	Boolean	Single	True
Outputs			
Window Handle	Integer	Single	Inspect.
X	Real	Single	True
Y	Real	Single	True
Left Button	Boolean	Single	True
Middle Button	Boolean	Single	True
Right Button	Boolean	Single	True
Is Fullscreen	Boolean	Single	True
Actual Backbuffer Width	Integer	Single	True
Actual Backbuffer Height	Integer	Single	True
GDI Out	GDI ->	Single	True
String Output		Manual	True
ID	Integer	Single	Inspect.

Resample (Spreads)

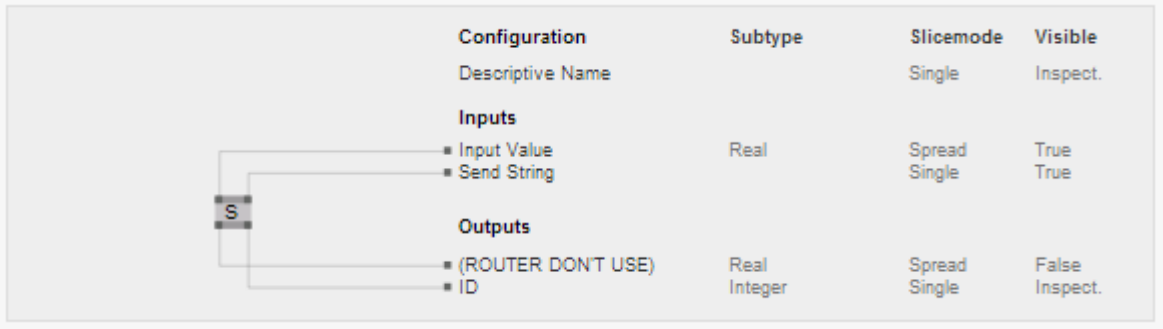


Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Input	Real	UpStream	True
BinSize	Integer	Spread	Hidden
Mode	MagnificationMode	Spread	True
Cyclic Spread	Boolean	Spread	Hidden
Cyclic Values	Boolean	Spread	Hidden
Hermite Tension	Real	Spread	Inspect.
Hermite Bias	Real	Spread	Inspect.
Spread Count	Integer	Spread	True
Outputs			
Output	Real	Spread	True
ID	Integer	Single	Inspect.

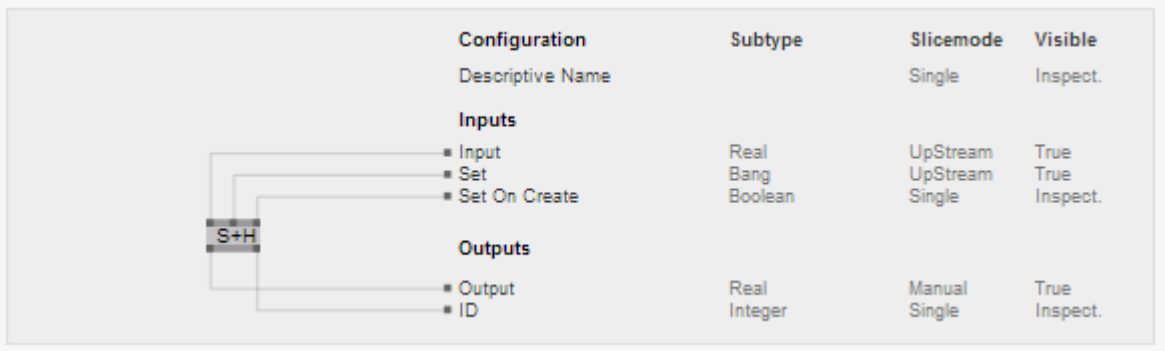
Reverse (Spreads)



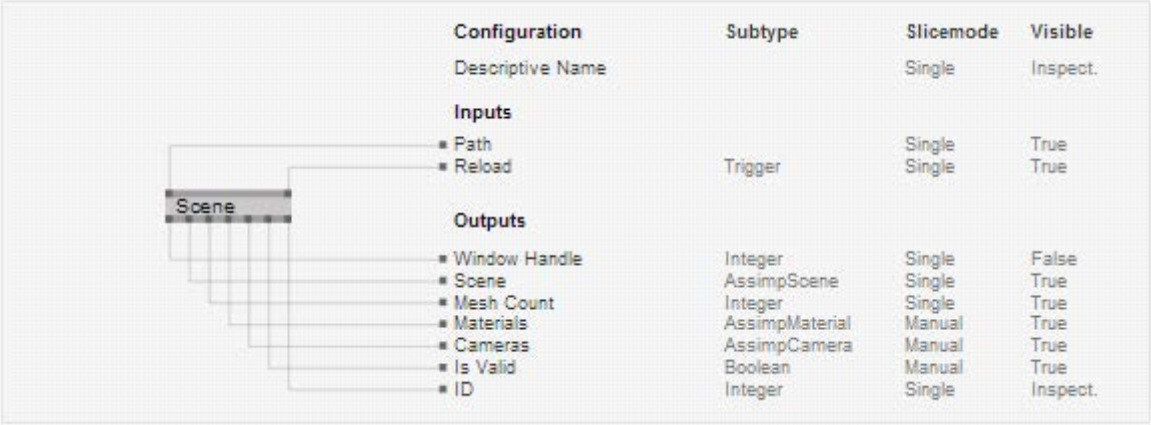
S (Value)



S+H (Animation)

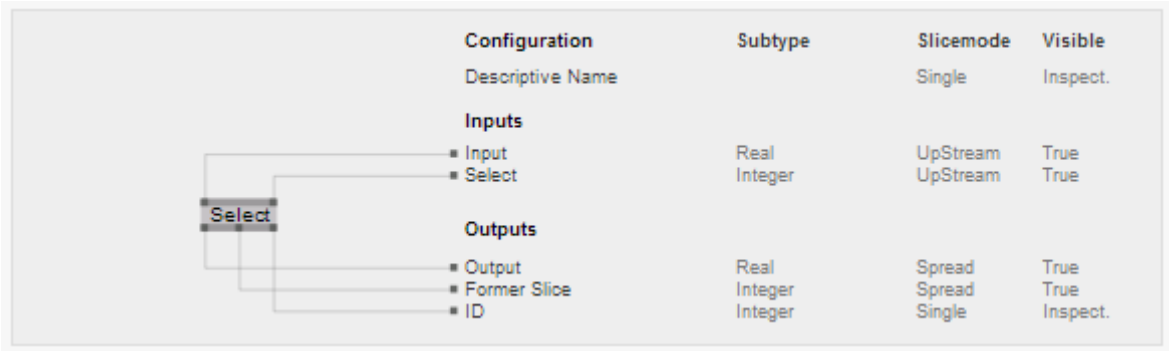


Scene (Assimp)



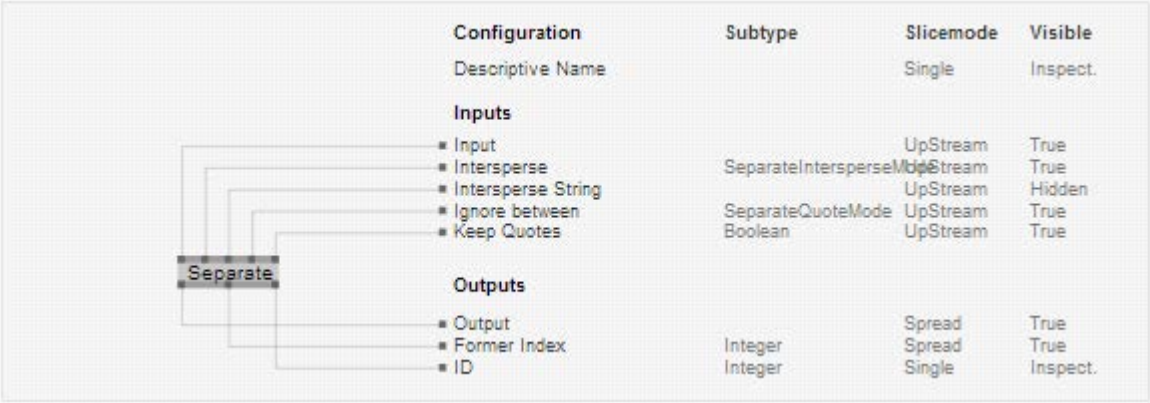
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Path		Single	True
Reload	Trigger	Single	True
Outputs			
Window Handle	Integer	Single	False
Scene	AssimpScene	Single	True
Mesh Count	Integer	Single	True
Materials	AssimpMaterial	Manual	True
Cameras	AssimpCamera	Manual	True
Is Valid	Boolean	Manual	True
ID	Integer	Single	Inspect.

Select (Value)



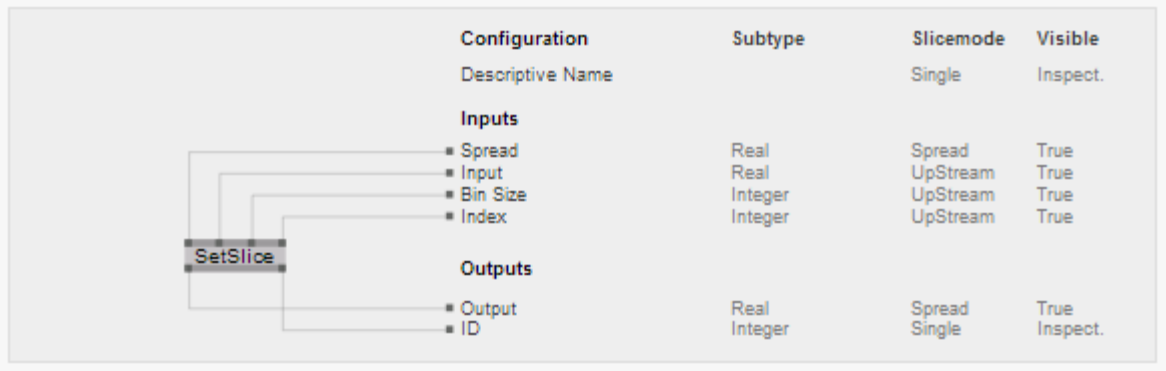
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Input	Real	UpStream	True
Select	Integer	UpStream	True
Outputs			
Output	Real	Spread	True
Former Slice	Integer	Spread	True
ID	Integer	Single	Inspect.

Separate (String)



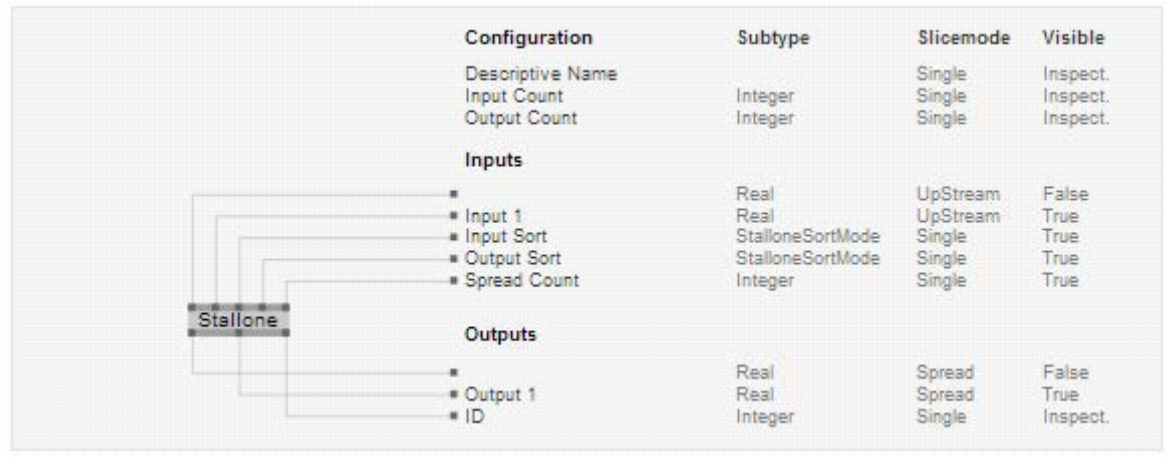
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Input		UpStream	True
Intersperse	SeparateIntersperseMode	UpStream	True
Intersperse String		UpStream	Hidden
Ignore between	SeparateQuoteMode	UpStream	True
Keep Quotes	Boolean	UpStream	True
Outputs			
Output		Spread	True
Former Index	Integer	Spread	True
ID	Integer	Single	Inspect.

SetSlice (Spreads)



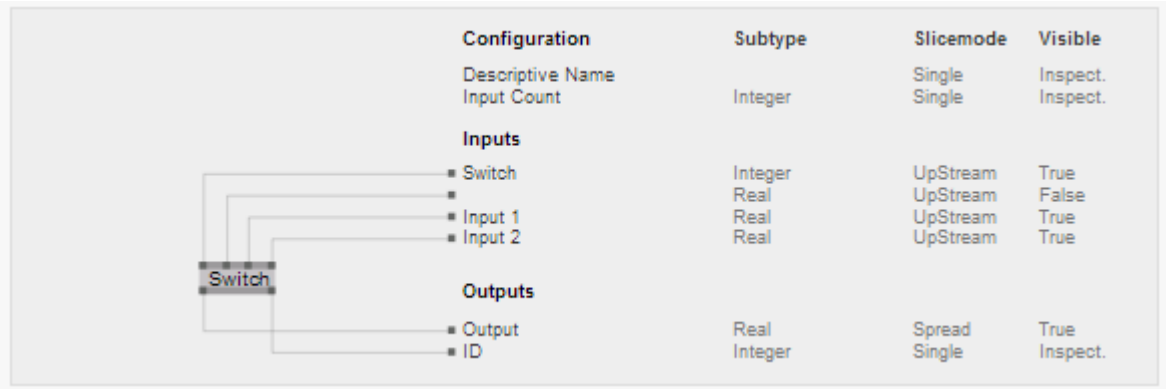
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Spread	Real	Spread	True
Input	Real	UpStream	True
Bin Size	Integer	UpStream	True
Index	Integer	UpStream	True
Outputs			
Output	Real	Spread	True
ID	Integer	Single	Inspect.

Stallone (Spreads)



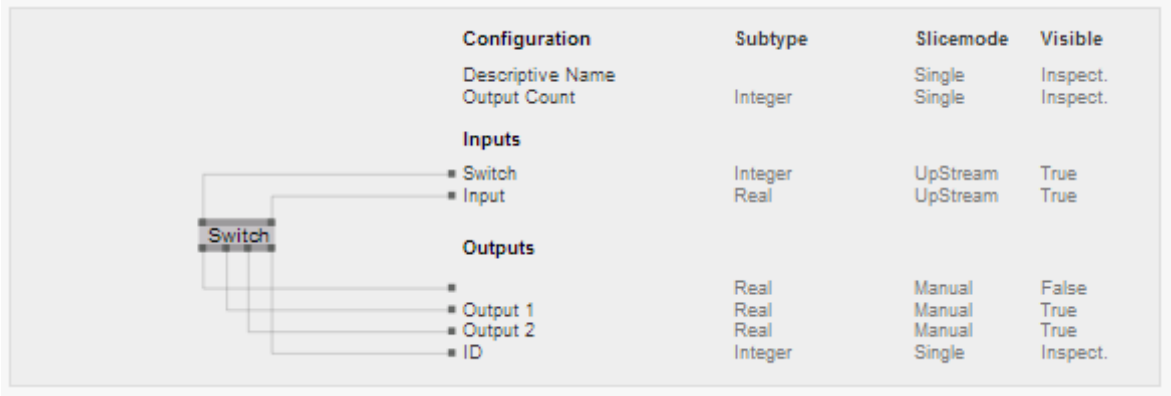
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Input Count	Integer	Single	Inspect.
Output Count	Integer	Single	Inspect.
Inputs			
	Real	UpStream	False
Input 1	Real	UpStream	True
Input Sort	StalloneSortMode	Single	True
Output Sort	StalloneSortMode	Single	True
Spread Count	Integer	Single	True
Outputs			
	Real	Spread	False
Output 1	Real	Spread	True
ID	Integer	Single	Inspect.

Switch (Value Input)



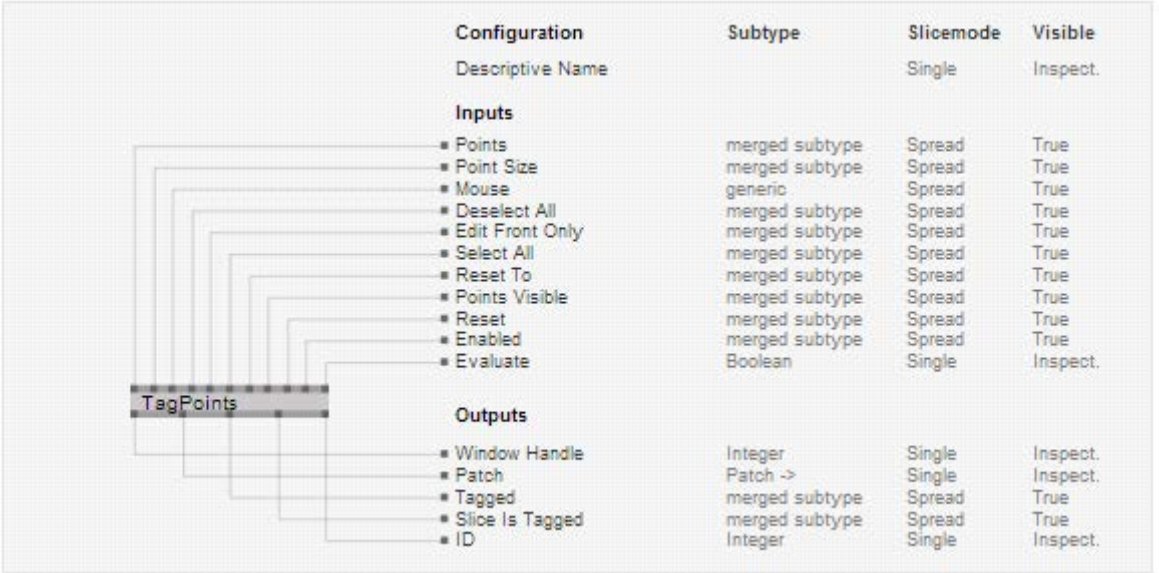
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Input Count	Integer	Single	Inspect.
Inputs			
Switch	Integer	UpStream	True
	Real	UpStream	False
Input 1	Real	UpStream	True
Input 2	Real	UpStream	True
Outputs			
Output	Real	Spread	True
ID	Integer	Single	Inspect.

Switch (Value Output)



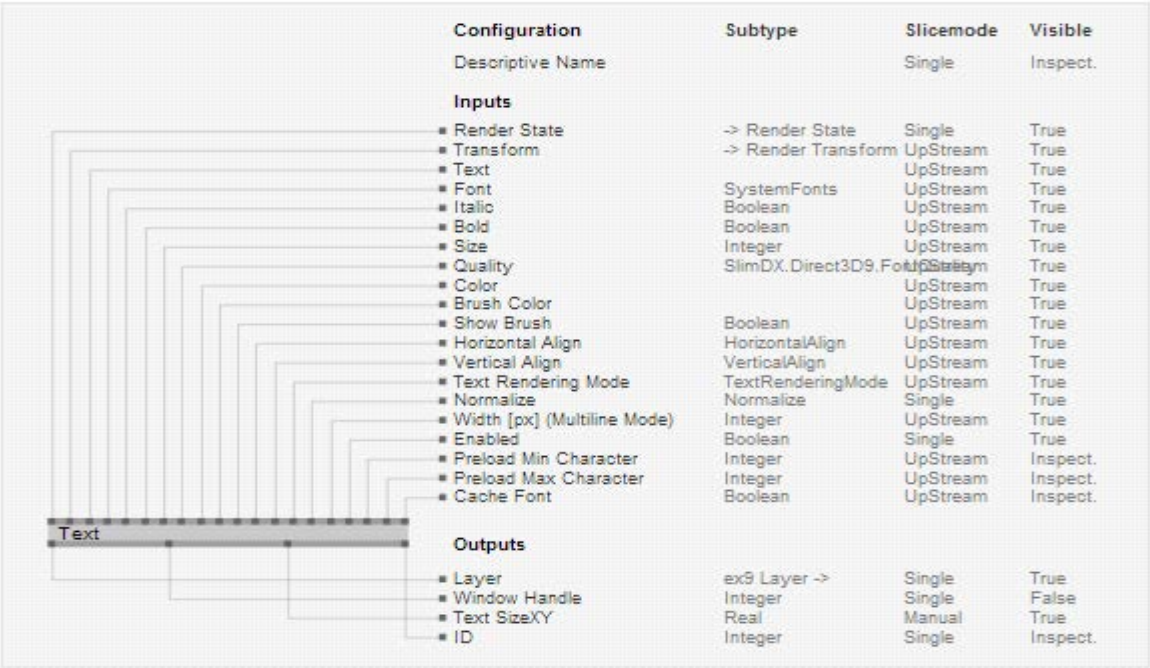
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Output Count	Integer	Single	Inspect.
Inputs			
Switch	Integer	UpStream	True
Input	Real	UpStream	True
Outputs			
	Real	Manual	False
Output 1	Real	Manual	True
Output 2	Real	Manual	True
ID	Integer	Single	Inspect.

TagPoints (3D Vector)



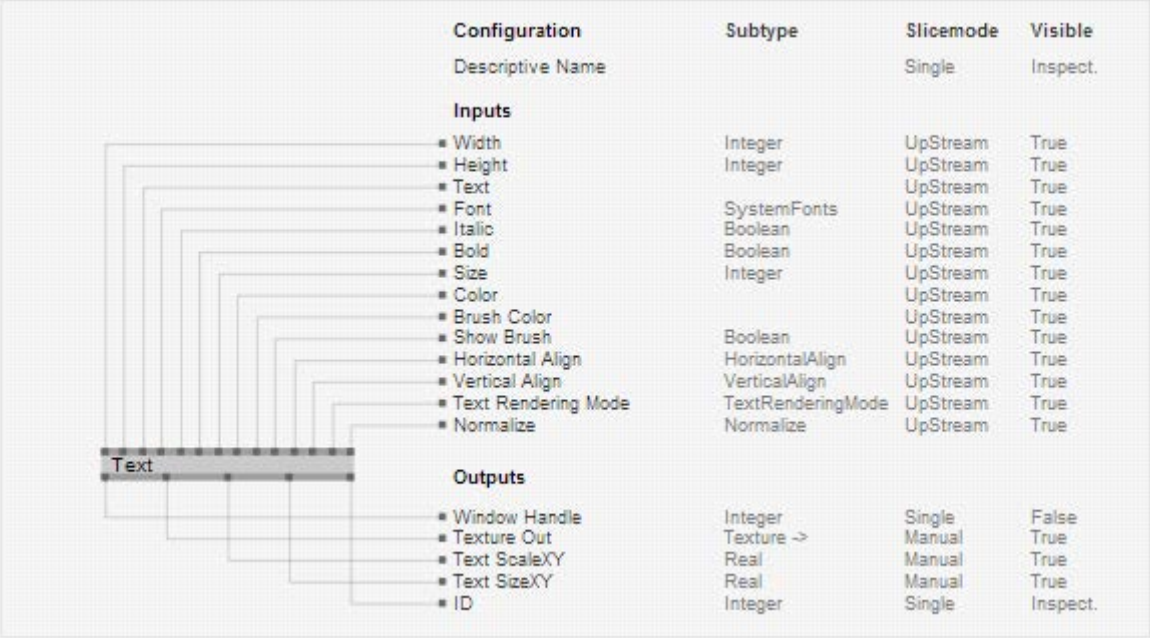
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Points	merged subtype	Spread	True
Point Size	merged subtype	Spread	True
Mouse	generic	Spread	True
Deselect All	merged subtype	Spread	True
Edit Front Only	merged subtype	Spread	True
Select All	merged subtype	Spread	True
Reset To	merged subtype	Spread	True
Points Visible	merged subtype	Spread	True
Reset	merged subtype	Spread	True
Enabled	merged subtype	Spread	True
Evaluate	Boolean	Single	Inspect.
Outputs			
Window Handle	Integer	Single	Inspect.
Patch	Patch ->	Single	Inspect.
Tagged	merged subtype	Spread	True
Slice Is Tagged	merged subtype	Spread	True
ID	Integer	Single	Inspect.

Text (EX9)



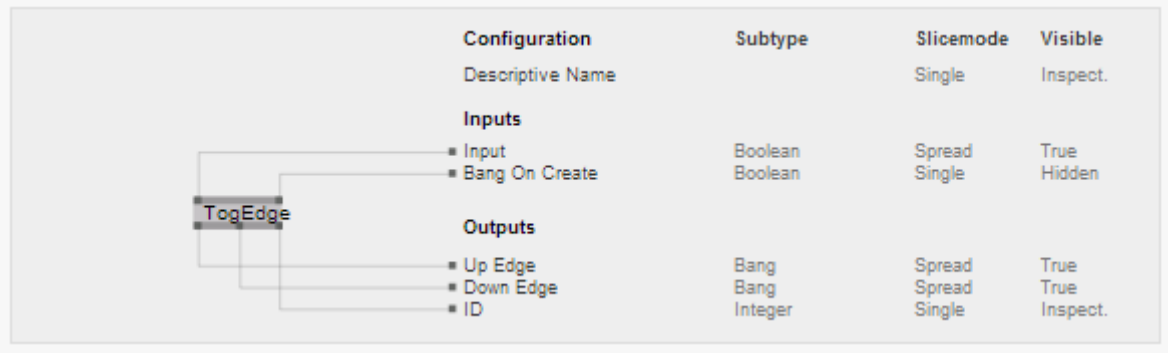
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Render State	-> Render State	Single	True
Transform	-> Render Transform	UpStream	True
Text		UpStream	True
Font	SystemFonts	UpStream	True
Italic	Boolean	UpStream	True
Bold	Boolean	UpStream	True
Size	Integer	UpStream	True
Quality	SlimDX.Direct3D9.Fo	UpStream	True
Color		UpStream	True
Brush Color		UpStream	True
Show Brush	Boolean	UpStream	True
Horizontal Align	HorizontalAlign	UpStream	True
Vertical Align	VerticalAlign	UpStream	True
Text Rendering Mode	TextRenderingMode	UpStream	True
Normalize	Normalize	Single	True
Width [px] (Multiline Mode)	Integer	UpStream	True
Enabled	Boolean	Single	True
Preload Min Character	Integer	UpStream	Inspect.
Preload Max Character	Integer	UpStream	Inspect.
Cache Font	Boolean	UpStream	Inspect.
Outputs			
Layer	ex9 Layer ->	Single	True
Window Handle	Integer	Single	False
Text SizeXY	Real	Manual	True
ID	Integer	Single	Inspect.

Text (EX9.Texture)



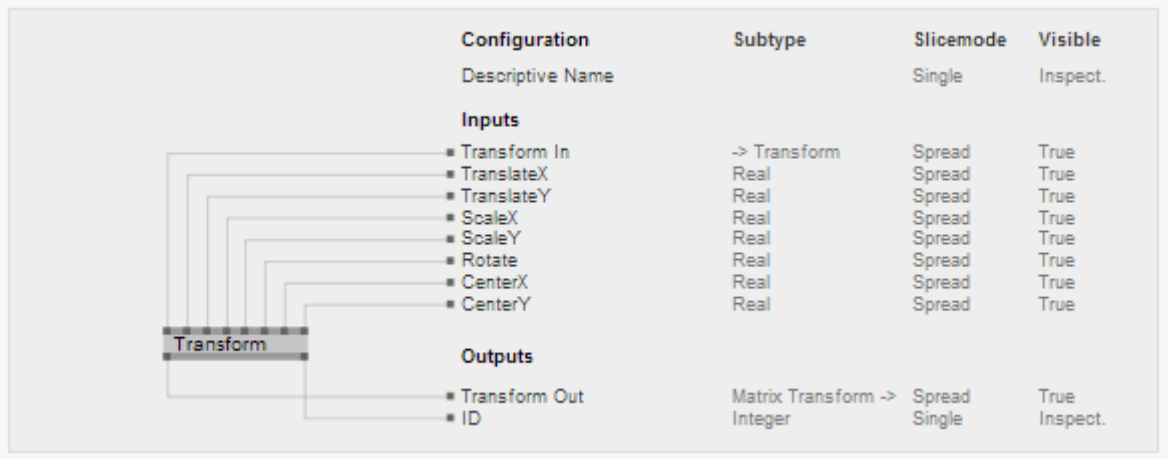
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Width	Integer	UpStream	True
Height	Integer	UpStream	True
Text		UpStream	True
Font	SystemFonts	UpStream	True
Italic	Boolean	UpStream	True
Bold	Boolean	UpStream	True
Size	Integer	UpStream	True
Color		UpStream	True
Brush Color		UpStream	True
Show Brush	Boolean	UpStream	True
Horizontal Align	HorizontalAlign	UpStream	True
Vertical Align	VerticalAlign	UpStream	True
Text Rendering Mode	TextRenderingMode	UpStream	True
Normalize	Normalize	UpStream	True
Outputs			
Window Handle	Integer	Single	False
Texture Out	Texture ->	Manual	True
Text ScaleXY	Real	Manual	True
Text SizeXY	Real	Manual	True
ID	Integer	Single	Inspect.

TogEdge (Animation)



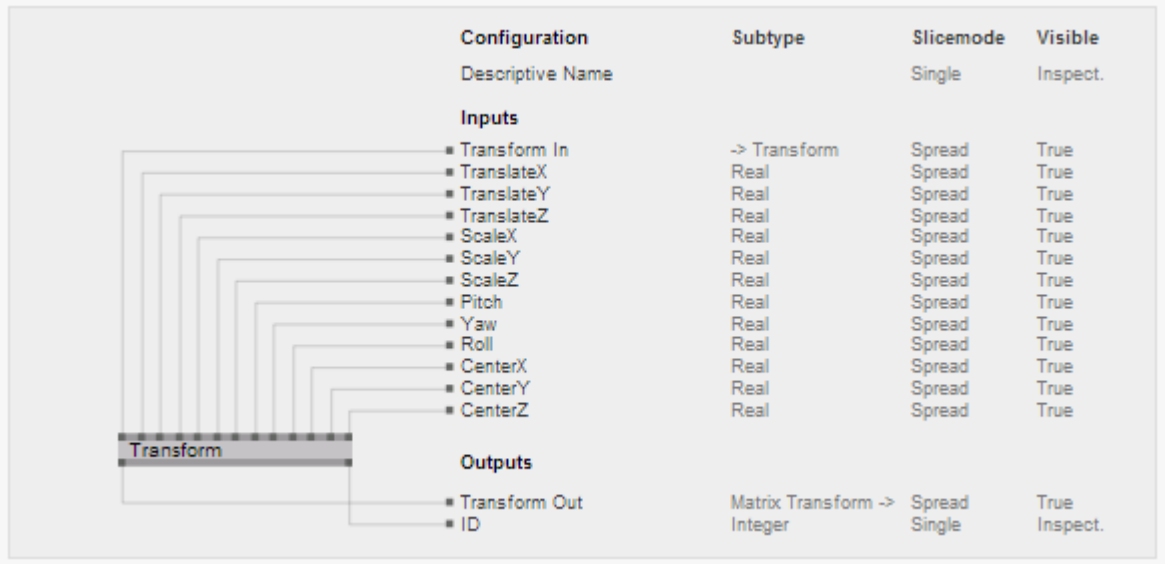
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Input	Boolean	Spread	True
Bang On Create	Boolean	Single	Hidden
Outputs			
Up Edge	Bang	Spread	True
Down Edge	Bang	Spread	True
ID	Integer	Single	Inspect.

Transform (Transform 2d)



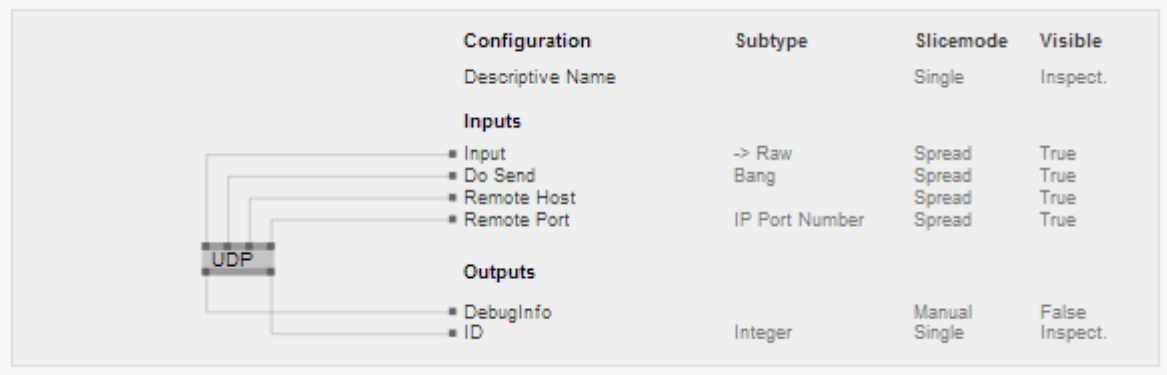
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Transform In	-> Transform	Spread	True
TranslateX	Real	Spread	True
TranslateY	Real	Spread	True
ScaleX	Real	Spread	True
ScaleY	Real	Spread	True
Rotate	Real	Spread	True
CenterX	Real	Spread	True
CenterY	Real	Spread	True
Outputs			
Transform Out	Matrix Transform ->	Spread	True
ID	Integer	Single	Inspect.

Transform (Transform 3d)



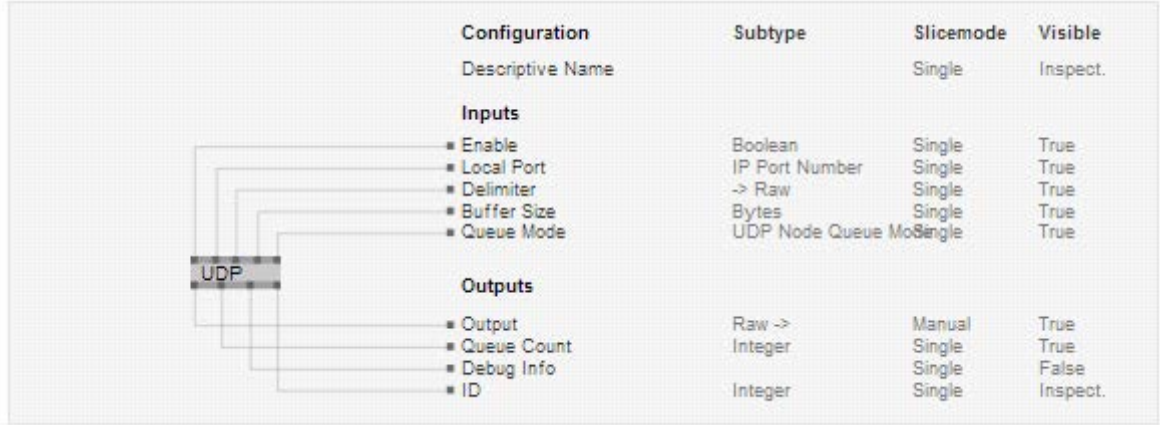
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Transform In	-> Transform	Spread	True
TranslateX	Real	Spread	True
TranslateY	Real	Spread	True
TranslateZ	Real	Spread	True
ScaleX	Real	Spread	True
ScaleY	Real	Spread	True
ScaleZ	Real	Spread	True
Pitch	Real	Spread	True
Yaw	Real	Spread	True
Roll	Real	Spread	True
CenterX	Real	Spread	True
CenterY	Real	Spread	True
CenterZ	Real	Spread	True
Outputs			
Transform Out	Matrix Transform ->	Spread	True
ID	Integer	Single	Inspect.

UDP (Network Client)



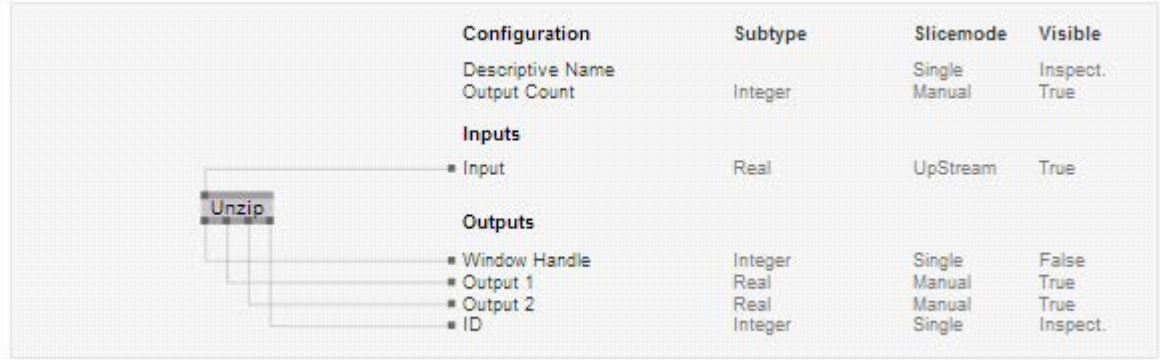
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Input	-> Raw	Spread	True
Do Send	Bang	Spread	True
Remote Host		Spread	True
Remote Port	IP Port Number	Spread	True
Outputs			
DebugInfo		Manual	False
ID	Integer	Single	Inspect.

UDP (Network Server)



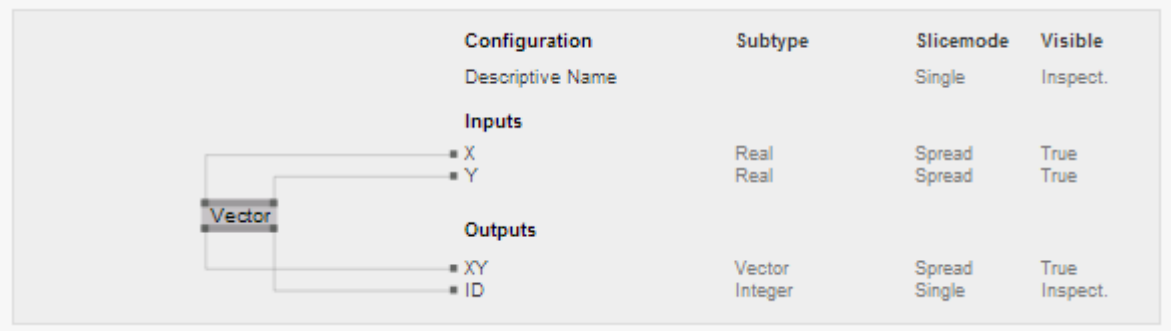
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Enable	Boolean	Single	True
Local Port	IP Port Number	Single	True
Delimiter	-> Raw	Single	True
Buffer Size	Bytes	Single	True
Queue Mode	UDP Node Queue Mo	Single	True
Outputs			
Output	Raw ->	Manual	True
Queue Count	Integer	Single	True
Debug Info		Single	False
ID	Integer	Single	Inspect.

Unzip (Value)

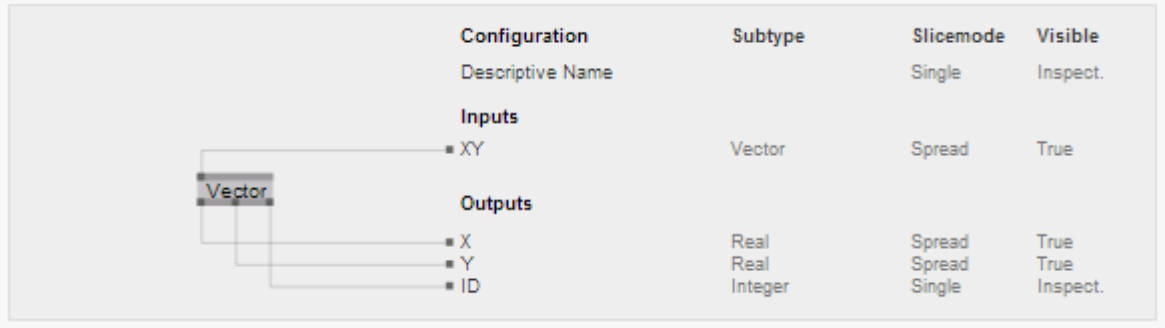


Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Output Count	Integer	Manual	True
Inputs			
Input	Real	UpStream	True
Outputs			
Window Handle	Integer	Single	False
Output 1	Real	Manual	True
Output 2	Real	Manual	True
ID	Integer	Single	Inspect.

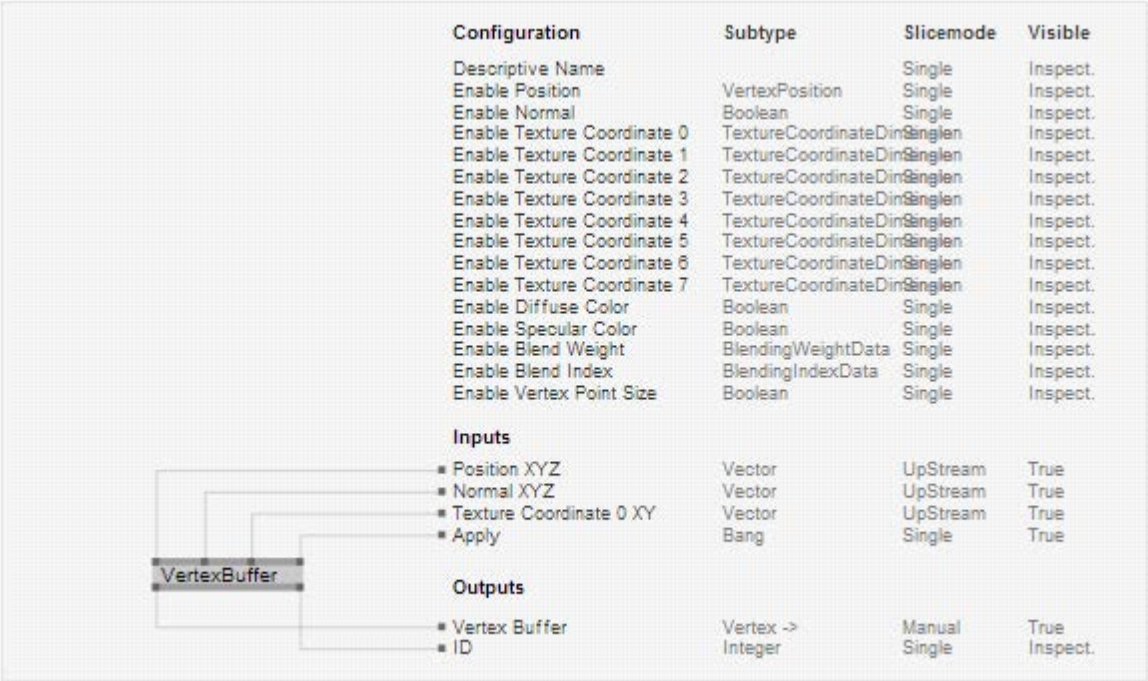
Vector (2d Join)



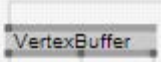
Vector (2d Split)



VertexBuffer (EX9.Geometry Join)



VertexBuffer (EX9.Geometry Split)



Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Device	Integer	Single	Inspect.
Enable Position	VertexPosition	Single	Inspect.
Enable Normal	Boolean	Single	Inspect.
Enable Tangent	Boolean	Single	Inspect.
Enable Binormal	Boolean	Single	Inspect.
Enable Texture Coordinate 0	TextureCoordinateDim	Single	Inspect.
Enable Texture Coordinate 1	TextureCoordinateDim	Single	Inspect.
Enable Texture Coordinate 2	TextureCoordinateDim	Single	Inspect.
Enable Texture Coordinate 3	TextureCoordinateDim	Single	Inspect.
Enable Texture Coordinate 4	TextureCoordinateDim	Single	Inspect.
Enable Texture Coordinate 5	TextureCoordinateDim	Single	Inspect.
Enable Texture Coordinate 6	TextureCoordinateDim	Single	Inspect.
Enable Texture Coordinate 7	TextureCoordinateDim	Single	Inspect.
Enable Diffuse Color	Boolean	Single	Inspect.
Enable Specular Color	Boolean	Single	Inspect.
Enable Blend Weight	BlendingWeightData	Single	Inspect.
Enable Blend Index	BlendingIndexData	Single	Inspect.
Enable Vertex Point Size	Boolean	Single	Inspect.
Inputs			
Vertex Buffer	-> Vertex	UpStream	True
Outputs			
Position XYZ	Vector	Manual	True
Normal XYZ	Vector	Manual	True
ID	Integer	Single	Inspect.

VideoIn (DShow9)

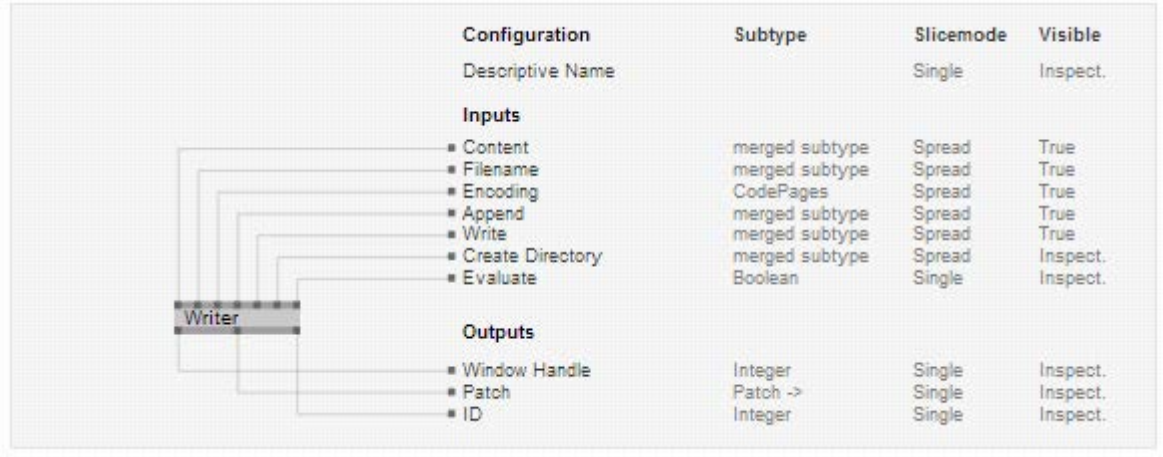
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Expose Graph to ROT	Boolean	Single	Inspect.
Reference Clock	ReferenceClocks	Single	Inspect.
Force Preview Output (via SmartR	Boolean	Single	Inspect.
Inputs			
Enabled	Boolean	Single	True
Property Page	Bang	Single	True
Framerate	Real	Single	True
Width	Integer	Single	True
Height	Integer	Single	True
Video Format	VideoFormats	Single	True
Analog Video Standard	AnalogVideoFormats	Single	True
Analog Video Input	AnalogVideoInputs	Single	True
Driver	VideoInputs	Single	True
Brightness	Integer	Single	Inspect.
Contrast	Integer	Single	Inspect.
Hue	Integer	Single	Inspect.
Saturation	Integer	Single	Inspect.
Sharpness	Integer	Single	Inspect.
Gamma	Integer	Single	Inspect.
ColorEnable	Integer	Single	Inspect.
WhiteBalance	Integer	Single	Inspect.
Backlight Compensation	Integer	Single	Inspect.
Gain	Integer	Single	Inspect.
Trigger Mode	Boolean	Single	Inspect.
Trigger	Bang	Single	Inspect.
Pan	Integer	Single	Inspect.
Tilt	Integer	Single	Inspect.
Roll	Integer	Single	Inspect.
Zoom	Integer	Single	Inspect.
Exposure	Integer	Single	Inspect.
Iris	Integer	Single	Inspect.
Focus	Integer	Single	Inspect.
Adjust System Time	Integer	Single	Inspect.
Outputs			
Video	DirectShow ->	Single	True
Preview	DirectShow ->	Single	True
Stream Capabilities		Single	True
Actual StreamFormat		Single	True
ID	Integer	Single	Inspect.

VideoTexture (EX9.Texture VMR9)

Writer (Raw)

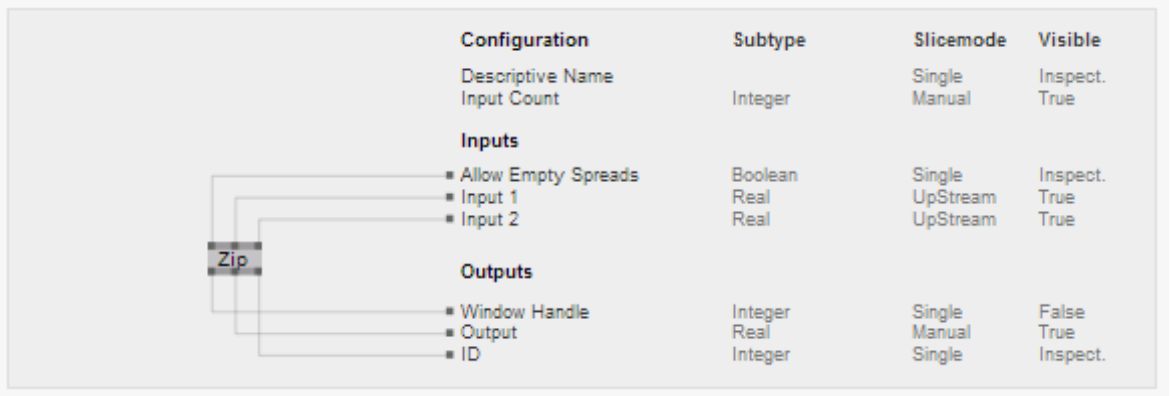
Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Content	-> Raw	Spread	True
Filename		Spread	True
Write	Bang	Spread	True
Append	Boolean	Spread	True
Create Directory	Boolean	Spread	Inspect.
Outputs			
ID	Integer	Single	Inspect.

Writer (String)



Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Inputs			
Content	merged subtype	Spread	True
Filename	merged subtype	Spread	True
Encoding	CodePages	Spread	True
Append	merged subtype	Spread	True
Write	merged subtype	Spread	True
Create Directory	merged subtype	Spread	Inspect.
Evaluate	Boolean	Single	Inspect.
Outputs			
Window Handle	Integer	Single	Inspect.
Patch	Patch ->	Single	Inspect.
ID	Integer	Single	Inspect.

Zip (Value)



Configuration	Subtype	Slicemode	Visible
Descriptive Name		Single	Inspect.
Input Count	Integer	Manual	True
Inputs			
Allow Empty Spreads	Boolean	Single	Inspect.
Input 1	Real	UpStream	True
Input 2	Real	UpStream	True
Outputs			
Window Handle	Integer	Single	False
Output	Real	Manual	True
ID	Integer	Single	Inspect.