VVVV

Programovací jazyk, který místo textového rozhraní pro programování využívá grafických objektů.

Jednotlivé operace nebo funkce jsou reprezentovány **uzly** a jsou propojeny pomocí **spojnic (links)**. Spojnice jsou vytvářeny pomocí myši.

Každý uzel může přijímat, zpracovávat nebo generovat data. Uzly mohou tyto funkce i kombinovat. Vstupy a výstupy jednotlivých uzlů jsou označeny malými černými čtverečky v horní a dolní části uzlu a nazývají se **špendlíky (pins)**, někdy též **inlets** a **outlets**.

Inlets reprezentují parametry/argumenty funkce, zatímco outlets reprezentují výsledek dané funkce.

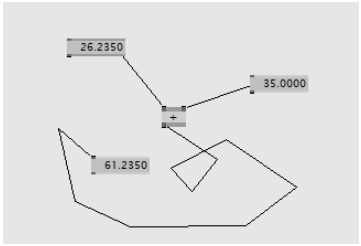
Menu v programu je vyvoláno pomocí stisknutí prostředního tlačítka myši nebo pomocí kombinace SPACE + pravé tlačítko myši. New Patch (Ctrl+P) vytvoří nové okno. Pro vložení nového uzlu klikneme dvakrát levým tlačítkem myši a otevře se seznam jmen uzlů, které lze vytvořit. Jména jsou seřazena podle abecedy.

Vytvoříme si uzel, který se jmenuje ***+(Value)***. Najetím myši na nově vytvořený uzel se zobrazí jeho popisek, tažením uzlu pomocí levého tlačítka myši můžeme měnit jeho polohu. Pokud myší najedeme na pins tohoto uzlu, zobrazí se nápověda, o jaký pin se jedná. V případě našeho uzlu jsou všechny hodnoty nastaveny defaultně na 0. Hodnotu v pins lze změnit tak, že na něj klikneme pravým tlačítkem myši a přepíšeme hodnotu. Druhou možností, jak změnit hodnotu v pinu, je kliknout pravým tlačítkem na pin a táhnout myší nahoru nebo dolů. Po změně hodnot v input pins se můžeme podívat, co je uloženo v output pinu – mělo by se jedna o součet hodnot v input pins.

Pro přehlednější zobrazení aktuální hodnoty v jednotlivých uzlech se používají tzv. **IOBoxes**. Jsou to uzly, které vizualizují hodnotu přímo ve čtverečku reprezentujícím uzel. IOBox je možné vložit pomocí uzlu ***IOBox(Value Advanced)***. Protože IOBox je často používaným typem uzlu, byla pro něj vytvořena rychlá zkratka – dvojitý klik pravým tlačítkem myši.

Spojování uzlů

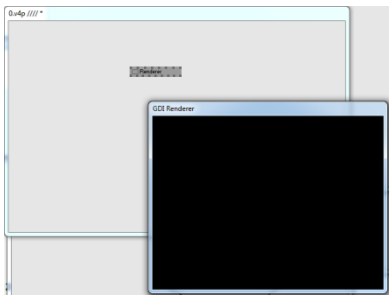
Spojení uzlů je jednoduché. Stačí myší (LTM) kliknout na některý pin IOBoxu a všechny piny ve scéně, které jsou schopny se s označeným pinem propojit, jsou zvýrazněny (zvětší se). Vybráním jednoho z těchto zvýrazněných pinů dojde k propojení. Propojení lze dosáhnout i obyčejným tažením myši mezi dvěma piny.

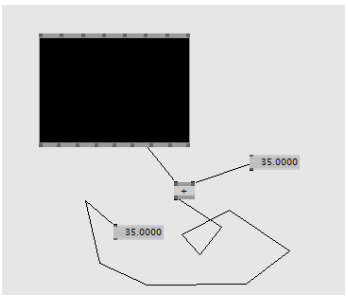


Pro navrácení hodnoty v uzlu na defaultní slouží kombinace ALT + pravé tlačítko myši nad příslušným uzlem.

Renderování

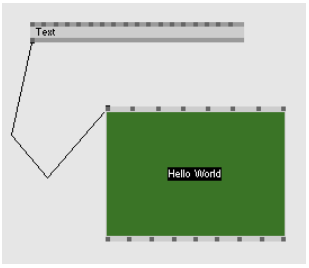
Pro vizuální zobrazení výstupu musíme mít k dispozici uzel pro renderování. Můžeme si zvolit z několika různých uzlů pro renderování, díky čemuž máme více možností, jak zobrazit různé typy dat. Prozatím si vystačíme s uzlem typu ***Renderer(GDI)***. Tento uzel má asociováno okno, které se objeví ihned po vytvoření uzlu. Pokud bychom chtěli toto okno umístit přímo do prostoru, kde máme všechny uzly, stačí na okno kliknout, aby bylo aktivní, a stisknout ALT + 2. Vrácení zpět do samostatného okna se provede pomocí ALT + 1.





Pro změnu barvy pozadí renderovacího okna je potřeba nalézt input pin označený jako Backround Color a pravým tlačítkem myši se zobrazí barevné pole. V něm lze měnit brightness (světlost) – tažením pravým tlačítkem myši nahoru/dolů, hue (odstín) – Caps Lock + pravé tlačítko myši, staturation (sytost) – při tažení zmáčknout CTRL, průhlednost se Shiftem.

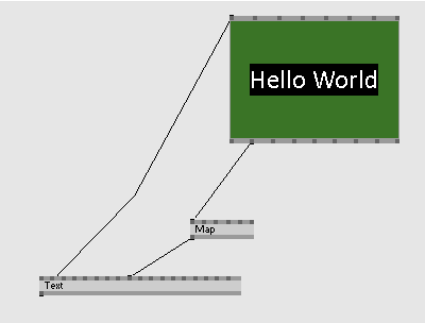
Pro zobrazení textu v renderovacím okně musíme vytvořit uzel ***Text(GDI)***, který musíme propojit s renderovacím uzlem – výstupní pin textového uzlu se vstupním pinem renderovacího uzlu. Pro vložení textu do textového uzlu musíme v jeho horních pinech najít ten s nápisem Text a pravým tlačítkem aktivovat políčko, do kterého napíšeme text „Hello World“.



Pomocí pinů v horní části textového uzlu se dá změnit font, velikost písma atd.

Nyní zkusíme do renderovacího okna zavést drobnou interakci. Vytvoříme navázání změny velikosti textu na vertikální pohyb myší. Dále nastavíme horní a dolní limit pro velikost písma, abychom se při změně pomocí myši drželi v rozumných mantinelech. Provedeme to tak, že namapujeme rozsah hodnot, které poskytuje pin Y v renderovacím uzlu (rozsah -1 až 1) na rozsah 15 až 50, který odpovídá minimální a maximální velikosti požadovaného fontu. K tomu budeme opotřebovat uzel ***Map(Value)***.

Můžeme jej vytvořit jako nový uzel nebo je možné modifikovat některý existující uzel, který už nepotřebujeme. Takovým je třeba ***+(Value)***. Dvojitým klikem LTM můžeme přepsat název uzlu, čímž změníme jeho typ i počet pinů. Všimněte si, že vazba output pinu na IOBox se ale neztratila. Nyní můžeme spojit výstup pinu Y renderovacího uzlu na pin Size u textového uzlu – propojení realizujeme přes uzel Map.



Pro uzel Map musíme nastavit hodnoty jeho input pinů následovně:

- Source Minimum: -1
- Source Maximum: 1
- Destination Minimum: 15
- Destination Maximum: 50
- Mapping: Clamp

Když nyní v renderovacím okně hýbeme myší nahoru a dolů, dochází k automatickému zvětšování a zmenšování nápisu.

Inspector

Měnit hodnoty uvnitř jednotlivých uzlů lze i elegantnějším způsobem, než jsme si ukázali. Pomocí CTRL + I spustíme tzv. **inspektor**, který poskytne přehled o nastavených hodnotách v daném uzlu. Stačí pouze kliknout na daný uzel a inspektor ukáže informace o všech pinech daného uzlu a samozřejmě je umožní editovat. Inspektor navíc obsahuje tzv. **konfigurační piny**, které nejsou k dispozici v základním zobrazení uzlu, jak jsme je používali dosud. To je dáno tím, že se nepředpokládá, že by docházelo k častým změnám jejich nastavení.

VVVV umožňuje manipulaci s více uzly najednou. Pro příklad si do scény vložíme 3 uzly typu ***IOBox(Color)***. (Stačí vložit pouze jeden, který zkopírujeme pomocí CTRL + c, CTRL + v.) Pokud nyní chceme např. změnit barvu všem těmto uzlům najednou, stačí je označit obdélníkovým výběrem. Pokud nyní měním hodnoty v inspektoru, mění se hodnoty ve všech třech uzlech.

Využití více inspektorů najednou

Pomocí CTRL SHIFT + I můžeme vyvolat další inspektor. Když nyní klikáme na uzly, v obou inspektorech se zobrazují stejné informace, což nám moc nepomůže. V každém inspektor okně nahoře je ale tlačítko Attach to Selection, které umožní asociaci daného inspektora s aktuálně vybranými uzly. Opětovným kliknutím na tlačítko (tentokrát se jmenuje Attached) tuto vazbu

zrušíme. Takto je možné si přiřadit inspektor uzlu nebo sadě uzlů, se kterými pracujeme intenzivně a ostatní uzly procházíme ve druhém inspektoru.

Spreads

Nástroj, který umožňuje pracovat s několika instancemi dat. *Spread* není ve skutečnosti nic jiného, než seznam obsahující libovolná data. Jeden záznam v tomto seznamu je označován jako *Slice*.

Každý pin daného uzlu nemusí nutně obsahovat hodnotu pouze jedné instance dat, ke kterým se vztahuje. Potenciálně může obsahovat celý „spread“, tedy seznam dat. Piny tedy mohou obsahovat tzv. ***slices***, které tato data obsahují. V inspektoru je objevíme mezi dvěma hlavními sloupci – levý ukazuje hodnoty v pinech a pravý jejich jména. Úzký šedý pás mezi těmito sloupci na některých místech obsahuje značku >, která říká, že daný pin může potenciálně obsahovat více slices najednou. Pokud je tato značka >>, znamená to, že pin již více slices obsahuje. Kliknutím na značku je možné si prohlédnout jednotlivé hodnoty ve všech slices.

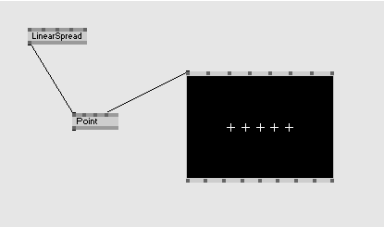
Generování spreads

Spreads je možné vytvořit například pomocí vložení uzlu ***LinearSpread(Spreads)***. Při vložení se nic speciálního nestane. K tomu dojde, až když změníme hodnotu pinu *SpreadCount* např. na hodnotu 5. Na první pohled se nic nestalo, ale pokud klikneme na >> v inspektoru u pinu Output, tak se rozbalí všech 5 vytvořených slices.

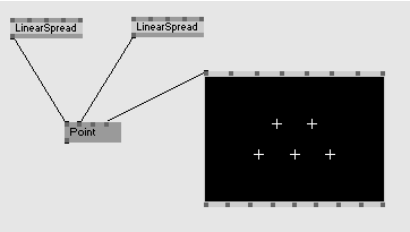
Existují i další uzly, které dokáží generovat spreads: ***CircularSpread(Spreads)***, ***RandomSpread(Spreads), I(Spreads)***.

K čemu to je?

Použití spreads si ukážeme na následujícím příkladě. Do scény si vložíme kromě ***LinearSpread(Spreads)*** uzly ***Renderer(GDI)*** a ***Point(GDI)***. Poté spojíme output pin uzlu Point s input pinem uzlu Renderer. Výsledkem je jedno + vykresleno ve středu renderovací obrazovky. Nyní si na Point napojíme LinearSpread, který má hodnotu SpreadCount nastavenu na 5. Dojde k tomu, že znak + bude vykreslen 5krát, protože renderer dostal 5 souřadnic, na které má + vykreslit.



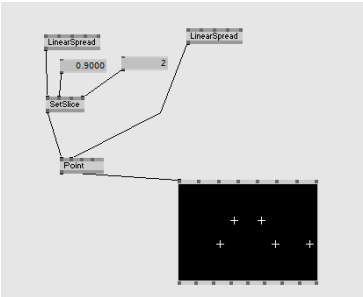
Uzel Point se sám vypořádal s poskytnutým spread. Jak se ale Point vypořádá například se situací, kdy mu přidáme do input pinu Y další spread, který bude mít nastavenu SpreadCount na 2? Výsledek bude:



V tomto případě Point obdržel 5 hodnot pro input X a 2 hodnoty pro input Y. Když vykresluje první +, vezme první slice pinu X a první slice pinu Y. U druhého + vezme druhý slice pro X a druhý slice pro Y. U třetího + vezme třetí slice pro X, ale pro Y už třetí slice neexistuje, proto vezme znovu první slice pro Y, atd.

Takovýmto způsobem zpracovává spreads libovolný uzel, který je podporuje. Zjistit, zda daný uzel podporuje spreads, lze v inspektoru – pokud obsahuje znak >, pak podporuje spreads.

Kromě automatického vypořádání se se slices je možné přistupovat k slices jednotlivě. K tomu slouží uzel ***SetSlice(Spreads)***, jehož nejpravější input pin slouží k tomu, abychom řekli, který slice chceme modifikovat. Slices jsou indexovány od nuly, takže pokud chceme změnit např. 3. slice, musíme zadat hodnotu 2. Druhý pin zleva slouží pro definici nové hodnoty pro daný slice.



Další možností je využití ***I(Spreads)*** je možné modifikovat nejen jeden slice, ale slices v určitém rozsahu.

IOBox

Na začátku tutoriálu jsme použili uzel ***IOBox(Value Advanced)***. VVVV obsahuje IOBoxy i pro další datové typy.

IOBox(String)

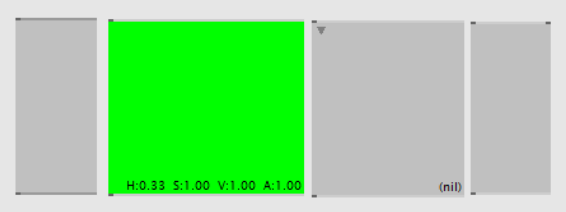
IOBox(Color)

IOBox(Enumerations)

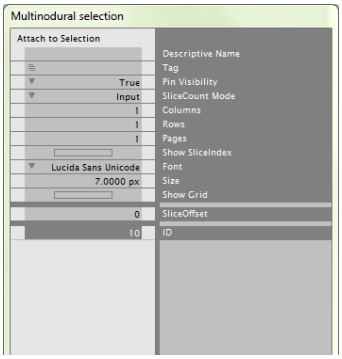
IOBox(Node)

Tyto uzly mohou sloužit jako vstupní místa do běžícího programu nebo jako výstupní místa z programu.

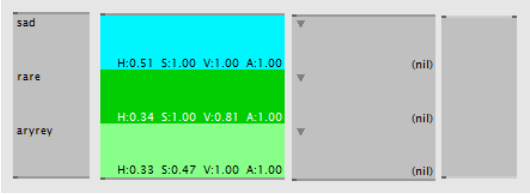
Zkusíme si všechny tyto typy IOBoxů vložit do scény a ukážeme si na nich resize všech uzlů najednou. K tomu stačí všechny uzly označit a najet na pravý dolní roh jednoho z uzlu a táhnout.



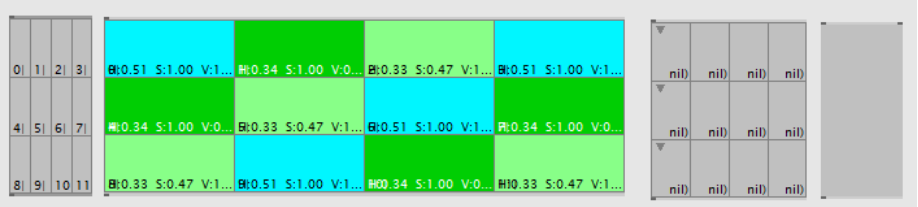
Kromě IOBox(Node), který je trochu speciální, mají všechny IOBoxy podobné vlastnosti. Vybereme všechny uzly kromě IOBox(Node) a podíváme se do inspektoru.



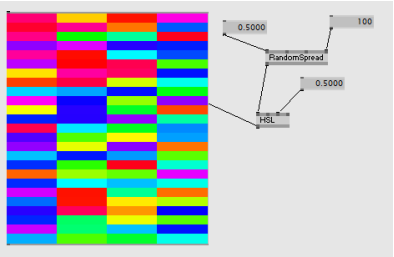
V inspektoru můžeme změnit font a velikost písma, ale důležitější a zajímavější je možnost změny pinů Columns, Rows a Pages. Pokud v tomto defaultním nastavení chceme mít například 3 řady (tzn., že potřebujeme 3 slices), tak musíme změnit SliceCount Mode na ColsRowsPages. Pak můžeme nastavit počet řad na 3 a každou řadu modifikovat zvlášť.



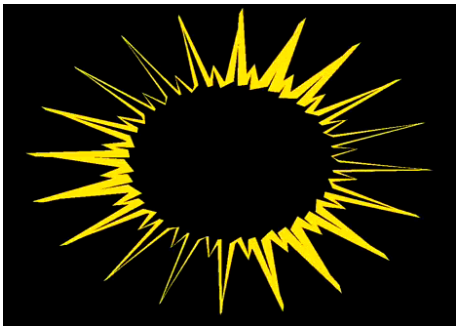
Poté můžeme zapnout Show SliceIndex a ShowGrid a přidat další sloupce:



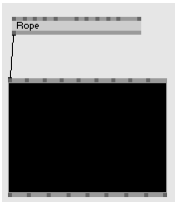
Nyní vytvoříme IOBox(Color), který na vstupu obdrží 100 slices náhodných barev, které zobrazí ve 4 sloupcích a 25ti řádcích. Toho dosáhneme vložením uzlů RandomSpread(Spreads), HSL(Color Join) a několika IOBox(Value Advanced). Pro IOBox nastavíme v inspektoru počet řádků a sloupců a vypneme Show Value.



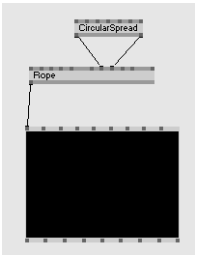
Ukázka:



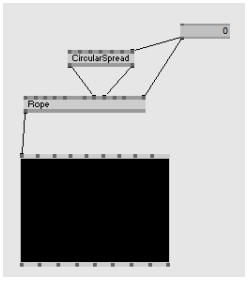
Nejdříve vložíme do scény uzel ***Renderer(EX9)*** a pomocí ALT+2 jej vložíme do scény. Potom vložíme objekt ***Rope(DX9)*** a spojíme jej s rendererem.



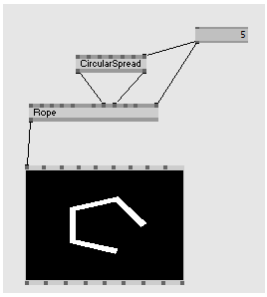
Teď docílíme toho, že objekt Rope, který je zatím neviditelný, dostane tvar kružnice. K tomu vložíme do scény objekt ***CircularSpread(Spreads)*** a spojíme jeho output pin X s input pinem X uzlu Rope. Totéž pro Y.



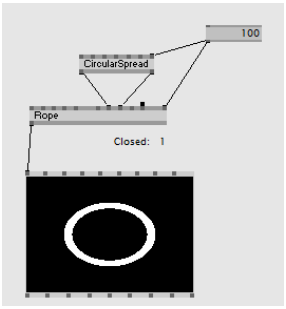
Nyní vložíme uzel ***IOBox(Value Advanced)*** a jeho output pin napojíme na input pin Spread Count uzlu CircularSpread a na input pin Resolution uzlu Rope.



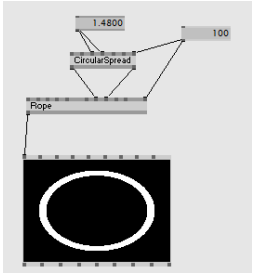
Když teď v IOBoxu zvětšíme hodnotu například na 5, tak uvidíme v renderovacím okně objekt o pěti uzlech, které odpovídají jednotlivým slices objektu CircularSpread.



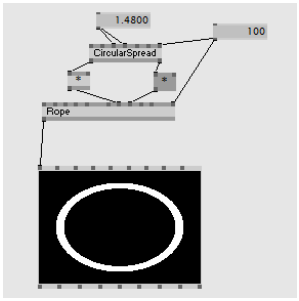
Když tuto hodnotu nyní zvětšíme např. na 100, dostaneme neuzavřenou kružnici. Uzavření dosáhneme nastavením hodnoty pinu Closed u uzlu Rope na 1.



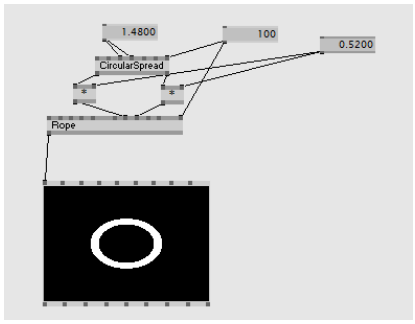
Nyní vložíme další uzel ***IOBox(Value Advanced)***, který bude ovlivňovat výšku a šířku kružnice. Proto jeho output pin propojíme s input piny Width a Height uzlu CircularSpread. Změnou hodnoty v tomto IOBoxu pak můžeme měnit velikost kružnice.



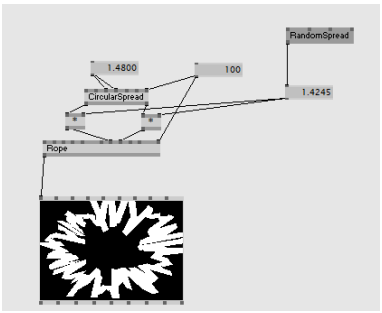
Velikost kružnice můžeme ovlivňovat i jiným způsobem. Pokud vynásobíme output hodnoty uzlu CircularSpread nějakou hodnotou, opět dojde ke změně velikosti uzlu. K tomu vložíme do scény dva uzly pro násobení, ****(Value)***. Pomocí těchto uzlů propojíme výstupní piny CircularSpread se vstupními piny pro X a Y uzlu Rope.



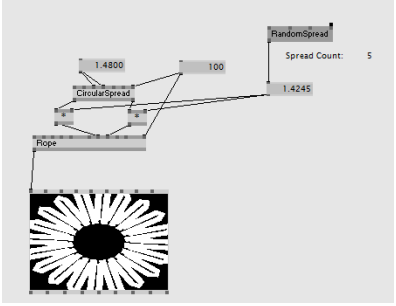
Nyní budeme potřebovat další uzel ***IOBox(Value Advanced)***, který propojíme s uzly pro násobení. Změnou hodnoty v tomto naposledy vloženém uzlu opět můžeme měnit velikost kružnice.



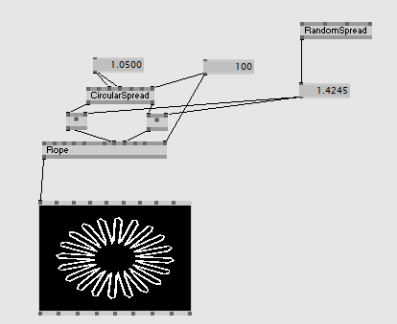
Nyní je velikost kružnice počítána jako kombinace dvou hodnot, které jsou uloženy ve dvou IOBoxech (na obrázku ty s hodnotami 1.48 a 0.52). Pro náhodné generování hodnot poloměru kružnice můžeme přidat do scény objekt ***RandomSpread(Spreads)***, který spojíme IOBoxem propojeným s uzly pro násobení. Hodnoty v input pinech tohoto random spread budou: Input = 1, Width = 1 a Spread Count nastavíme např. na 100. Výsledek je:



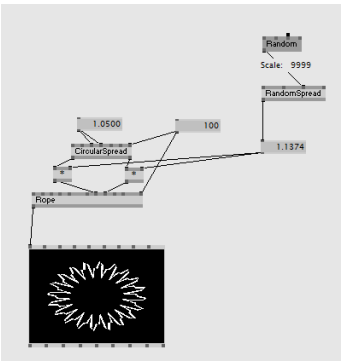
Došlo k vynásobení sta náhodných hodnot s hodnotami z CircularSpread a výsledek byl aplikován na X a Y hodnoty v uzlu Rope. Pokud nyní chceme dosáhnout pravidelnějšího tvaru rope, snížíme hodnotu Spread Count v input pinu uzlu RandomSpread – např. na 5.



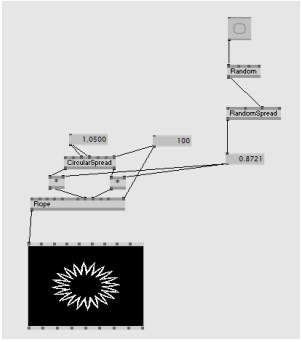
Poté zmenšíme tloušťku čáry – změnou hodnoty v input pinu Thickness uzlu Rope, případně si můžeme opět pohrát s velikostí kružnice změnou hodnoty v IOBoxu ovlivňujícím input piny X a Y uzlu CircularSpread.



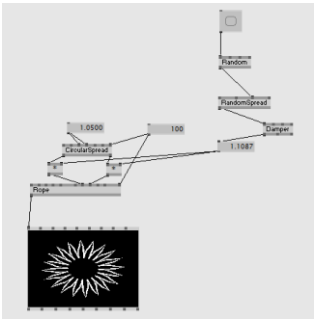
Změna tvaru rope se nyní dá dosáhnout změnou hodnoty v input pinu Seed uzlu RandomSpread. Nyní, s každou změnou hodnoty v Seed, změní se tvar rope. Pokusíme se změnu hodnoty Seed zautomatizovat. Opět chceme, aby se hodnota pro Seed generovala náhodně. K tomu využijeme další uzel – do scény vložíme uzel ***Random(Value)***. Jeho první input pin – Enabled – ovlivňuje, zda se aktuálně generují náhodná čísla. Dále nastavíme hodnotu v input pinu Is Integer na 1, protože naše Seed hodnoty jsou také typu integer. Hodnota v pinu Scale je defaultně 1, ale u RandomSpread uzlu, na který budeme tento uzel napojovat, může odpovídající pin Random Seed nabývat libovolné hodnoty. Proto do Scale pinu nastavíme vysokou hodnotu, řekněme 9999. Nyní uzly konečně propojíme. Můžeme si všimnout, že když není měním hodnotu Enabled v Random(Value) uzlu, dochází k automatickému překreslování tvaru rope.



Nyní si vytvoříme nový uzel, který vložíme dvojklikem pravým tlačítkem myši a zvolíme Bang. S ním spojíme pin Enabled uzlu Random(Value). Nyní pokaždé, když klikneme pravým tlačítkem myši do uzlu Bang, dojde k jednorázovému překreslení tvaru rope.

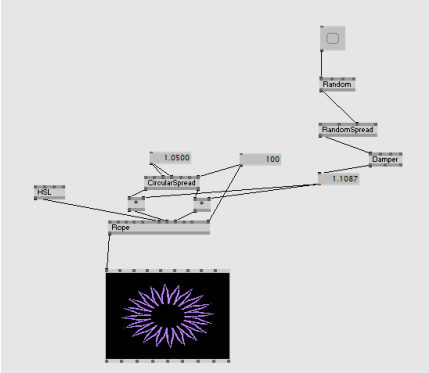


Nyní bychom mohli chtít, aby přechod mezi jednotlivými tvary po kliknutí na Bang byl plynulejší. K tomu využijeme tzv. **filtr**. Ten je reprezentován dalším uzlem – ***Damper(Animation)***. Tento uzel bude filtrovat změnu pozice a bude propojovat uzly RandomSpread a IOBox dávající vstupy do uzlu pro násobení.

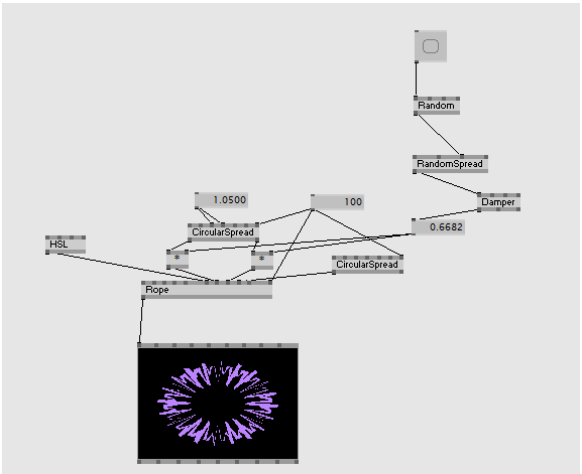


Když teď klikneme pravým tlačítkem na Bang, dojde k plynulému přechodu na jiný tvar.

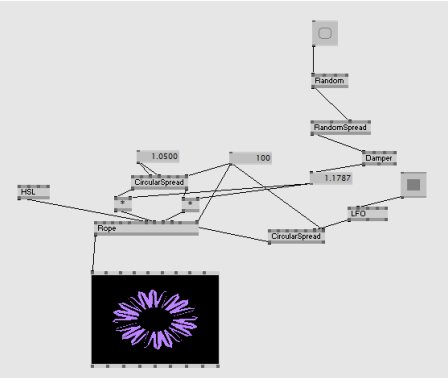
Nastal čas, kdy rope obarvíme. K tomu vložíme uzel ***HSL(Color Join)*** a spojíme jej s input pinem pro barvu uzlu Rope. Nyní při změně hodnot v tomto uzlu dochází ke změně barvy rope.



Dále můžeme ovlivňovat například tloušťku rope. K tomu vložíme uzel ***CircularSpread(Spreads)*** a spojíme jej s input pinem Thickness uzlu Rope. Čára rope je najednou příliš tlustá a proto změníme hodnotu v pinu Width na např. 0.06. Hodnotu v pinu Factor nastavíme na 3 a změníme hodnotu Spread Count na 100 – propojením s IOBoxem obsahujícím tuto hodnotu.



Nakonec vložíme uzel ***LFO(Animation)***, čímž zkusíme dosáhnout toho efektu, že tloušťka bude po rope jakoby rotovat. Vložíme si ještě uzel ***ToggleButton*** (dvojklikem pravého tlačítka myši) a propojíme jej s LFO uzlem – pinem Pause. Output pin uzlu LFO propojíme s pinem Phase uzlu CircularSpread.



Nyní si můžeme zapnout Toggle tlačítko a pomocí tlačítka Bang měnit tvar.

Celý tutoriál zde:

http://www.youtube.com/watch?v=kp32U_Ov_tM&list=PLE5653092A420C4A6&index=27

Pokud bychom chtěli náš výsledek napojit na zvukovou stopu, lze toho dosáhnout jednoduše:

https://www.youtube.com/watch?v=OyBrvr4pYhE