

VST Plugin: Phaser Dokumentation

Lukas Zembrot: 2170344

Alexander Menk: 2170471



Einleitung

Da dieses Projekt fast ausschließlich mit der Berechnung des Filters durch gezieltes Einsetzen, sowie der Erstellung des User Interfaces zusammenhängt und weniger mit der Programmierung, haben wir uns entschieden uns auf diese beiden Themengebiete zu beschränken.

User Interface

Da Effektgeräte für Instrumente weit verbreitet sind, haben wir uns entschieden die entsprechende Interface-Metapher zu verwenden. Das Plugin ist aufgebaut aus einem blauen Pedal-artigen Gehäuse, welches Knöpfe, Regler und ein Display besitzt. Um einen eher plastischen Look zu erreichen wurde deswegen mit Licht versucht ein dreidimensionales Aussehen zu verleihen, ohne die Lesbarkeit des Plugins zu stören. Aus diesem Grund verwenden wir eine 3D Software für die Erstellung des Hintergrunds.

Workflow

Die Arbeit an der GUI lässt sich grob in vier Phasen aufteilen. Zuerst, größtenteils noch während der Projektfindung, wurde das optische Konzept festgelegt(siehe Gruppenkonzept). Die wichtigsten Merkmale waren die vorher abgesprochenen Merkmale, sowie die Farbe Blau(Himmel), die Assoziationen von digital und Freiheit, einem großen Raum weckt. Als Material sollte das Pedal eine typische Lackierung mit stark reflektierenden Elementen im Lack bekommen. Erste Experimente bestanden deswegen aus einigen Material-studies.

Daraufhin konnte die zweite Phase, das Modellieren und Textuieren des 3D Objektes im Hintergrund starten(Siehe Abbildung 2.). Die genaue Ausleuchtung war bis zum Schluss der schwierige Teil, da ein guter Specular-(Glanz) Effekt nicht kommen wollte. Erst eine Environment-Map(zu sehen in Abbildung 1) als Hintergrund führte zum Effekt. Um das Bild für das Plugin fertig zu machen wurde der Hintergrund unsichtbar gemacht und taucht nur als Reflektion auf. In Phase 3 wurden die Knöpfe mit dem Programm Knobman erstellt.

Knobman erlaubt es die Knöpfe als Serie



Abbildung 1: Material Study

von maskier- und damit Drehbaren Objekten zu realisieren und diese als Sprites zu exportieren (zu finden in /resources). Zum Schluss wurden der Render noch nachbearbeitet und unter anderem mit Schrift versehen (Fonts ebenfalls in /resources zu finden).

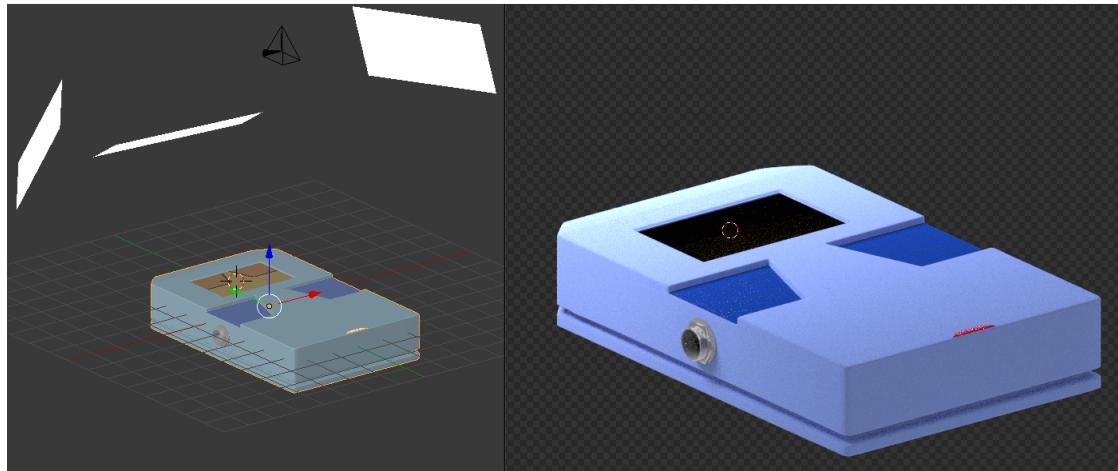


Abbildung 2: Ansicht in Blender

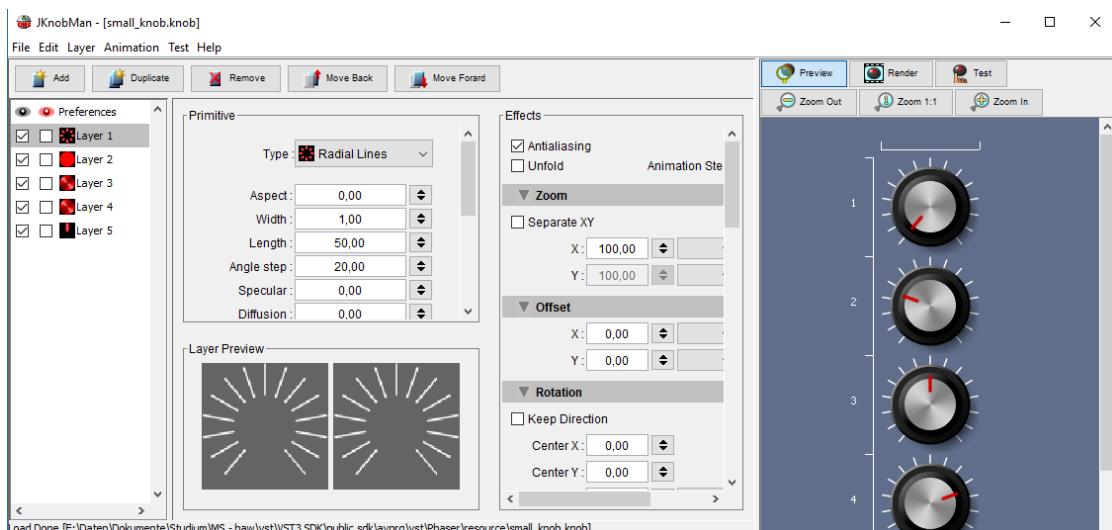


Abbildung 3: Knobman

In VST GUI werden nun die einzelnen Teile zusammengefügt. Mehrere Abstürze, sowie Probleme mit der Fenstergröße führten dazu, dass ein Teil der Bearbeitungen in der plugin.uidesc stattfanden, wobei es sich um eine XML-Datei handelt. Besonders ist, dass die Knöpfe eine Folge von Bildern sind, bei der eine Grundhöhe n angegeben wird, welche die Maske zum Anzeigen des Knopfes im Programm einfach nur um n

Höhen weiter bewegt, je nachdem welcher Wert angezeigt wird. Dabei gab es ebenfalls Probleme mit vstgui, da aus der Dokumentation nicht ersichtlich war, wie die Sprites angeordnet werden sollten oder, dass die Sprites nicht Programm-intern zugeschnitten werden dürfen.

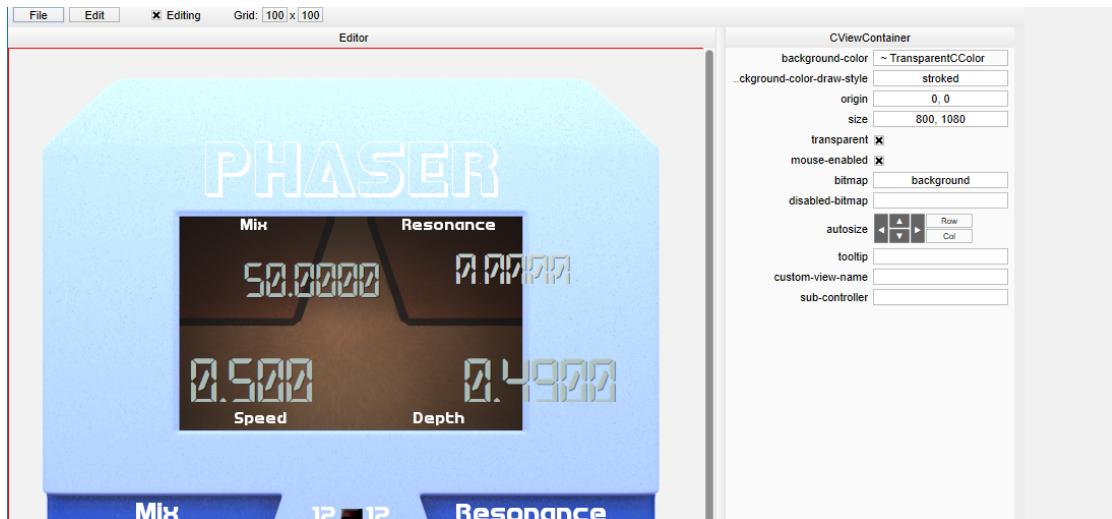


Abbildung 4: VSTEeditor

Schnittstelle zwischen GUI und Backend

Damit zwischen der VST-GUI und dem Backend, der *PluginProcessor* Klasse kommuniziert werden kann, bedient sich die *VSTGUI* keinen Events, wie sie in Java-Swing oder JavaScript zu finden sind, sondern bedarf einfacher Abfragen, ähnlich vieler C Bibliotheken. Während jedes Ticks muss abgefragt werden, ob sich ein Wert in der GUI verändert hat, und wenn ja, was damit passieren soll.

Dieses passiert mit der *hasInputParameterChanged(data, tagId)* Methode, welche über die *PluginAdapter* Klasse vererbt wird. Der Wert kann nun als Prozent([0,1] - *getInputParameterChange(data, tagId)*) oder als direkter Wert(*getPlainValue(tagId)*), festgelegt über die *Parameters.addParameter* Funktion(z.B [0,4], oder eine Serie von Strings). Die Zuweisung welcher Wert welcher Klasse entspricht, läuft über eine Indexierung mit sogenannten Tags. Parameter *n* wird mittels Enum zur einfachen Identifizierung einem Element in der GUI(im vstEditor unter tags) zugewiesen. Dabei war zu lernen, dass Einstellungen der einzelnen Klassen im vstEditor durch die *PluginAdapter* Klasse nicht beachtet werden können.

Allpassfilter und Phaser Effekt

Allpassfilter

Ein Allpassfilter ist ein Filter der keine Frequenzen auslöscht oder verstärkt, sondern möglichst gleichmäßig ausgibt und nur variabel die Phase verschiebt(also die Zeit in der das Signal bei gleicher Periode ihre Nullstellen hat). Damit kann ein Allpassfilter auch über Zeit überdurchschnittlich stark ausschlagende Frequenzen(Peaks) auslöschen, also den Dynamikbereich verkleinern. Durch den Phase-verschiebenden Effekt ist ein Allpassfilter eine Möglichkeit um einen Phaser oder Flanger zu implementieren.

Um den Filter zu finden, wurden die bekannten Filter in ein *Microsoft-Excel* Dokument eingetragen (siehe Abbildung 5, oben), so dass Testwerte (unten links) automatisch berechnet und als Graf angezeigt werden(rechts). Auf dieser Basis wurde durch intelligentes Einsetzen die Werte solange verändert bis sie den Eigenschaften des Gesuchten Filters entsprechen.

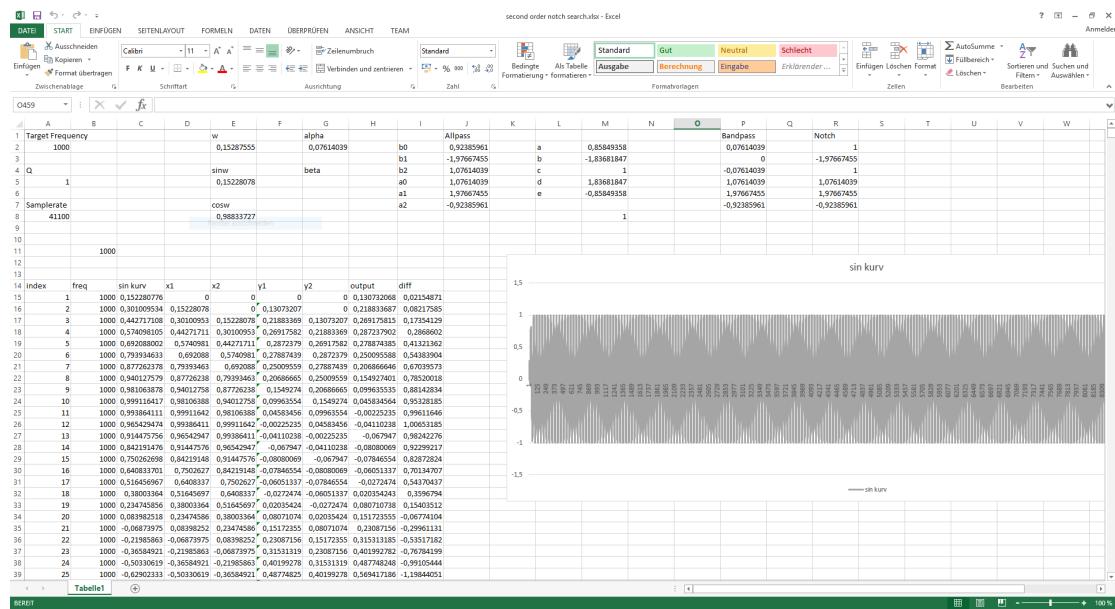


Abbildung 5: Suche nach dem Allpassfilter in Excel

Phasing

Bei einem Phaser wird nun das ursprüngliche Signal mit einer Serie von Allpassfiltern versehen. Diese verschieben die Phase des Signals. Wird nun das Ursprungssignal auf die gefilterte Spur addiert, löschen sich die Phasenverschobenen Teile des Signals mit den Ursprünglichen aus. Die auszulöschenden Teile des Signals werden durch eine Modulation durch einen Low Frequency Oscillator erzeugt. Die Stärke des Effekts wird durch mehrere Parameter versehen, die Breite des auszulöschenden Bereichs, die Mischung der Signale (die stärkste klangliche Veränderung findet bei einer Mischung von 50% der Signale statt), sowie die Anzahl der Filter und natürlich die Schwingung des Effektes, die auf dem Gerät Speed heißt. Ein Resonanzwert dient zur weiteren Verfremdung des Signals, der in unserer Implementation eher ein Reverb, als ein Resonance darstellt.