27. bis 29. März 2014, Zürich & Lausanne



WORKSHOP «GENERATIVE VISUAL TOOLS FOR MUSICIANS AND VISUALISTS»

präsentiert von videokultur.ch

Konzept und Realisation

MIGROS kulturprozent

CREATIVE

PROGRAMMING TOOLKITS

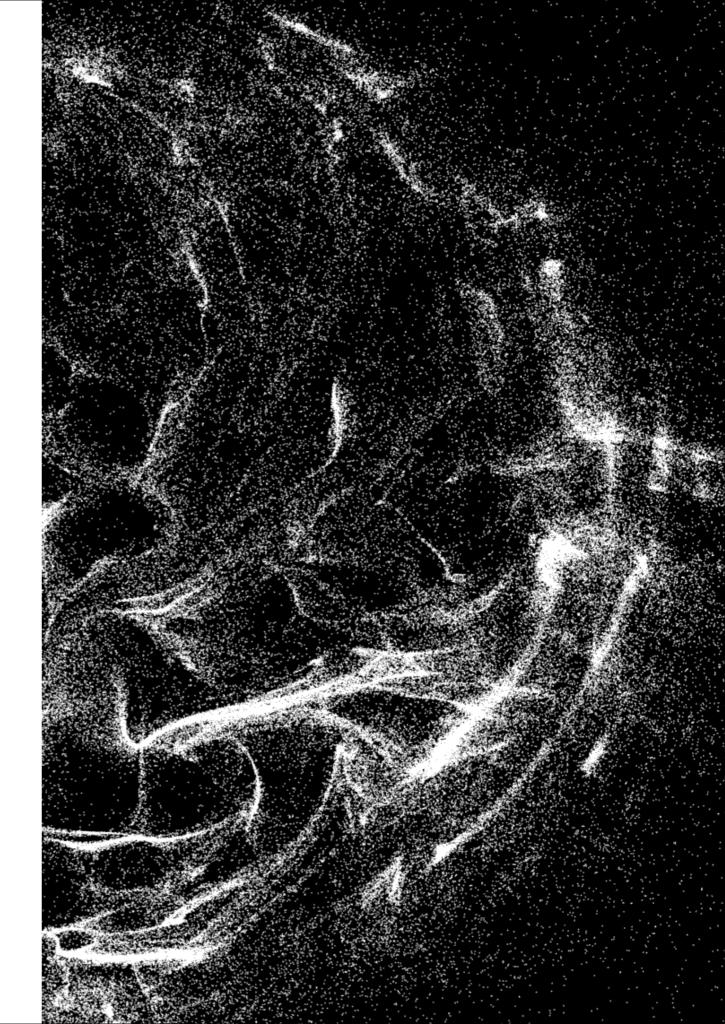
Unter Creative Programming Toolkits verstehen wir Software-Teile welche uns helfen vielfältige Medien-kunst zu erstellen.

Viele dieser Werkzeuge wurden durch Künstler für Künstler erstellt und die bekanntesten sind unter einer open-source Lizenz veröffentlicht.

Solche Werkzeuge bieten grundlegend eine höhere Abstraktionsebene für den Zugriff auf Hardware-nahe Komponenten.

Creative Programming tools werden nicht nur von Künstlern verwendet, sondern auch von Wissenschaftlern und Studenten. Sie können sehr schnell Prototypen erstellen um damit ideen zu visualisieren oder zu entwickeln.

Um creative Programming tools sind meist ganze Ökosysteme entstand, das heisst es findet ein reger Austausch zwischen den Benutzern statt, Benutzer stellen eigene Erweiterungen und Vereinfachungen zur Verfügung, erstellen Tutorials uns Screencasts, helfen in Diskussionsforen, oder nehmen sogar aktiv an der Entwicklung des frameworks selbst teil. es gibt auch solche die weitergehende komplexere Applikation komplett auf einem solchen framework erstellen und weiter verbreiten, zum teil sogar wieder mit eigenen kleinen Ökosystemen.





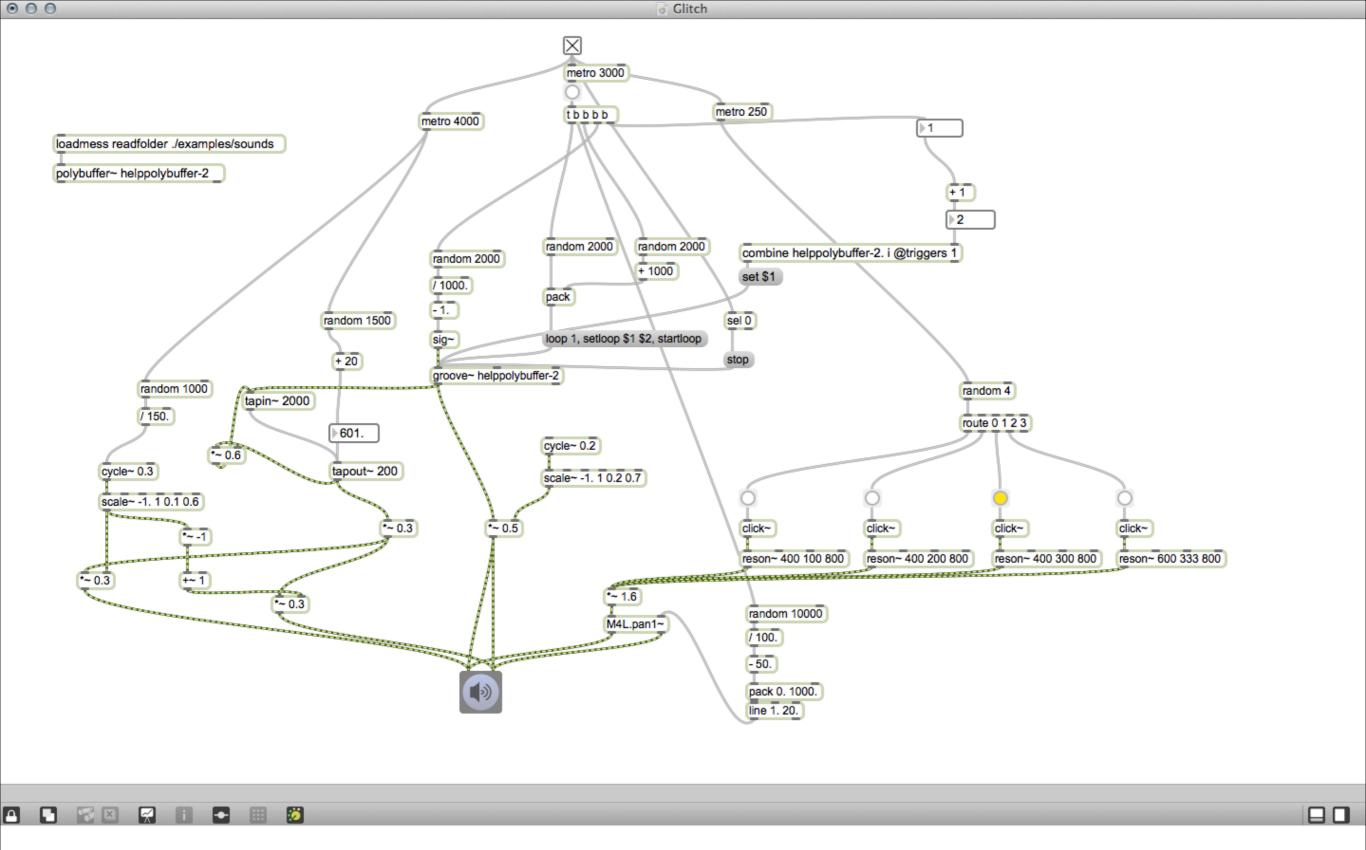
"Creative-Coding" ist eine Art von Computer-Programmierung, in dem das Ziel ist, etwas ausdrucksvoll anstatt etwas funktionellen erstellen. Es wird verwendet, um Live-Visuals zu erstellen, sowie Erstellung visueller Kunst und Design, Kunstinstallationen, Projektionen und Mapping Projektion, Klangkunst, Werbung, Produkt-Prototypen, und vieles mehr.

```
HUP! 0x7fb97dd9b8d0
                                                                                     #include <stdlib.h>
HUP! 0x7fb97dd9b920
HUP! 0x7fb97dd9b920
                                                                                     *define pi 3,1415926
                                                                                     *define sr 48000.0
                                     make
                                                                                     typedef double R:
nothing added to commit but untracked files present (use "git add" to track)
                                                                                     typedef long int Z:
go.c: In function 'go':
90.C:
                                                                                     *define COUNT 16
make: *** [go.so] Error 1
diff -git a/src/go.c b/src/go.c
                                                                                     tupedef struct {
index 0498e27.,b03b737 100644
                                                                                       R time;
    a/src/go.c
                                                                                       R oscs[COUNT];
+++ b/src/go.c
                                                                                       R mods[COUNT]:
@@ -24,7 +24,7 @@ typedef struct {
                                                                                       R dlts[COUNT];
                                                                                       R plck[COUNT];
float go(S *s, float x) {
   Z \text{ tick} = 0;
  if (s->tempo < 0.0001) {
   s->time += 1/sr;
   if (s->time > s->tempo) {
     s->time = 0;
                                                                                     float go(S *s, float x) {
[session-2012-09-29-145438 fd5b207] go 2012-09-29T16;22;56+0100
                                                                                       Z tick = 0:
1 file changed, 1 insertion(+), 1 deletion(-)
                                                                                       if (s->tempo < 0.0001) { exit(0); }
diff -git a/src/go.c b/src/go.c
                                                                                       s->time += 1/sr;
index b03b737..d437880 100644
                                                                                       if (s->time > s->tempo) {
   a/src/go.c
                                                                                         s->time = 0;
+++ b/src/go.c
                                                                                         tick = 1:
@@ -29,7 +29,7 @@ float go(S *s, float x) {
                                                                                         s->tempo *= 0.99:
   if (s->time > s->tempo) {
     s->time = 0;
                                                                                       R \cos = 0;
     tick = 1;
                                                                                       for (Z i = 0; i < COUNT; ++i) {
                                                                                         if (s->time == 0) {
     s->tempo *= 0.99:
                                                                                           s-dlts[i] = rand()/(R)RAND_MAX;
                                                                                           s\rightarrow dcay[i] = exp(-0.125 * pow(rand()/(R)RAND_MAX, 4));
   R \cos = 0;
                                                                                           s\rightarrow rnge[i] = rand() % 512;
```

"Live-Coding" Live coding (sometimes referred to as 'on-the-fly programming', 'just in time programming') is a programming practice centred upon the use of improvised interactive programming. Live coding is often used to create sound and image based digital media, and is particularly prevalent in computer music, combining algorithmic composition with improvisation. Typically, the process of writing is made visible by projecting the computer screen in the audience space, with ways of visualising the code an area of active research. There are also approaches to human live coding in improvised dance. Live coding techniques are also employed outside of performance, such as in producing sound for film or audio/visual work for interactive art installations.

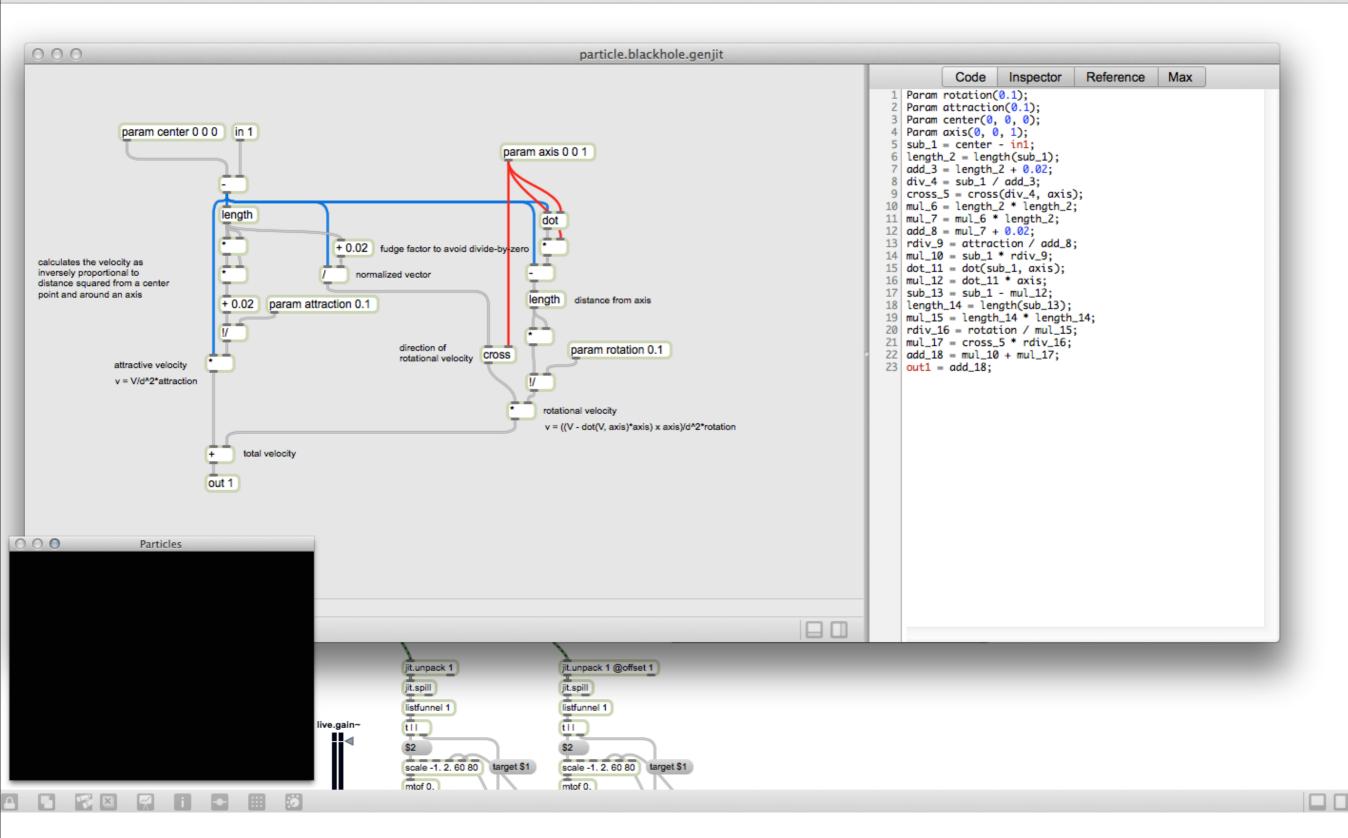
TOOLS

ZWEI WICHTIGE UNTERSCHIEDE



TOOLS BASIEREND AUF **NODES** (PATCHES)

GRAPHISCHE ENTWICKLUNGSUMGEBUNG



"PATCHES" Auf Node basierten Tools entwickelte Anwendungen werden gewöhnlich als Patches bezeichnet. Patches bestehen aus einem Netzwerk von Knoten, die einfache oder komplexe Operationen durchführen. Patches können erstellt, bearbeitet und neu verbunden werden, während sie ausgeführt werden.

M_2_3_01 | Processing 2.0.3

```
void draw() {
if (savePDF) beginRecord(PDF, timestamp()+".pdf");
 background(255);
 strokeWeight(1);
 noFill();
 translate(0, height/2);
 // draw oscillator with freq and phi
 if (drawFrequency) {
   stroke(0, 130, 164);
   beginShape();
   for (int i=0; i<=pointCount; i++) {</pre>
     angle = map(i, 0,pointCount, 0,TWO_PI);
     y = sin(angle * freq + radians(phi));
     y = y * (height/4);
     vertex(i, y);
   endShape();
 // draw oscillator with modFreq
 if (drawModulation) {
   stroke(0, 130, 164, 128);
   beginShape();
   for (int i=0; i<=pointCount; i++) {
     angle = map(i, 0,pointCount, 0,TWO_PI);
     y = \cos(\text{angle} * \text{modFreq});
     y = y * (height/4);
     vertex(i, y);
   endShape();
 // draw both combined
 stroke(0);
```

8

strokeWeight(2);
beginShape();

y = info * carrier;

for (int i=0; i<=pointCount; i++) {
 angle = map(i, 0,pointCount, 0,TWO_PI);</pre>

float info = sin(angle * freq + radians(phi));

float carrier = cos(angle * modFreq);

000

TOOLS BASIEREND AUF CODE (TEXT)

TEXT ENTWICKLUNGSUMGEBUNG

NODES

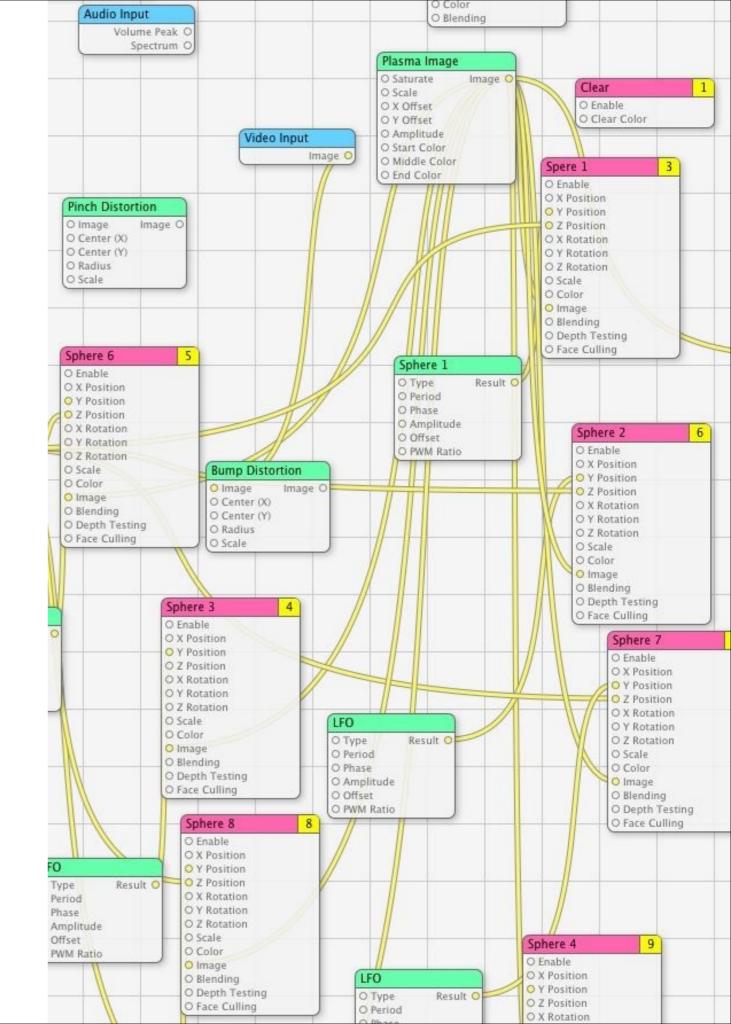
QUARTZ COMPOSER - MAX MSP - WWV - VUO, PRAXIS LIVE, PURE DATA

QUARTZ COMPOSER

Quartz Composer ist ein Programm von Apple, mit dem man sogenannte Patches entwickeln kann, die dann von der Quartz Grafik-Engine ausgeführt werden.

Quartz ist die Grafikschicht des Apple Betriebssystems Mac OS X. Es handelt es sich um eine Bibliothek für 2D- und 3D-Grafiken, die das grundlegende Darstellungsmodell für Mac OS X bildet.

Plattform: Max OS X

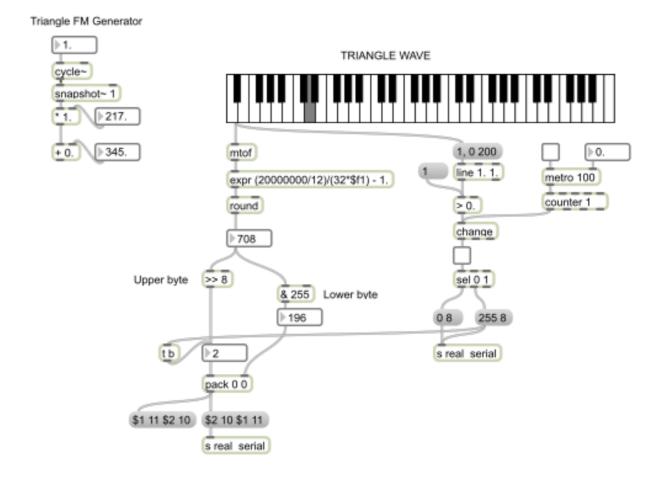


MAX | MSP | JITTER

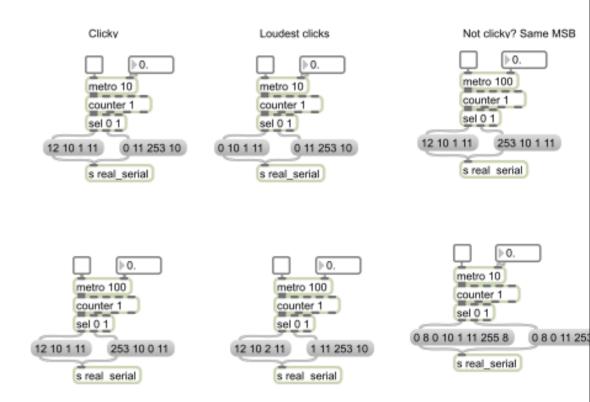
Max | MSP ist eine graphische
Entwicklungsumgebung für Musik und
Multimedia von Cycling'74, die für
Echtzeitprozesse ausgelegt ist. Sie
wird seit 20 Jahren von Komponisten,
Musikern, Softwareentwicklern und
Künstlern eingesetzt, um interaktive
Software selbst zu erstellen –
unabhängig von den ästhetischen
Vorgaben kommerzieller Produkte.

Insbesondere im **Live-Betrieb** ist diese Sprache in der zeitgenössischen **elektronischen Musik**, im Bereich des **Live-Video** und unter Laptop-Künstlern weit verbreitet.

Plattform: Max OS X, Windows



TRIANGLE WAVE STRESS TESTS

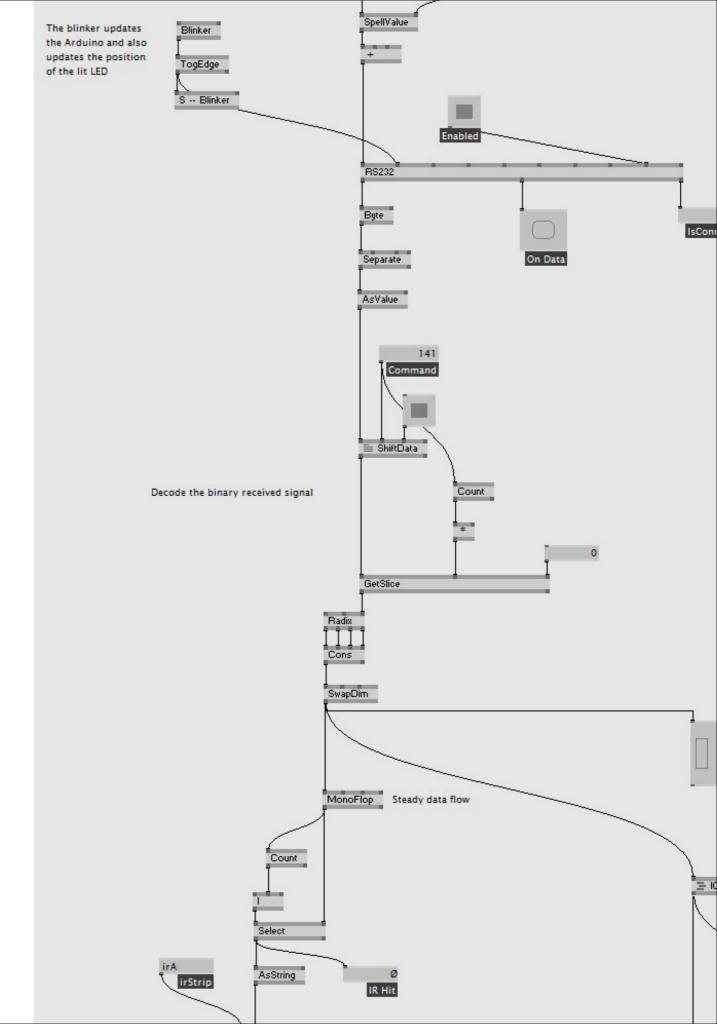


VVVV

VVVV (auch v4 oder v-vier) ist ein Hybrid-visuelle | Text Programmier-Toolkit für einfaches Prototyping von interaktiven, visuellen Installationen und generativen Kunst.

Es skaliert robust auch für große Umgebungen mit physikalischen Schnittstellen, **Echtzeit -Motion-Grafik-, Audio-und Video** und kann mit vielen Benutzern gleichzeitig interagieren.

Plattform: Windows (OS X und Linux angekündet)



CODE

Processing, OpenFrameworks, Cinder, SuperCollider, Polycode, NodeBox

PROCESSING

Processing ist eine Open Source-Programmiersprache für die Programmierung von Bildern, Animation und Sound.

Es wurde speziell für Studenten, Künstler und Designer entwickelt. Durch den einfachen Aufbau eignet sich Processing gut als Einstieg in die Grundstrukturen des Programmierens.

Processing wurde von Künstlern und Designern entwickelt, um eine Alternative zu kommerzieller Software zu bieten.

Processing basiert auf einer Javaähnlichen objektorientierten Programmiersprache.

Plattform: Windows, Max OS X, Linux, IOS, Android, Arduino



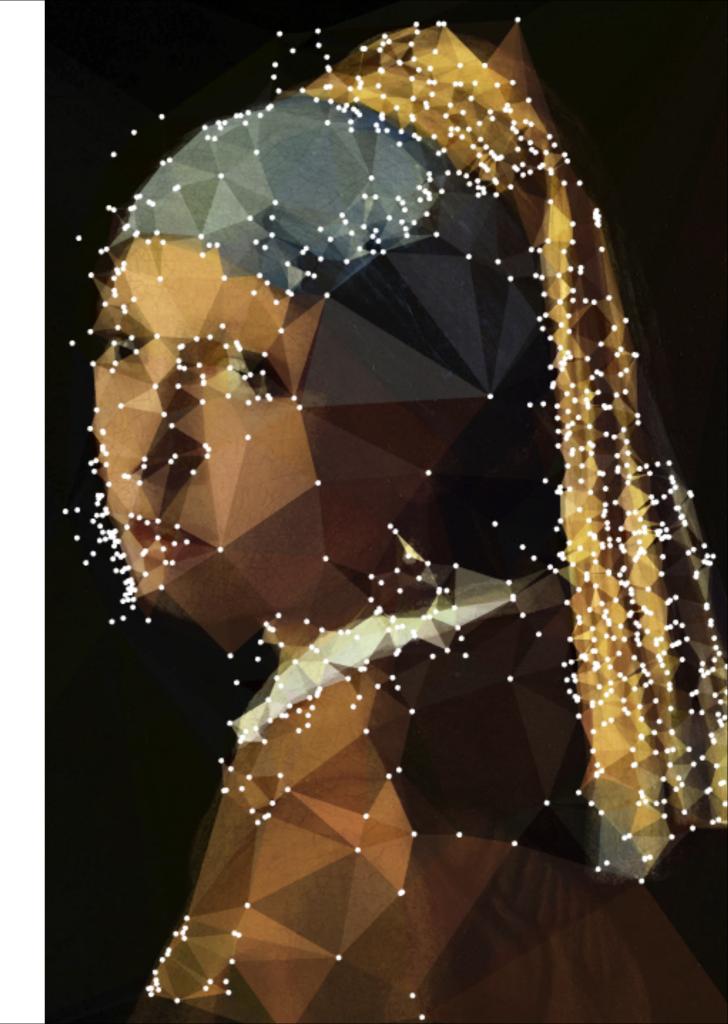
```
void draw() {
 if (savePDF) beginRecord(PDF, timestamp()+".pdf");
 background(255);
 strokeWeight(1);
 noFill();
 translate(0, height/2);
 // draw oscillator with freq and phi
  if (drawFrequency) {
   stroke(0, 130, 164);
   beginShape();
   for (int i=0; i ← pointCount; i++) {
     angle = map(i, 0,pointCount, 0,TWO_PI);
     y = sin(angle * freq + radians(phi));
     y = y * (height/4);
     vertex(i, y);
   endShape();
 // draw oscillator with modFreq
 if (drawModulation) {
   stroke(0, 130, 164, 128);
   beginShape();
   for (int i=0; i<=pointCount; i++) {</pre>
     angle = map(i, 0,pointCount, 0,TWO_PI);
     y = cos(angle * modFreq);
     y = y * (height/4);
     vertex(i, y);
   endShape();
 // draw both combined
 stroke(0);
 strokeWeight(2);
 beginShape();
 for (int i=0; i<=pointCount; i++) {
   angle = map(i, 0,pointCount, 0,TWO_PI);
   float info = sin(angle * freq + radians(phi));
   float carrier = cos(angle * modFreq);
   y = info * carrier;
```

CINDER

Cinder ist eine Programmierbibliothek entworfen um der Sprache C++ erweiterte Visualisierung Fähigkeiten zu geben.

Cinder bietet eine leistungsstarke, intuitiv Toolbox für die Programmierung von Grafik-, Audio-, Video-, Netzwerk-, Bildverarbeitung und Rechengeometrie.

Plattform: Windows, Max OS X, IOS



OPENFRAMEWORKS

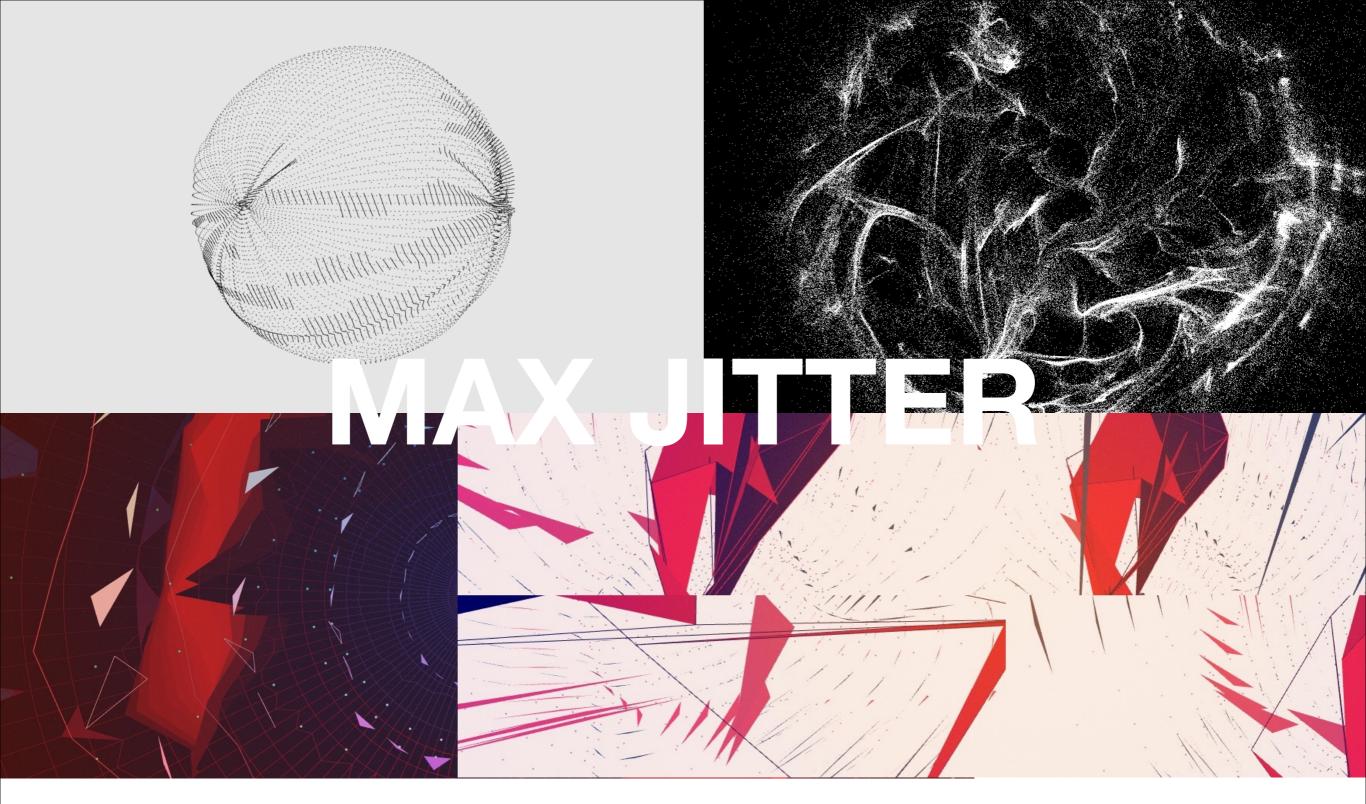
Openframeworks ist ein Open-Source Toolkit konzipiert für "creative coding" und ist in **C++** geschrieben.

Openframeworks bietet eine vereinfachte Schnittstelle zu leistungsfähigeren Bibliotheken für Media Manipulation, Hardware Kontrolle und Netzwerkkommunikation.

Es besteht aus einem Kern von Funktionen für **2D-und 3D-Grafik-, Sound und Videoverarbeitung.**

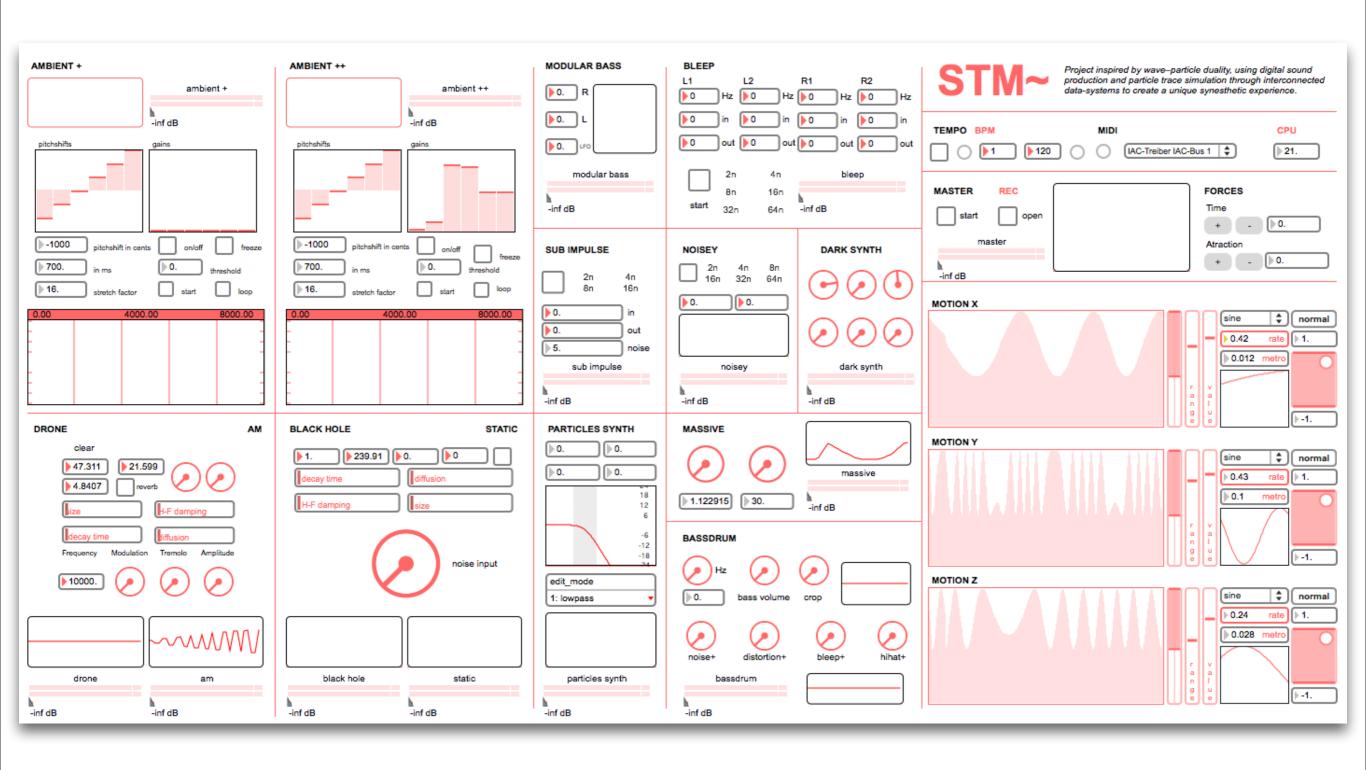
Plattform: Windows, Max OS X, Linux, IOS, Android

```
Classes > 📶 PseudoTerminal.m > 🔟 -insertTab:atInde
            iTerm )
 Find $
3621
3623
3624
     - (void)fitTabToWindow:(PTYTab*)aTab
3625
3626
         NSSize size = [TABVIEW contentRect].size;
3627
         PtyLog(@"fitTabToWindow calling setSize for content size
3628
          [aTab setSize:size];
3629
3630
3631
     - (void)insertTab:(PTYTab*)aTab atIndex:(int)anIndex
3632
3633
         PtyLog(@"insertTab:atIndex:%d", anIndex);
3634
         assert(aTab);
3635
         if ([TABVIEW indexOfTabViewItemWithIdentifier:aTab] == N
3636
              for (PTYSession* aSession in [aTab sessions]) {
3637
                  [aSession setIgnoreResizeNotifications:YES];
3638
3639
              NSTabViewItem* aTabViewItem = [[NSTabViewItem alloc]
A640
              [aTabViewItem setLabel:@""];
3641
3642
              assert(aTabViewItem);
              [aTab setTabViewItem:aTabViewItem];
3643
              PtyLog(@"insertTab:atIndex - calling [TABVIEW insert
3644
              [TABVIEW insertTabViewItem:aTabViewItem atIndex:anIn
3645
              [aTabViewItem release];
3646
              [TABVIEW selectTabViewItemAtIndex:anIndex];
3647
              if ([self windowInited] && !_fullScreen) {
3648
                  [[self window] makeKeyAndOrderFront:self];
3649
3650
              [[iTermController sharedInstance] setCurrentTerminal
3651
3652
3653
3654
       (void)insertSession:(PTYSession *)aSession atIndex:(int)an
3655
3656
         PtyLog(@"%s(%d):-[PseudoTerminal insertSession: 0x%x atI
3657
                 __FILE__, __LINE__, aSession, index);
3658
3659
         if (aSession == nil) {
3660
              return;
3661
3662
3663
         if ([[self allSessions] indexOfObject:aSession] == NSNot
3664
              // create a new tab
3665
              PTYTab* aTab = [[PTYTab alloc] initWithSession:aSess
3666
              [aSession setIgnoreResizeNotifications:YES];
3667
```



Videobearbeitung und 3D-Grafik in Echtzeit

Insbesondere im Live-Betrieb ist diese Sprache in der zeitgenössischen elektronischen Musik, im Bereich des Live-Video und unter Laptop-Künstlern weit verbreitet.



CUSTOM SOFTWARE INTERFACE FOR REAL TIME AUDIO PROCESSING

Max for Live erweitert Ableton Live um die aus Max | MSP bekannten Möglichkeiten um die Erstellung eigener virtueller Instrumente und Effekte.

Eigens kreierte Synthesizer, algorithmische Kompositions-Werkzeuge, direkt Interaktion mit Hardwaregeräten, generierte Live-Visuals, Licht Steuerung – all das ist möglich und nahtlos in Benutzeroberfläche und Workflow von Ableton Live integriert.