Anhang

A Server

A.1 server_config.inc.php

```
<?php // server_config.inc.php

$config[sr]  = 44100; // globale Samplerate
$config[bit] = 16;    // globale Bitrate

// Verzeichnisse (ohne letzten, abschliessenden '/')
$config[rootdir] = ".."; // relativ
$config[bindir]  = "$config[rootdir]/organs";

// Dateiname der temporaer erzeugten Datei.wav
// (zur weiteren Verarbeitung fuer Plugins)
$config[tmpFilename] = "lf_download_tmpfile.wav";


// zulaessiger Mime-Typ, der mittels eines Regulaeren Ausdrucks
// ueberprueft wird
// -> wegen Verwendung in Regex muss '/' mit '\' escaped werden
// -> momentan implementiert: datei.mp3, datei.ogg oder datei.wav

//$config[mimetype] = "(audio|application)\/(x-)?(mpeg|ogg)";
// -> audio/mpeg, audio/x-mpeg, application/mpeg, application/x-mpeg,
//    audio/ogg, audio/x-ogg, ...
$config[mimetype] = "(audio|application)\/(x-)?(mpeg|ogg|wav)";


// Shell-Umgebungs-Variablen setzen (-> fuer csound)
putenv("LD_LIBRARY_PATH=$config[rootdir]/organs/lib");
putenv("OPCDIR=$config[rootdir]/organs/lib/Csound");

// Programm-Pfade
$bin[sndinfo] = "$config[bindir]/csound -U sndinfo";
$bin[pvanal]  = "$config[bindir]/csound -U pvanal";

// csound-Argumente
// -m2 -> Ausgabe-Grad der (Fehler-)Nachrichten
// -d -> grafische Darstellung  der ftable unterdruecken
// -W -> Datei.wav erzeugen
// -A -> Datei.aiff erzeugen
// -o -> Ausgabe-Datei
$bin[csound]  = "$config[bindir]/csound -m2 -d -W -o";
//$bin[csound]  = "$config[bindir]/csound -m2 -d -A -o";

$bin[sox]     = "$config[bindir]/sox";
// -V = verbose; -t auto = autodetect soundfileformat
```

```
//$bin[sox_paraIn] = "-V -t auto";
$bin[sox_paraIn] = "-t auto";
// -r = samplerate; -w = 16bit; -c 1 = mono
$bin[sox_paraMonoOut] = "-r $config[sr] -w -c 1";
$bin[soxmix]  = "$config[bindir]/soxmix";

$bin[lame]    = "$config[bindir]/lame";
$bin[oggdec]  = "$config[bindir]/oggdec";
$bin[oggenc]  = "$config[bindir]/oggenc";
$bin[oggenc_quality]  = "--quality 10"; // 0 (low) bis 10 (high)
$bin[ogginfo] = "$config[bindir]/ogginfo";

$bin[stderr2stdout] = "2>&1"; // Error-Out nach Standard-Out umleiten
$bin[no_stderr] = "2>/dev/null"; // Error-Out nach /dev/null umleiten,
                                 // also unterdruecken/loeschen

// Standard Linux-Tools (fuer die Ermittlung des Mime-Typ's)
//$bin[wget]   = "/usr/bin/wget";
$bin[wget]   = "/usr/bin/wget --tries=3";
//$bin[wget]   = "/usr/bin/wget --tries=3 --user-agent='Mozilla'";
$bin[file]   = "/usr/bin/file";

// momentan unbenutzt -> jetzt mit sox trim/stretch geregelt
$bin[dd]   = "/bin/dd";


// auf Existenz der notwendigen externen Programme pruefen:
foreach($bin as $prog) {
    // falls $prog mit '/', './' oder '../' beginnt,
    // Ausdruck bis zum ersten Whitespace untersuchen
    // -> also evtl. angegebene Argumente ignorieren
    // (sonst ergibt is_file() Fehler)
    if (preg_match("/^(\.*\/\S+)/", $prog, $match)) {
        if (!is_file($match[1])) {
            $errorMsg = "cannot find necessary binary '$prog'";
            exit("ERROR: $errorMsg (defined in config.inc.php)\n");
        }
    }
}


// horizontale Gliederungs-Linie
$config[hr] = "\n==================================";
$config[hr] .= "==================================\n";


// Default-Variablen
// -> werden in $config-Array uebernommen, falls kein entsprechendes
```

```
//    --Argument uebergeben wurde (werden gesetzt, auch wenn das
//    entsprechende --Argument nicht uebergeben werden kann
//    -> s. $options-Array)
$default = array(
    // localhost + andere konfigurierten Adressen (z.B. eth0)
    "host"   => "0.0.0.0",
    //"host"   => "127.0.0.1", // localhost
    "port"   => "12965", // Port: Life -> l=12 i=9 f=6 e=5 --> 12965
    //"tmpdir" => "$config[rootdir]/tmp",
    "tmpdir" => "/tmp",
    "liFedir" => "$config[rootdir]/liFe",

    "maxsize" => "2.0", // [MB]
    "maxlength" => "8.0", // [hh:mm:ss.frac]

    // noch nicht implementiert:
    "logfile" => "$config[rootdir]/liFe.log",
    "email"  => "ruebler@folkwang-hochschule.de"
    );

// verfuegbare Optionen, die als '--Argument Wert' gesetzt
// werden koennen/duerfen
$options = array("host","port","tmpdir","liFedir","show",
                 "maxsize", "maxlength");

// Argumente, mit denen die Hilfe ($Usage) aufgerufen werden kann
$help_options = array("--help", "-help", "-h");


$tryHelp = "Try '$argv[0] --help' for more information.\n";


$deF[host] = "default: $default[host]";
$deF[port] = "default: $default[port]";
$deF[tmpdir] = "default: $default[tmpdir]";
$deF[liFedir] = "default: $default[liFedir]";
$deF[maxsize] = "default: $default[maxsize]";
$deF[maxlength] = "default: $default[maxlength]";

$Usage = "Usage:\t$argv[0] [OPTION]...

options:
\t--host HOST\t use HOST ($deF[host])
\t--port PORT\t PORT to listen on ($deF[port])
\t--tmpdir DIR\t use DIR to store downloaded soundfiles
\t\t\t ($deF[tmpdir])
\t--liFedir DIR\t use DIR to store liFe-soundfiles
\t\t\t ($deF[liFedir])
```

```
\t--maxsize SIZE\t maximum SIZE [in MB] of the download file
\t\t\t ($deF[maxsize])
\t--maxlength DUR\t maximum DURation [in hh:mm:ss.frac] of soundfiles
\t\t\t ($deF[maxlength])
\t--show OUTPUT\t 'details' for verbose output
\t\t\t 'all' for more verbose output
\t\t\t 'ALL' for debug output
\t\t\t 'plugins' for all available plugins
\t\t\t 'config' for a dump of config variables
\t\t\t (verbose level can be overwritten by
\t\t\t  client integer argument '--verbose')
\t--help, -h\t display this help and exit

";


// Nachricht, mit der der server-Prozess vom client
// gestoppt werden kann
$serverDie = "server::DIE";


// Plugins bereitstellen & Default-Plugin definieren
// (entsprechende Var's in $default-Array aufnehmen)
include("./plugin_config.inc.php");


// EOFrank
?>
```

A.2 server_functions.inc.php

```
<?php // server_functions.inc.php

// ============================================================
// Function
// ============================================================
function logg($logString) {
    global $config;

    // Datum & Zeit im Format YYMMDD-hh:mm:ss
    $datum = date("ymd-H:i:s");
    echo "$datum $logString\n";

} // end function
```

```
// ============================================================
// Function
// ============================================================
function get_file($datei) {
    global $config, $bin, $verbose;

    clearstatcache(); // Datei-Status-Cache loeschen (-> filesize)

    // Spezial-Zeichen, die die Shell interpretieren koennte,
    // aus Sicherheitsgruenden verhindern
    $file[remote] = escapeshellcmd($datei);
    // basename -> nur Datei-Name ohne Pfad
    $file[name]   = basename($file[remote]);
    $file[local]  = "$config[tmpdir]/$config[timestamp]_" . $file[name];


    /*
     * Datei-Statistik ermitteln (Groesse & Mime-Typ)
     */

    // falls $file[remote] mit 'http://' beginnt, Statistik mit
    // $bin[wget] + Regex ermitteln
    if (preg_match("/^http:\/\//i", $file[remote])) {
        // --server-response -> HTTP-Response-Header ausgeben
        // --spider -> kein Download der Datei
        // $bin[stderr2stdout] -> Error-Out nach Standard-Out umleiten,
        //                        damit in output-Array geschrieben wird

        $wget_args = "--server-response --spider $file[remote]";
        $wget_command = "$bin[wget] $wget_args $bin[stderr2stdout]";
        exec($wget_command, $wget_output);

        if ($verbose > 2) {
            echo "\n\$wget_output: "; print_r($wget_output);
        }

        foreach ($wget_output as $zeile) {
            // Datei-Typ ermitteln
            $regex1 = "/^\s?\d+ Content-Type: (.*)/i";
            if (preg_match("$regex1", $zeile, $match)) {
                $file_mimetype = $match[1];
            }
            // Datei-Groesse ermitteln
            $regex2 = "/^\s?\d+ Content-Length: (\d+)/i";
            if (preg_match("$regex2", $zeile, $match)) {
                $file[size] = $match[1]; // eingeklammerter Ausdruck
            }
```

```
    }
}

// falls $file[remote] mit ‘ftp://’ beginnt
if (preg_match("/^ftp:\/\//i", $file[remote])) {
    if (is_file($file[remote])) {
        $file[size] = filesize($file[remote]);
    } else {
        //$error[] = "unable to open ’$file[remote]’";
        $error[] = "unable to open file";
    }
}

// falls $file[remote] mit ‘file://’ oder ‘/’ beginnt
if (preg_match("/^(file:\/\/|\/)/i", $file[remote])) {
    if (is_file($file[remote])) {
        $file[size] = filesize($file[remote]);

        $regex = "/^file:\/\/(.*)/i";
        if (preg_match("$regex", $file[remote], $match)) {
            $localFile = $match[1]; // Pfad ohne ’ftp://’
        } else {
            $localFile = $file[remote];
        }

        $mimetype = ‘$bin[file] -i $localFile‘;
        if (preg_match("/^\S+ (.+)/i", $mimetype, $match)) {
            $file_mimetype = $match[1];
        }

    } else {
        //$error[] = "unable to open ’$file[remote]’";
        $error[] = "unable to open file";
    }
}

// Datei-Groesse & Mime-Typ richtig?
if (empty($file[size])) {
    $error[] = "couldn’t determine filesize (file doesn’t exist?!)";
} else {
    $file[sizeKB] = $file[size] / 1024;
    $file[sizeMB] = $file[sizeKB] / 1024;
    echo "\n\tfile[size]: $file[size] bytes ($file[sizeMB] MB)\n";
}

if ($file[size] > $config[maxsize]) {
    $error[] = "file ist too big (maximum size: $config[maxsize])";
}
```

```
    if (preg_match("/($config[mimetype])/i", $file_mimetype, $match)) {
        $file[mimetype] = $match[1];
        echo "\tfile[mimetype]: $file[mimetype]\n";
    } else {
        $error[] = "wrong mime type: ‘$file_mimetype’";
    }



    /*
     * Datei-Download (in’s temporaere Verzeichnis ‘$config[tmpdir]’)
     */

    if (!empty($error)) { // falls Fehler aufgetreten sind
        echo "\n";
        foreach ($error as $errMsg) { // Fehlermeldung(en) ausgeben
            echo "\tERROR: $errMsg\n";
        }
        return FALSE;
    } else { // falls keine Fehler aufgetreten sind
        if (copy($file[remote], $file[local])) {
            echo "\n\tcopy:  $file[name]  ->  $file[local]\n";
             $file[md5] = md5_file($file[local]);
             $file[sha1] = sha1_file($file[local]);
            return $file;
        } else {
            echo "\n\tERROR: copy: unable to open file\n";
            return FALSE;
        }
    }
} // end function



// ============================================================
// Function
// ============================================================
function make_init_version($fileArray) {
    global $config, $verbose;
    // falls liFe.wav im angebenen Verzeichnis nicht existiert,
    // die uebergebene Datei ggf. nach wav umwandeln & nach
    // liFe.wav umbenennen (verschieben)

    // datei.mp3/ogg nach datei.wav konvertieren
    $fileArray[wav] = convert_to_wav($fileArray);

    // Datei beschneiden bzw. stauchen/strecken
```

```
    $sndFile = $fileArray[wav];
    $Length = $config[liFe_maxlength];
    if ($config[use_resize_function] == "truncate") {
         truncate_file_2_length($sndFile, $Length);
    } else {
        stretch_file_2_length($sndFile, $Length + 2);
          truncate_file_2_length($sndFile, $Length);
    }

    // Datei.wav ggf. auf globale Samplerate resamplen
    // & Lautstaerke optimieren
    normalize($fileArray[wav]);


    rename($fileArray[wav], $config[liFe]);
    echo "\n\tInfo: no liFe.forM (‘$config[liFe]’) found,\n";
    echo "\t      using ‘$fileArray[remote]’ as initial version\n";
//    echo "\t      using ‘$client[file]’ as initial version\n";

    if ($verbose) {
        echo "\n\$fileArray: "; print_r($fileArray);
    }

    // Datei-Array freigeben
    unset($fileArray);

} // end function




// ============================================================
// Function
// ============================================================
function convert_to_wav($fileArray) {
    global $config, $bin, $verbose;

    $tmpFile = "$config[tmpdir]/$config[tmpFilename]";

    if (preg_match("/mpeg/i", $fileArray[mimetype])) {
        $lame_args = "--decode $fileArray[local] $tmpFile";
        $lame_output = ‘$bin[lame] $lame_args $bin[stderr2stdout]‘;
        if ($verbose) {
            echo "\n\tInfo: using lame to decode ‘$fileArray[local]’\n";
        }
        if ($verbose > 2) { echo $lame_output; }
    } elseif (preg_match("/ogg/i", $fileArray[mimetype])) {
        $oggdec_args = "-o $tmpFile $fileArray[local]";
```

```
        $oggdec_output = `$bin[oggdec] $oggdec_args $bin[stderr2stdout]`;
        if ($verbose) {
            echo "\n\tInfo: using oggdec to decode `$fileArray[local]'\n";
        }
        if ($verbose > 2) { echo $oggdec_output; }
    } elseif (preg_match("/wav/i", $fileArray[mimetype])) {
        copy($fileArray[local], $tmpFile);
//         if ($verbose) {
        $infoMsg =  "`$fileArray[local]' is wav, ";
        $infoMsg .= "copied to $tmpFile";
            echo "\n\tInfo: $infoMsg\n";
//         }
    }

    return $tmpFile;
} // end function



// ============================================================
// Function
// ============================================================
function encode_wav_2_ogg($file) {
    global $bin, $verbose;

//     if ($verbose) {
        echo "\n\tInfo: using oggenc to encode `$file'\n";
//     }

    // Datum & Zeit im Format YYYY-MM-DD hh:mm:ss
    $datum = date("Y-m-d H:i:s");

    $oggenc_args = "--date '$datum' --artist 'liFe.fOrm' ";
    $oggenc_args .= "--title 'generated with a genetic structure ";
    $oggenc_args .= "written in php by Frank Ruebler ";
    $oggenc_args .= "licensed under the GPL' ";
    $oggenc_args .= "--comment 'eMail=ruebler@folkwang-hochschule.de' ";
    $oggenc_args .= "$bin[oggenc_quality] $file";
    $oggenc_output = `$bin[oggenc] $oggenc_args $bin[stderr2stdout]`;

    if ($verbose > 2) { echo $oggenc_output; }

} // end function


// ============================================================
// Function
// ============================================================
```

```
function truncate_file_2_size($file, $size) {
    global $bin, $verbose;

    if (!is_file($file)) {
        $errMsg = "cannot open file ‘$file’";
        echo "\n\tERROR: truncate_file_2_size(): $errMsg\n";
    } else {
        $file_size = filesize($file);

        $file_name = basename($file);
        $file_path = dirname($file);
        $file_orig = "$file_path/$file_name.orig_size";

        if ($file_size > $size) {
            rename($file, $file_orig);
            $dd_args = "bs=$size count=1 if=$file_orig of=$file";
            $dd_output = ‘$bin[dd] $dd_args $bin[stderr2stdout]‘;
            echo "\ndd said: \n$dd_output\n\n";
            echo "\tInfo: ‘$file’ truncated to $size\n";
            echo "\t      (original copied to ‘$file_orig’)\n\n";
        } elseif ($verbose) {
            echo "\n\tInfo: file NOT truncated!\n";
        }

    }
} // end function



// ============================================================
// Function
// ============================================================
function truncate_file_2_length($file, $length) {
    global $bin, $verbose;

    if (!is_file($file)) {
        $errMsg = "cannot open file ‘$file’";
        echo "\n\tERROR in truncate_file_2_length(): $errMsg\n";
    } else {

        $file_name = basename($file);
        $file_path = dirname($file);
        $file_orig = "$file_path/$file_name.truncate_orig";


        // $length muss im Format hh:mm:ss.frac vorliegen
        // (z.B. 10.5 = Zehn 1/2 Sekunden)
        // falls die Datei-Laenge unter der zu beschneidenen liegt,
```

```
        // kommt der sox-Aufruf einem Kopieren der Datei gleich

        // 0 -> Anfang der Datei, also nur Ende be-/abschneiden
        $start  = 0;
        $soxCmd = "$bin[sox] $file_orig $file trim $start $length";


            rename($file, $file_orig);
        $sox_output = `$soxCmd $bin[stderr2stdout]`;

        echo "\n\tInfo: `$file_name' truncated to $length\n";
        echo "\t      (original copied to `$file_orig')\n";

        if ($verbose > 1) { echo "\n$soxCmd\n"; }
        if (!empty($sox_output)) {
            echo "\n[truncate_file_2_length()] \$sox_output:\n";
            echo "$sox_output\n";
        }

    }

} // end function



// ============================================================
// Function
// ============================================================
function stretch_file_2_length($file, $length) {
    global $bin, $verbose;
    // kann zu Vergroesserung & Verkleinerung (Streckung & Stauchung)
    // der Datei genutzt werden, da Faktor aus originaler & uebergebener
    // Laenge errechnet wird


    if (!is_file($file)) {
        $errMsg = "cannot open file `$file'";
        echo "\n\tERROR in stretch_file_2_length(): $errMsg\n";
    } else {

        // $filenameArray = get_file_names($file);

        $file_name = basename($file);
        $file_path = dirname($file);
        $file_orig = "$file_path/$file_name.stretch_orig";

        $file_stat = get_file_stats($file);
```

```
        // Faktor, um den die Datei getreckt/gestaucht werden muss,
        // damit sie die uebergebene Laenge erreicht
        $factor = $length / $file_stat[length];

        $soxCmd = "$bin[sox] $file_orig $file stretch $factor";


        if ($factor != 1.0) {
            rename($file, $file_orig);
            $sox_output = `$soxCmd $bin[stderr2stdout]`;

            echo "\n\tInfo: `$file_name' stretched to $length\n";
            echo "\t      (original copied to `$file_orig')\n";

            if ($verbose) {
                echo "\t      Stretch-Factor: $factor\n";
            }
            if ($verbose > 1) { echo "\n$soxCmd\n"; }
            if (!empty($sox_output)) {
                echo "\n[stretch_file_2_length()] \$sox_output:\n";
                echo "$sox_output\n";
            }
        } else {
            $infoMsg = "`$file_name' fits with $length";
            $infoMsg .= " (no stretching need to be done)";
            echo "\n\tInfo: $infoMsg\n";
        }

    }

} // end function



// ============================================================
// Function
// ============================================================
function split_stereo_file($file_wav, $file_wavL, $file_wavR) {
    global $bin, $verbose;

    if (!is_file($file_wav)) {
        $errorMsg = "cannot open file `$file_wav'";
        echo "\tERROR in function split_stereo_file(): $errorMsg\n";
    } else {
        // sox-Effekt `avg' zum Trennen der Kanaele
         $LeftChannel  = "$bin[sox_paraMonoOut] $file_wavL avg -l";
         $RightChannel = "$bin[sox_paraMonoOut] $file_wavR avg -r";
//         $LeftChannel  = "-c 1 $file_wavL avg -l";
```

```
//         $RightChannel = "-c 1 $file_wavR avg -r";

        $left_args = "$file_wav $LeftChannel";
        $right_args = "$file_wav $RightChannel";

        $sox_output   = `$bin[sox] $left_args $bin[stderr2stdout]`;
        $sox_output  .= `$bin[sox] $right_args $bin[stderr2stdout]`;

        if ($verbose) {
            echo "\tsplit_stereo_file():\t$file_wav\n";
            echo "\t\t-> Left:  \t$file_wavL\n";
            echo "\t\t-> Right: \t$file_wavR\n";
        }
        if (!empty($sox_output)) {
            echo "\n[split_stereo_file()] \$sox_output:\n";
            echo "$sox_output\n";
        }
    }

} // end function



// ============================================================
// Function
// ============================================================
function get_file_names($datei) {

    $name[basename] = basename($datei); // nur Datei.Endung
    $name[dirname] = dirname($datei);

    preg_match("/^(.*)\.wav$/", $datei, $match);
    $name[filename] = $match[1]; // Datei-Name ohne Datei-Endung
    $name[wavL] = $name[filename] . "-L.wav";
    $name[wavR] = $name[filename] . "-R.wav";

    return $name;

} // end function



// ============================================================
// Function
// ============================================================
function get_file_stats($datei) {
    global $bin, $verbose;
    // mit sox und sndinfo Statistiken der $datei erstellen
```

```
// Array mit folgenden Informations-Feldern wird zurueckgegeben:
// srate, channels, bit, length, headersize, datasize,
// samples, dur, maxamp, minamp, roughfreq, voladjust,
// size, sizeKB, sizeMB, md5, sha1, mimetype


if (!is_file($datei)) {
    return "ERROR: cannot open file ‘$datei’";
}

if ($verbose > 1) { echo "\n\t===> Stats: $datei <===\n\n"; }


clearstatcache(); // Datei-Status-Cache loeschen (-> filesize)


// erforderliche Regex-Kontrollzeichen:
// \b -> Wort-Grenze (durch Leerzeichen,Komma,\n,... begrenzt)
// \s -> Whitespaces (Leerzeichen, Tabulator)
// \S -> alles ausser Whitespaces


// === $bin[sndinfo] ===
$sndinfo_output = ‘$bin[sndinfo] $datei $bin[no_stderr]‘;
if ($verbose > 2) {
    echo "\$sndinfo_output:\n"; print_r($sndinfo_output);
}

// srate, channels, bit, length (seconds)
$regex1 = "/srate \b(\S+)\b, \b(\S+)\b, \b(\S+)\b.*, \b(\S+)\b.*\n/";
if (preg_match("$regex1", $sndinfo_output, $match)) {
    if ($verbose > 2) {
        echo "\n(sndinfo) \$match: "; print_r($match);
    }
    $stats[srate] = $match[1];
    if ($match[2] == "monaural") { $stats[channels] = 1;
    } elseif ($match[2] == "stereo") { $stats[channels] = 2;
    } else { $stats[channels] = "unknown format"; }
    $stats[bit] = $match[3];
    $stats[length] = $match[4];
}
// headersize, datasize, samples
$regex2 = "/headersiz \b(\S+)\b, \b\S+\b \b(\S+)\b \(\b(\S+)\b.*\)/";
if (preg_match("$regex2", $sndinfo_output, $match)) {
    if ($verbose > 2) {
        echo "\n(sndinfo) \$match: "; print_r($match);
    }
```

```
    $stats[headersize] = $match[1];
    $stats[datasize] = $match[2];
    $stats[samples] = $match[3];
}



// === $bin[sox] ===
// -e  -> keine Ausgabe-Datei, sondern stderr verwenden
$sox_output = `$bin[sox] $datei -e stat $bin[stderr2stdout]`;
if ($verbose > 2) {
    echo "\n\$sox_output:\n"; print_r($sox_output); echo "\n";
}

$parse = array(
    "dur" => "Length",
    "maxamp"  => "Maximum\s+amp",
    "minamp"  => "Minimum\s+amp",
    "roughfreq" => "Rough\s+freq",
    "voladjust" => "Volume\s+adj");

foreach ($parse as $statvar => $val) {
    if (preg_match("/\n$val.*:\s*(\S+)\b/i", $sox_output, $match)) {
        if ($verbose > 2) {
            echo "\t(sox-stat) $statvar => $match[1]\n";
        }
        $stats[$statvar] = $match[1];
    }
}



// === size, sizeKB, sizeMB, md5, sha1, mimetype ===
$stats[size] = filesize($datei);
$stats[sizeKB] = $stats[size] / 1024;
$stats[sizeMB] = $stats[sizeKB] / 1024;

$stats[md5] = md5_file($datei);
$stats[sha1] = sha1_file($datei);

$file_output = `$bin[file] -i $datei`;
if (preg_match("/^.*:\s*(.*)/", $file_output, $match)) {
    $stats[mimetype] = $match[1];
}

if ($verbose > 1) { echo "\n\$stats: "; print_r($stats); }

return $stats;
```

```
} // end function



// ============================================================
// Function
// ============================================================
function normalize($wav_file) {
    global $config, $bin, $verbose;
    // uebergebene wav-Datei mit sox normalisieren/resamplen,
    // falls erforderlich

    if (!is_file($wav_file)) {
        echo "\n\tERROR in normalize(): ";
        echo "cannot open file ‘$wav_file’\n";
    } else {

        // Statistik der Sound-Datei.wav (srate,bit,voladjust)
        $wav_stats = get_file_stats($wav_file);

        $wav[name] = basename($wav_file);
        $wav[path] = dirname($wav_file);
        $wav[orig] = "$wav[path]/$wav[name].orig_stats";


        // Samplerate-Umwandlung
        if ($wav_stats[srate] != $config[sr]) {
            $arg[srate] = "-r $config[sr]";
            $arg[resample] = "resample";
        }

        // Bit-Umwandlung
        //if ($wav_stats[bit] != $config[bit]) { }
        if ($wav_stats[bit] != 16) {
            // -w -> 16 bit words
            // -s -> signed linear data encoding
            $arg[bit] = "-w -s";
        }


        // - 0.2 -> um clipping zu vermeiden
        $gain = $wav_stats[voladjust] - 0.2;
        if ($gain != 1.0) {
            $arg[vol] = "vol $gain";
        }

        if (!empty($arg)) { // also etwas fuer sox zu tun ist
            rename($wav_file, $wav[orig]);
```

```
            // sox-Aufruf
            //$arg[filetype] = "-t auto";
            //$arg[filetype] = "-t wav";

            $soxArgsIn  = "$arg[filetype] $wav[orig]";
            $soxArgsOut = "$arg[bit] $arg[srate] $wav_file";
            $soxArgsOut .= " $arg[resample] $arg[vol]";

            $sox_args = "$soxArgsIn $soxArgsOut";
            $soxCmd = "$bin[sox] $sox_args $bin[stderr2stdout]";
            $sox_output = `$soxCmd`;

            if ($verbose) {
                $infoMsg = "using following command";
                $infoMsg .= " to normalize `$wav[name]'";
                echo "\n\tInfo: $infoMsg\n\n";
                echo $soxCmd . "\n\n";
            } else { echo "\n\tInfo: `$wav[name]' normalized\n"; }

            if (!empty($sox_output)) {
                echo "\n[normalize()] \$sox_output:\n$sox_output\n";
            }

        } else { // also nichts fuer sox zu tun ist
            if ($verbose) {
                $infoMsg = "no normalization need to be done";
                $infoMsg .= " on `$wav[name]'";
                echo "\n\tInfo: $infoMsg\n\n";
            }
        }
    }
} // end function



// EOFrank
?>
```

A.3 server.php

```
#! ../organiSm/php5 -c ../organiSm/php5.ini
<?php // server.php

include("./server_config.inc.php");
include("./server_functions.inc.php");


// uebergebene Argumente verarbeiten/ueberpruefen
// ($argv[0] -> Programm-Name, $argv[1] -> erstes Argument, ...)
//echo "\$argc: $argc\n\$argv: "; print_r($argv);

// 1 -> beim ersten Argument beginnen (Programm-Name ueberspringen)
$i = 1;
while ($i < $argc) {
    // falls Argument mit '-' beginnt
    if (preg_match("/^--?([-a-zA-Z]+)/", $argv[$i], $match)) {
//        echo "\$match[1]: $match[1]\n";
        $option_orig = $argv[$i];
        $option_name = $match[1]; // Name ohne '--'

        // naechstes Argument, welches der zu setzende Wert sein sollte
        $i++;
        $value = $argv[$i];
        $config[$option_name] = $value;

        if (in_array($option_orig, $help_options)) {
            exit($Usage);
        }
        if (!in_array($option_name, $options)) {
            echo "ERROR: unknown option '$option_orig'\n";
            exit($tryHelp);
        }
        if (empty($value)) {
            echo "ERROR: missing value for '$option_name'\n";
            exit($tryHelp);
        }
        echo "option set: '$option_name'\t->  '$value'\n";
        $i++;
    } else {
        echo "ERROR: unknown argument '$argv[$i]'\n";
        exit($tryHelp);
    }
}


// Default-Werte setzen
foreach ($default as $defName => $defWert) {
```

```
    if (empty($config[$defName])) { $config[$defName] = $defWert; }
}


// fuer den Grad an (Detail-)Ausgaben int-Wert benutzen
// -> if ($verbose > 2) { echo ... }
if ($config[show] == "details") { $verbose = $config[verbose] = 1; }
if ($config[show] == "all") { $verbose = $config[verbose] = 2; }
if ($config[show] == "ALL") { $verbose = $config[verbose] = 3; }

// alle Konfigurations-Variablen ausgeben
if ($config[show] == "config") {
    echo "\n\$config: "; print_r($config);
    echo "\n\$default: "; print_r($default);
    echo "\n\$bin: "; print_r($bin);
    echo "\n\$plugin: "; print_r($plugin);
    echo "\nClient command that stops the server: ‘$serverDie’\n";
    exit;
}

if ($config[show] == "plugins") {
    echo "\navailable plugins:\n";
    foreach($plugin as $plugin_name => $plugin_file) {
        echo "\t‘$plugin_name’";
        if($default[plugin] == $plugin_name) {echo "   (default)";}
        echo "\n";
    }
    echo "\n";
    exit;
}
// Maximale Groesse der uebergebenen Datei, die
// in das temporaere Verzeichnis kopiert/heruntergeladen wird
$MB = 1024 * 1024; // 1 MegaByte
if (is_numeric($config[maxsize])) { // falls zulaessige Zahl
    $config[maxsize] = $config[maxsize] * $MB;
} else {
    echo "ERROR: ‘$config[maxsize]’ is not a valid number\n";
    exit($tryHelp);
}

// Definition der Sound-Dateien-Laengen [hh:mm:ss.frac]
// entweder durch $default[maxlength] oder durch client-Argument
if (is_numeric($config[maxlength])) { // falls zulaessige Zahl
    // Init-liFe.fOrm (falls liFe.wav noch nicht existent)
    // -> entspricht also der maximalen Laenge bei der "Geburt"
    $config[liFe_maxlength] = $config[maxlength];

    // heruntergeladenen Datei.wav (zu weiteren Verarbeitung)
```

```
    $config[file_maxlength] = $config[maxlength];
} else {
    echo "ERROR: ‘$config[maxlength]’ is not a valid number\n";
    exit($tryHelp);
}


if (!is_dir($config[tmpdir])) {
    echo "ERROR: ‘$config[tmpdir]’ is not a valid directory\n";
    exit($tryHelp);
}

if (!is_dir($config[liFedir])) {
    echo "ERROR: ‘$config[liFedir]’ is not a valid directory\n";
    exit($tryHelp);
}

// Variablen-Zuweisung muss hier stehen, damit evtl. --liFedir
// beruecksichtigt werden kann, bzw. $config[liFedir] = $default[liFedir]
// gesetzt wurde
// -->> ein server.php pro liFe.f0rm
//      -> wird ueber ‘-port xxx -liFedir’ definiert
//      also definiert das ueber ‘-liFedir’ angegebene Verzeichnis,
//      um welche liFe.f0rm es sich handelt
$config[liFe] = "$config[liFedir]/liFe.wav";

if ($verbose > 1) {
    echo "\n\$config: "; print_r($config);
    echo "\n\$default: "; print_r($default);
    echo "\n\$plugin: "; print_r($plugin);
}

$host = $config[host];
$port = $config[port];

// server mit host:port verknuepfen
// @ -> keine php-Fehlermeldung ausgeben
//      -> Fehler werden in $errno & $errstr geschrieben
$socket = @stream_socket_server("tcp://$host:$port", $errno, $errstr);

if (!$socket) {
    echo "ERROR: $errno - $errstr\n";
} else {
    echo "\n*** Server started: " . date("Y-m-d, H:i:s") . "\n";
    echo "*** Listening on $host (port $config[port])\n";

    while (1) { // Endlos-Schleife
        // mit Client verbinden
```

```
//  @ -> keine Fehlermeldung ausgeben
//       (‘Connection timed out...’)
// -1 -> heisst eigentlich kein timeout;
//       passiert aber doch nach ca. 70min (?)
while ($conn = @stream_socket_accept($socket,-1)) {

    echo $config[hr];
    logg("[ === begin of operation === ]"); echo "\n";

    // vom client empfangenen String mit Optionen (Argumenten)
    // in Array zerlegen (max. 8192 bytes empfangen)
    $transmission = fread($conn, 8192);
    $client_trans = explode(" ", $transmission);
    for ($i=0; $i < count($client_trans); $i++) {
        $key = $client_trans[$i];
        $i++;
        $val = $client_trans[$i];
        $client[$key] = $val;
    }

    // verbose von $client gesetzt
    if (!empty($client[verbose])) {
        $verbose = $client[verbose];
        $infoMsg = "client set verbose-level ‘$client[verbose]";
        echo "\tInfo: $infoMsg’\n\n";
    } else {
        // auf server-Wert (zurueck-)setzen (s.o.)
        $verbose = $config[verbose];
    }

    // true -> remote socket name (client)
    // false -> local socket name (server)
    $client[remoteName] = stream_socket_get_name($conn, true);

    if ($verbose > 1) {
        echo "\n\$client: ";
        print_r($client);
        echo "\n";
    }

    fclose($conn); // Verbindung zum client beenden

    if ($client[file] == $serverDie) { break; }

    echo "\tClient ($client[remoteName]) submitted URL:\n";
    echo "\t‘$client[file]’\n";

    if (!empty($plugin[$client[plugin]])) {
```

```
    $infoMsg  = "using plugin ‘$client[plugin]’";
    $infoMsg .= " (defined by client-option)";
    echo "\n\tInfo: $infoMsg\n";
    $config[plugin] = $client[plugin];
} else {
    if ($verbose) {
        $infoMsg = "using default-plugin ‘$default[plugin]’";
        echo "\n\tInfo: $infoMsg\n";
    }
    $config[plugin] = $default[plugin];
}


// vom Client definierte resize-Funktion
// -> ‘truncate’ (‘trim’) oder ‘stretch’
if ($client[resize] == "truncate") {
    $infoMsg  = "using resize function ‘truncate’";
    $infoMsg .= " (defined by client-option)";
    echo "\n\tInfo: $infoMsg\n";
    $config[use_resize_function] = $client[resize];
} else {
    if ($verbose) {
        $infoMsg = "using default resize function ‘stretch’";
        echo "\n\tInfo: $infoMsg\n";
    }
    $config[use_resize_function] = "stretch";
}

// unix-timestamp -> fuer Datei-Namen & random-seed
// erst 2 Sekunden warten, um evtl. gleichen timestamp
// aus vorhergegangenem Durchlauf zu vermeiden
// (erst bei SEHR schnellen Rechnern relevant ;-)
sleep(2);
$config[timestamp] = time();
if ($verbose) {
    echo "\n\t\$config[timestamp]: $config[timestamp]\n";
}
mt_srand($config[timestamp]); // seed -> mt_rand


// check & Download der Datei
// -> gibt Array mit (u.a.) Pfad auf lokale Datei zurueck,
//    falls Download erfolgreich war
$file = get_file($client[file]);

if (!$file) { // falls get_file() fehlschlug
    echo "\n\tWarning: unable to use ‘$client[file]’\n";
    echo "\t         no operation done (\$file: FALSE)\n";
```

```
        // $client-Array freigeben
        unset($client);

    } elseif ($file && !is_file($config[liFe])) {
        // falls liFe.wav im angebenen Verzeichnis nicht
        // existiert, die uebergebene Datei ggf. nach wav
        // umwandeln & nach liFe.wav umbenennen (verschieben)
        make_init_version($file);

        // Array's freigeben
        unset($client, $file);

    } else { // Sound-Datei verarbeiten
        $liFe[wav] = $config[liFe];
        $liFe[alt] = "$config[liFedir]/$config[timestamp].wav";

        // MySQL-Datenbank-Abfrage -> noch nicht implementiert!
        //db_check($file);

        // datei.mp3/ogg nach datei.wav konvertieren
        $file[wav] = convert_to_wav($file);

        // Datei beschneiden, falls zu gross
        $sndFile = $file[wav];
        $Length = $config[file_maxlength];
        if ($config[use_resize_function] == "truncate") {
            truncate_file_2_length($sndFile, $Length);
        } else {
            stretch_file_2_length($sndFile, $Length + 2);
            truncate_file_2_length($sndFile, $Length);
        }

        // Datei-Pfad in Einzelteile zerlegen
        // -> basename, dirname, filename, wavL, wavR
        $file_name_array = get_file_names($file[wav]);
        // Array's zusammenfuegen; schon vorhandene Array-Keys
        // werden NICHT ueberschrieben!
        $file = $file + $file_name_array;
        $liFe_name_array = get_file_names($liFe[wav]);
        $liFe = $liFe + $liFe_name_array;


        // Datei.wav ggf. auf globale Samplerate resamplen
        // & Lautstaerke optimieren
        normalize($file[wav]);
        //normalize($liFe[wav]);
```

```
// Sound-Datei-Statistiken (srate, channels, ...)
// -> kommen als Array zurueck
$file[wav_stat] = get_file_stats($file[wav]);
$liFe[wav_stat] = get_file_stats($liFe[wav]);


// $datei[wavL] & $datei[wavR] erzeugen
// (muss als $datei[wav] vorhanden sein)
if ($file[wav_stat][channels] == 2) {
    $stereo = $file[wav];
    $left   = $file[wavL];
    $right  = $file[wavR];
    split_stereo_file($stereo, $left, $right);
} elseif ($file[wav_stat][channels] == 1) {
    $file[wavL] = $file[wavR] = $file[wav]; // Mono
} else {
    $errMsg = "unsupported channelformat";
    echo "\tERROR: $errMsg (‘$file[wav]’)\n";
}

if ($liFe[wav_stat][channels] == 2) {
    $stereo = $liFe[wav];
    $left   = $liFe[wavL];
    $right  = $liFe[wavR];
    split_stereo_file($stereo, $left, $right);
} elseif ($liFe[wav_stat][channels] == 1) {
    $liFe[wavL] = $liFe[wavR] = $liFe[wav]; // Mono
} else {
    $errMsg = "unsupported channelformat";
    echo "\tERROR: $errMsg (‘$liFe[wav]’)\n";
}


// Soundprocessing
$liFe[neu] = use_plugin($config[plugin], $liFe, $file);

if (!is_file($liFe[neu])) {
    echo "\tWarning: plugin ‘$config[plugin]’ failed\n";
} else {
    rename($liFe[wav], $liFe[alt]);
    rename($liFe[neu], $liFe[wav]);
    echo "\trename:  $liFe[wav]  ->  $liFe[alt]\n";
    echo "\trename:  $liFe[neu]  ->  $liFe[wav]\n";
    // neu erzeugte Datei normalisieren
    // -> wird in plugin ggf. erledigt!
    //normalize($liFe[wav]);
}
```

```
                // ins Ogg-Vorbis Format enkodieren (-> liFe.ogg)
                encode_wav_2_ogg($liFe[wav]);
                // noch nicht implementiert:
                //$liFe[mp3] = encode_wav_2_mp3($liFe[wav]);

                if ($verbose) {
                    echo "\n\$file: "; print_r($file);
                    echo "\n\$liFe: "; print_r($liFe);
                }

                // temporaere L/R-Dateien loeschen, falls vorhanden
                if ($file[wavL] != $file[wav] && $file[wavR] != $file[wav]
                    && is_file($file[wavL]) && is_file($file[wavR])) {
                    unlink($file[wavL]);
                    unlink($file[wavR]);
                }
                if ($liFe[wavL] != $liFe[wav] && $liFe[wavR] != $liFe[wav]
                    && is_file($liFe[wavL]) && is_file($liFe[wavR])) {
                    unlink($liFe[wavL]);
                    unlink($liFe[wavR]);
                }

                // Variablen freigeben/zerstoeren
                // (-> werden pro Client-Anfrage, bzw. Schleifen-Durchlauf
                //     neu gesetzt)
                unset($client, $file, $liFe);
            }
            echo "\n";
            logg("[ === end of operation === ]");
        }
        if ($client[file] == $serverDie) {
            echo "\nServer killed by client ";
            echo "on host $client[remoteName]\n\n";
            break;
        }
    }

    fclose($socket);

    echo $config[hr];
    echo "*** Server stopped: " . date("Y-m-d, H:i:s");
    echo $config[hr];
    echo "\n\n";
}

// EOFrank
?>
```

A.4 server+dir+port.sh

```
#! /bin/bash

LHOME="/home/morph/LforM/scripts/"
#LHOME="/cdrom/LforM/scripts/"

# Maximum size of the uploaded file
MAXSIZE="2.0" # [MB]

# length of soundfiles
MAXLENGTH="8.0" # [hh:mm:ss.frac]


# Argument-Anzahl ueberpruefen $# >= 1
if [ ! $# -ge 1 ]
    then
    echo -e "Usage: ${0} DIR [PORT]

\tDIR & DIR/upload will be created if necessary
\tPORT is optional (default: 12965)
\toutput will be written to DIR/lf.log

\tedit this script to get more access to server control arguments
\t(like maximum values for size & length)
"
    exit 1
fi

LDIR=$1
LTMP="${LDIR}/download"
LLOG="${LDIR}/lf.log"
# LLOG="${LDIR}/log.txt"

# PORT auf Default setzen, falls nicht uebergeben
if [ -z $2 ] # -z => zero length
    then
    LPORT="12965"
else
    LPORT=$2
fi

# Verzeichnisse anlegen, falls noch nicht existent
if [ ! -d ${LDIR} ]
    then
    echo -e "\nliFe directory doesn't exist, creating it."
    echo -e "mkdir ${LDIR}"
    mkdir ${LDIR}
fi
```

```
if [ ! -d ${LTMP} ]
    then
    echo -e "\nupload directory doesn't exist, creating it."
    echo -e "mkdir ${LTMP}\n"
    mkdir ${LTMP}
fi


DIRS="-liFedir ${LDIR} -tmpdir ${LTMP}"
MAX="-maxsize ${MAXSIZE} -maxlength ${MAXLENGTH}"

CMD="./server.php -port ${LPORT} ${DIRS} ${MAX}"

echo "executing following command:"
echo "${CMD} >> ${LLOG} 2>&1 &"

cd ${LHOME}
${CMD} >> ${LLOG} 2>&1 &
```

B Plugins

B.1 plugin_config.inc.php

```
<?php // plugin_config.inc.php

// u.a. csound-pvanal bereitstellen:
include("plugin_functions.inc.php");

// hier definierte Plugins koennen mit
// ‘client.php -plugin PLUGIN-NAME’ benutzt werden

// default-plugin wird benutzt, falls kein ’-plugin’
// uebergeben wurde, bzw. Plugin-Name nicht existiert
//$default[plugin] = "pvCross";
$default[plugin] = "grain";

// $plugin[PLUGIN-NAME] = "plugin_file.php";
$plugin[pvCross] = "plugin_pvCross.php";
$plugin[grain] = "plugin_grain.php";
$plugin[template] = "plugin_template.php";


// definierte Plugin-Dateien einbinden,
// bzw. auf Existenz derselbigen pruefen
foreach ($plugin as $dateiName) {
    if (is_file("./$dateiName")) {
        include("./$dateiName");
    } else {
        $errorMsg  = "cannot open plugin-file ‘$dateiName’";
        $errorMsg .= " (defined in plugin_config.inc.php)";
        exit("ERROR: $errorMsg\n");
    }
}



// ============================================================
// Function
// ============================================================
function use_plugin($pluginName, $liFe, $file) {

    echo "\n\t==========[ begin plugin ‘$pluginName’ ]==========\n\n";

    // Plugin-Funktion, die benutzt werden soll
    // (in der jeweiligen Plugin-Datei definiert)
    // $liFe -> Array der jetzigen LiFe.fOrm Sound-Datei
    // $file -> Array der zu verarbeitende, heruntergeladene Datei
```

```
    switch($pluginName) {
        case "pvCross":
            $postFile = plugin_pvCross($liFe, $file);
            break;
        case "grain":
            $postFile = plugin_grain($liFe, $file);
            break;
        case "template":
            $postFile = plugin_template($liFe, $file);
            break;
    }

    echo "\n\t==========[ end plugin ‘$pluginName’ ]==========\n\n";

    return $postFile;
} // end function


// EOFrank
?>
```

B.2 plugin_functions.inc.php

```
<?php // plugin_functions.inc.php

// ============================================================
// Function
// ============================================================
function pvanal($wavInputFile, $pvOutputFile) {
    global $bin;

    $pvanal_args = "$wavInputFile $pvOutputFile";
    $exec_output = ‘$bin[pvanal] $pvanal_args $bin[no_stderr]‘;

    return $exec_output;
} // end function



// ============================================================
// Function
// ============================================================
function Array2File($data_array, $file_handle) {
    // $file_handle wird uebergeben => fopen/fclose ausserhalb
    $br = "\n";
```

```
    foreach ($data_array as $line) {
        fwrite ($file_handle, "$line $br");
    }

    fwrite ($file_handle, "$br$br");
} // end function



// ============================================================
// Function
// ============================================================
function calculate_ftablesize($sampleFrames) {
    // Groesse der ftables errechnen
    // ceil -> rundet aUf die naechste Gannzzahl
    //                2er-log Samples
    $potenz = ceil((log10($sampleFrames)) * 3.32192809);
    $ftableSize = pow(2,$potenz); // 2 hoch Potenz

    return $ftableSize;
} // end function



// EOFrank
?>
```

B.3 Plugin: grain

B.3.1 plugin_grain_config.inc.php

```
<?php // plugin_grain_config.inc.php

// zusaetzliche Plugin-interne (lokale) Variablen
// (also Var's, die nur innerhalb des Plugins Gueltigkeit haben)

$conf[usr_orc] = "$usr[filename].orc";
$conf[usr_sco] = "$usr[filename].sco";

$conf[usr_out] = "$usr[filename]_GenEratEd.wav";
$conf[usr_outL] = "$usr[filename]_GenEratEd-L.wav";
$conf[usr_outR] = "$usr[filename]_GenEratEd-R.wav";

// liFe.wav wird momentan nicht von grain_plugin verarbeitet
// wird wie es ist mit $conf[usr_out] granular gemischt
//$conf[liFe_orc] = "$config[tmpdir]/liFe.orc";
//$conf[liFe_sco] = "$config[tmpdir]/liFe.sco";
$conf[liFe_out] = $liFe[wav]; // -> soxmix-Argument

$conf[mix_orc] = "$config[tmpdir]/grain_mix.orc";
$conf[mix_sco] = "$config[tmpdir]/grain_mix.sco";

// Name der neu erzeugten Datei wird zurueckgegeben
$conf[newliFe] = "$liFe[filename]_new.wav";


$conf[liFe_wavL] = $liFe[wavL];
$conf[liFe_wavR] = $liFe[wavR];

$conf[usr_wavL] = $usr[wavL];
$conf[usr_wavR] = $usr[wavR];


$conf[liFe_dur]  = $liFe[wav_stat][dur];  // [sec]
$conf[usr_dur]   = $usr[wav_stat][dur];  // [sec]

$conf[liFe_samples] = $liFe[wav_stat][samples];
$conf[usr_samples] = $usr[wav_stat][samples];


// Groesse der ftables errechnen
$conf[liFe_ftableSize] = calculate_ftablesize($conf[liFe_samples]);
$conf[usr_ftableSize] = calculate_ftablesize($conf[usr_samples]);


// wird auch fuer Seed in generate_grain_sco() benutzt
```

```
$conf[timestamp] = $config[timestamp];

// $conf[growing]  = $conf[timestamp] * 0.00000000003;
$conf[growing]  = $conf[timestamp] * 0.0000000002;

$file_durations = $conf[liFe_dur] + ($conf[usr_dur] * 0.0001);
$conf[TotalDur] = $file_durations + $conf[growing];  // [sec]


// sr, ksmps & kr -> csound.orc
$conf[sr]    = $config[sr];
$conf[ksmps] = 100;
$conf[kr]    = $conf[sr]/$conf[ksmps];  // 44100/100 = 441


$conf[GenEratE_bin] = "$config[bindir]/GenEratE";
$conf[soxmix_bin] = $bin[soxmix];


$conf[verbose] = $verbose;

if ($conf[verbose] > 1) { echo "\n\$conf: "; print_r($conf); }

// EOFrank
?>
```

B.3.2 plugin_grain.php

```
<?php // plugin_grain.php

// ============================================================
// Main-Function
// ============================================================
function plugin_grain($liFe, $usr) {
    global $bin, $config, $verbose;


    include("./plugin_grain_config.inc.php");


    // $conf[usr_orc] & $conf[usr_sco] generieren
    generate_grain_orc($conf);
    generate_grain_sco($conf);


    // csound-Aufruf um $conf[usr_out] zu erzeugen
    $usr_args = "$conf[usr_out] $conf[usr_orc] $conf[usr_sco]";
    $csound_output[0] = "$bin[csound] $usr_args $bin[stderr2stdout]\n";
    $csound_output[] = `$bin[csound] $usr_args $bin[stderr2stdout]`;



    // $conf[usr_out] auf $conf[TotalDur] stretchen
    stretch_file_2_length($conf[usr_out], $conf[TotalDur]);
    echo "\n";

    // Statistiken vom zuvor generierten $conf[usr_out]
    $stats = get_file_stats($conf[usr_out]);

    // ftable-Groesse fuer $conf[usr_out] errechnen
    $conf[usr_out_ftablesize] = calculate_ftablesize($stats[samples]);

    // $conf[usr_out] in L/R splitten
    split_stereo_file($conf[usr_out], $conf[usr_outL], $conf[usr_outR]);


    // granulares Mischen von $conf[liFe_wavL/R] & $conf[usr_outL/R]
    generate_grainMix_orc($conf);
    generate_grainMix_sco($conf);

    $mix_args = "$conf[newliFe] $conf[mix_orc] $conf[mix_sco]";
    $csound_output[] = "$bin[csound] $mix_args $bin[stderr2stdout]\n";
    $csound_output[] = `$bin[csound] $mix_args $bin[stderr2stdout]`;
```

```
    // mit soxmix zusammenmischen -> falls '--grain-mix plain'
    //                               -> noch nicht aktiviert <-
//     $mix_args = "$conf[usr_out] $conf[liFe_out] $conf[newliFe]";
//     $soxmix_output = '$conf[soxmix_bin] $mix_args $bin[stderr2stdout]';
//     if (!empty($soxmix_output)) {
//         echo "\n[plugin_grain()] \$soxmix_output:\n";
//         echo $soxmix_output;
//     }


    if ($conf[verbose]) {
        echo "\n\$csound_output: "; print_r($csound_output);
    }


    // temporaere Dateien loeschen
    $unlinkArray = array(
//         $conf[usr_orc],  $conf[usr_sco],
//         $conf[mix_orc],  $conf[mix_sco],
        $conf[usr_out], $conf[usr_outL], $conf[usr_outR]
        );

    if ($conf[liFe_out] != $liFe[wav]) {
        $unlinkArray[] = $conf[liFe_out];
    }

    foreach ($unlinkArray as $tmpFILE) {
        if (is_file($tmpFILE)) { unlink($tmpFILE); }
    }

    normalize($conf[newliFe]);

    echo "\n\t$conf[newliFe] generated\n";

    // Plugin-Ende, neu erzeugte Datei zurueckgeben
    return $conf[newliFe];

} // end function



// ============================================================
// Sub-Function
// ============================================================
function generate_grain_orc($conf) {

    if ($conf[verbose] > 1) {
```

```
        echo "\n\t===[   generate_grain_orc()   ]===\n";
    }


    $orc_file = fopen ("$conf[usr_orc]", "w");
    fwrite ($orc_file, "
; grain.orc

sr     = $conf[sr]
kr     = $conf[kr]
ksmps  = $conf[ksmps]
nchnls = 2


;*******;
 instr 1
;*******;

idur        = p3        ; Gesamt-Dauer
iamp        = ampdb(p4)*0.5
ismpstart   = p5        ; Anfangs-Sampleposition
itrans      = p6        ; Transpositions-Stufe (negativ=reverse)
iattack     = p7
irelease    = p8
isustain    = idur - (iattack+irelease)
ipan        = p9

iosciltable = 1
isamplL     = 2         ; f2
isamplR     = 3         ; f3

kamp linseg 0, iattack, iamp, isustain, iamp, irelease, 0

 aindx oscili itrans*idur*sr, 1/idur, iosciltable ; indx fuer tablei
 asigL tablei ismpstart+aindx, isamplL
 asigR tablei ismpstart+aindx, isamplR

 aleft  = sqrt(1-ipan)*kamp*asigL
 aright = sqrt(ipan)*kamp*asigR

 outs   aleft, aright

endin

");
    fclose ($orc_file);

    echo "\t$conf[usr_orc] generated\n";
```

```
} // end function



// ============================================================
// Sub-Function
// ============================================================
function generate_grain_sco($conf) {

    if ($conf[verbose] > 1) {
        echo "\n\t===[   generate_grain_sco()   ]===\n";
    }


    $sco_file = fopen ("$conf[usr_sco]", "w");
    fwrite ($sco_file, "
; grain.sco

f1 0 1024 7 0 1024 1 ; osciltable

f2 0 $conf[usr_ftableSize] 1 \"$conf[usr_wavL]\" 0 4 1 ; links
f3 0 $conf[usr_ftableSize] 1 \"$conf[usr_wavR]\" 0 4 1 ; rechts

;instr start dur ampdb smpstart transp attack release pan
;            (p3   p4     p5      p6     p7      p8    p9)
");


    // relative Laengen der Chromosomen (Summe 1.0 <=> 100%)
    // (nachgemessene Laengen der 23 menschlichen Chromosomenpaare)
    $ChromosomLaengen=array(
        0.083, 0.072, 0.067,
        0.061, 0.061,
        0.056, 0.050, 0.044, 0.050, 0.044, 0.044, 0.039,
        0.033, 0.033, 0.033,
        0.033, 0.028, 0.028,
        0.022, 0.022,
        0.022, 0.017,
        0.056);  // Summe=0.998

    // zufaellige Neuordnung (Rekombination) der Array-Elemente
    shuffle($ChromosomLaengen);

    if ($conf[verbose] > 1) {
        echo "\n\$ChromosomLaengen: ";
        print_r($ChromosomLaengen);
        echo "\n";
```

```
    }

    // Snd-Ausschnitts-Grenzen [in Samples] auf Basis der
    // ChromosomLaengen bestimmen
    $usrSampleGrenze[0]=0;  // Sample 0 der Snd-Datei
    $i=1; // bei Array-Index 1 (also 2. Position) beginnen
    foreach($ChromosomLaengen as $length) {
        $uSmpLg = $length * $conf[usr_samples]; // [samples]
        $usrSampleGrenze[$i] = round($usrSampleGrenze[$i-1] + $uSmpLg);
        $i++;
    }

    // letztes Sample definieren (wegen Rundungsfehler)
    $usrSampleGrenze[$i-1] = $conf[usr_samples];


    if ($conf[verbose] > 1) {
        echo "\n\$usrSampleGrenze: "; print_r($usrSampleGrenze);
    }



// Anzahl des SampleGrenzen-Arrays (-> fuer Min/MaxSample)
    // -1, da letztes Element kein MinSample sein kann
    $ArrayElementeAnzahl = (sizeof($usrSampleGrenze)) - 1;
    if ($conf[verbose] > 1) {
        echo "\n\n\$ArrayElementeAnzahl: $ArrayElementeAnzahl\n\n";
    }



// Default/Initial-Argumente fuer Aufruf von externem
// C-Programm 'GenEratE' definieren; muessen alle gesetzt
// werden, dammit in richtiger Reihenfolge im Array vorhanden
// -> Reihenfolge muss eingehalten werden, da sonst Programm-
//    Aufruf von 'GenEratE' fehlschlaegt

// Argument-Reihenfolge:
//     MinSample MaxSample GrainstreamNumber StartTime TotalDur
//     MinTrans MaxTrans MinAmp(db) MaxAmp(db) Pan(0-1)
//     Seed Samplerate MinDur MaxDur Attack Release

    $arg = array(
        "MinSample"   => "0",
        "MaxSample"   => "10",
        "GrainstreamNumber" => "40", // Anzahl der Grain-Streams
        "StartTime"   => "0",
        "TotalDur"    => $conf[TotalDur] / ($ArrayElementeAnzahl+1),
```

```
                        // <=> Gesamtlaenge / 24
        "MinTrans"    => "0.4",
        "MaxTrans"    => "0.7",
        "MinAmp"      => "60", // [db]
        "MaxAmp"      => "88", // [db]
        "Pan"         => "0.5", // 0 => links, 1 => rechts
        "Seed"        => "$conf[timestamp]",
        "Samplerate"  => "$conf[sr]",
        "MinDur"      => "0.03",
        "MaxDur"      => "0.04",
        "Attack"      => "0.01",
        "Release"     => "0.01"
        );

    if ($conf[verbose] > 1) {
        echo "\n\$arg: [[initial values]] "; print_r($arg);
    }



    // Sample-Auschnitte dem erzeugten Array entsprechend aneinanderreihen
    // alle 'Gen-Array'-Elemente durchlaufen:
    for($y=0; $y < $ArrayElementeAnzahl; $y++) {
        $arg[MinSample]=$usrSampleGrenze[$y];
        $arg[MaxSample]=$usrSampleGrenze[$y+1];

        // Dauer des SoundFile-Ausschnitts auf Basis des
        // durch MinSample & MaxSample definierten Bereichs
        // berechnen (-> fuer MinDur/MaxDur Default-Werte)
        $SampleAnzahl = $arg[MaxSample] - $arg[MinSample];
        $SndAusschnittsDauer = $SampleAnzahl / $arg[Samplerate];
        $arg[MinDur] = $SndAusschnittsDauer * 0.2; // [sec]
        $arg[MaxDur] = $SndAusschnittsDauer * 0.5; // [sec]

        // Achtung: Attack + Release < MinDur
        $arg[Attack]  = $arg[Release] = $arg[MinDur] * 0.4;
//        $arg[Attack]  = $arg[Release] = $SndAusschnittsDauer * 0.15;



        // $arg[umente] anfuegen
        // Array-Elemente durch Leerzeichen getrennt in String zusammenfassen
        $genArguments = implode(" ", $arg);

        if ($conf[verbose] < 3) {
            // stderr-Ausgabe von GenEratE unterdruecken
            $stderrOutput = "2>/dev/null";
        }
```

```
        $genCmd = "$conf[GenEratE_bin] $genArguments $stderrOutput";


        if ($conf[verbose] > 1) {
            echo "\n[$y] \$arg[StartTime]:\t $arg[StartTime]\n";
            echo "[$y] \$arg[TotalDur]:\t $arg[TotalDur]\n";
            echo "[$y] \$arg[MinSample]:\t $arg[MinSample]\n";
            echo "[$y] \$arg[MaxSample]:\t $arg[MaxSample]\n";
            echo "[$y] \$SndAusschnittsDauer: $SndAusschnittsDauer\n";
            echo "[$y] \$arg[MinDur]:\t $arg[MinDur]\n";
            echo "[$y] \$arg[MaxDur]:\t $arg[MaxDur]\n";
            echo "[$y] \$arg[Attack]:\t $arg[Attack]\n";
            echo "[$y] \$arg[Release]:\t $arg[Release]\n";
        }
        if ($conf[verbose] > 1) {
            echo "\n[$y] \$arg: [[altered values]] "; print_r($arg);
        }

        if ($conf[verbose] > 1) {
            echo "[$y] $genCmd\n";
        }

        if ($conf[verbose] > 2) {
            echo "\n[$y]  [=== GenEratE - stderrOutput ===]\n";
        }

        exec("$genCmd", $GenEratEd_Events);

        Array2File($GenEratEd_Events, $sco_file);

        unset($GenEratEd_Events);     // Array loeschen

        $arg[StartTime] = $arg[StartTime] + $arg[TotalDur];

        // Ueberlappung der aneinandergereihten Ausschnitte
        $Ueberlappung = 0.05; //.0003; 0.03;
        $arg[StartTime] = $arg[StartTime] - $Ueberlappung;
        //$arg[TotalDur] = $arg[TotalDur] + $Ueberlappung;
    }

    fwrite ($sco_file, "

e

");

    fclose ($sco_file);
```

```
    echo "\t$conf[usr_sco] generated\n";

} // end function



// ========================================================
// Sub-Function
// ========================================================
function generate_grainMix_orc($conf) {

    if ($conf[verbose] > 1) {
        echo "\n\t===[   generate_grainMix_orc()   ]===\n";
    }


    $orc_file = fopen ("$conf[mix_orc]", "w");
    fwrite ($orc_file, "
; grainMix.orc

sr     = $conf[sr]
kr     = $conf[kr]
ksmps  = $conf[ksmps]
nchnls = 2


;*******;
 instr 1
;*******;

idur        = p3        ; Gesamt-Dauer
iamp        = ampdb(p4)*0.5
ismpstart   = p5        ; Anfangs-Sampleposition
itrans      = p6        ; Transpositions-Stufe (negativ=reverse)
iattack     = p7
irelease    = p8
isustain    = idur - (iattack+irelease)

isamplL     = p9          ; ftable links
isamplR     = isamplL + 2 ; ftable rechts

iosciltable = 1

kamp linseg 0, iattack, iamp, isustain, iamp, irelease, 0

 aindx oscili itrans*idur*sr, 1/idur, iosciltable ; indx fuer tablei
 asigL tablei ismpstart+aindx, isamplL
```

```
 asigR tablei ismpstart+aindx, isamplR

 aleft  = kamp*asigL
 aright = kamp*asigR

 outs   aleft, aright

endin

");
    fclose ($orc_file);

    if ($conf[verbose]) { echo "\n"; }
    echo "\t$conf[mix_orc] generated\n";

} // end function



// ============================================================
// Sub-Function
// ============================================================
function generate_grainMix_sco($conf) {

    if ($conf[verbose] > 1) {
        echo "\n\t===[   generate_grainMix_sco()   ]===\n";
    }


    $sco_file = fopen ("$conf[mix_sco]", "w");
    fwrite ($sco_file, "
; grainMix.sco

f1 0 1024 7 0 1024 1 ; osciltable

f2 0 $conf[liFe_ftableSize] 1 \"$conf[liFe_wavL]\" 0 4 1 ; links
f4 0 $conf[liFe_ftableSize] 1 \"$conf[liFe_wavR]\" 0 4 1 ; rechts

f3 0 $conf[usr_out_ftablesize] 1 \"$conf[usr_outL]\" 0 4 1 ; links
f5 0 $conf[usr_out_ftablesize] 1 \"$conf[usr_outR]\" 0 4 1 ; rechts

; ftable -> Sound-Datei (f2/f4 oder f3/f5)
;instr start dur ampdb smpstart transp attack release ftable
;            (p3   p4     p5      p6     p7      p8    p9)
");

    // Minima & Maxima fuer Zufalls-Auswahl festlegen
    $def = array(
```

```
    "durMin" => 100, // [msec] -> muss fuer mt_rand() int sein
    "durMax" => 250, // [msec] -> muss fuer mt_rand() int sein
    "ampMin" => 75,  // [db]
    "ampMax" => 92   // [db]
    );


$smpstart = 0; // bei Sample 0 beginnen
$start = 0;    // ab Sekunde 0 hochzaehlen
$trans = 1;    // originale Transposition
$index = 0;    // Index-Zaehler fuer $mixEventArray

// grain-Ueberlappung
// Achtung: muss groesser als durMin (in sec) sein!
//$Ueberlappung = 0.08; // [sec]
$Ueberlappung = ($def[durMin] - 10) * 0.001; // [sec]

$maxDur = $def[durMax] * 0.001; // [msec] -> [sec]

// csound-Events erzeugen
while ( ($start+$maxDur) < $conf[TotalDur]) {
    $dur = mt_rand($def[durMin], $def[durMax]); // [msec]
    $dur = $dur * 0.001; // [msec] -> [sec]
    $amp = mt_rand($def[ampMin], $def[ampMax]);
    $attack = $release = $dur * 0.2; // [sec]

    // Sound-Datei zufaellig auswaehlen
    $ftable = mt_rand(2,3); // f2 oder f3

    if ($conf[verbose] > 1) {
        echo "\n[$index] start: $start, dur: $dur, amp: $amp, ";
        echo "smpstart: $smpstart, trans: $trans, ";
        echo "attack: $attack, release: $release, ";
        echo "ftable: $ftable";
    }

    // i1 $start $dur $amp $smpstart $trans $attack $release $ftable
    // naechsten Array-Eintrag mit Werten fuellen
    $mixEventArray[$index] = "  i1  $start  $dur  $amp ";
    $mixEventArray[$index] .= "  $smpstart    $trans    ";
    $mixEventArray[$index] .= "$attack   $release   $ftable";


    // Werte fuer den naechsten Durchlauf setzen:
    // Array-Index hochzaehlen
    $index++;
    // Start-Zeit hochzaehlen
    $start = $start + $dur - $Ueberlappung; // [sec]
```

```
        // naechste Sample-Start-Position errechnen
        $smpstart = round($start * $conf[sr]); // [samples]
    }

    Array2File($mixEventArray, $sco_file);

    unset($mixEventArray);     // Array loeschen

    fwrite ($sco_file, "e\n");

    fclose ($sco_file);

    if ($conf[verbose]) { echo "\n"; }
    if ($conf[verbose] > 1) { echo "\n"; }
    echo "\t$conf[mix_sco] generated\n";

} // end function



// EOFrank
?>
```

B.3.3 GenEratE.c

```
/******************
 **  GenEratE.c  **
 ******************/

#include <stdio.h>
#include <stdlib.h>

/* Zufallszahlen im Bereich [0.0, 1.0) */
#define frand() ((double) rand() / (RAND_MAX+1.0))


typedef struct input{
  int minsample, maxsample;
  int streamanzahl;
  double start, TotalDur;
  double mindur, maxdur;
  double mintrans, maxtrans;
  double minamp, maxamp;
  double attack, release, pan;
  int anzahl, samplerate, seed;
}INPUT;


void Parameter_Zuweisung(INPUT * in);
int int_zufall(int min, int max);
double float_zufall(double min, double max);
int * intvektor_fuellen(int vektor[], int min,
                        int max, int anzahl);
double * floatvektor_fuellen(double vektor[], double min,
                             double max, int anzahl);
void csound_score (INPUT in, int samplevektor[],
                   double durvektor[], double transvektor[],
                   double ampvektor[]);



/*********
 * MAIN: *
 *********/

main(int argc, char *argv[])
{
  /* argc = ArgumentenAnzahl
   * argc=1 => nur ProgramName, argc=2 => Prog+1Argument
   * argv[0] => ProgramName, argv[1] => 1. Argument */

  INPUT input;
```

```
  int * samplevektor;
  double * durvektor;
  double * transvektor;
  double * ampvektor;


/*  if (argc < 6) { */  /* Mind. 5 Argumente! (Prog+5 = 6) */
  if (argc < 17) {   /* Mind. 16 Argumente! (Prog+16 = 17) */

      fprintf(stderr,"Error: Not enough arguments!\n\n\
Usage: GenEratE  MinSample MaxSample GrainstreamNumber StartTime \
TotalDur\n\
                 [ MinTrans MaxTrans MinAmp(db) MaxAmp(db) Pan(0-1)\n\
                  Seed Samplerate MinDur MaxDur Attack Release ]\n\n\
ACHTUNG: Programm-Nutzung nur mit ALLEN [optionalen] Argumenten \
moeglich!!\n\
\t -> Default-Werte in Parameter_Zuweisung() werden nicht korrekt \
gesetzt/berechnet !?!?!\n\n");
      exit(1);
  }


  /* erforderliche Argumente, die auf der Kommandozeile uebergeben
   * werden muessen
   *
   * atoi => char nach int konvertieren
   * atof => char nach float konvertieren
   */
  input.minsample    = atoi(argv[1]);
  input.maxsample    = atoi(argv[2]);
  input.streamanzahl = atoi(argv[3]);
  input.start        = atof(argv[4]);
  input.TotalDur     = atof(argv[5]);


  /* [optionale Argumente]
   * -> Reihenfolge muss eingehalten werden
   *    (aehnlich einem csound-Opcode)
   */
  if(argc >= 7) { input.mintrans = atof(argv[6]); }
  if(argc >= 8) { input.maxtrans = atof(argv[7]); }

  if(argc >= 9) { input.minamp = atof(argv[8]); }
  if(argc >= 10) { input.maxamp = atof(argv[9]); }

  if(argc >= 11) { input.pan = atof(argv[10]); }

  if(argc >= 12) { input.seed = atoi(argv[11]); }
```

```
  if(argc >= 13) { input.samplerate = atoi(argv[12]); }

  if(argc >= 14) { input.mindur = atof(argv[13]); }
  if(argc >= 15) { input.maxdur = atof(argv[14]); }

  if(argc >= 16) { input.attack = atof(argv[15]); }
  if(argc >= 17) { input.release = atof(argv[16]); }



  /* Default-Werte fuer nicht gesetzte optionale
   * Argumente & Berechnungen durchfuehren
   * (Zeiger auf 'input' wird uebergeben)
   *
   * funktioniert momentan nicht!
   */
  Parameter_Zuweisung(&input);

  samplevektor = intvektor_fuellen(samplevektor, input.minsample,
                                   input.maxsample, input.anzahl);
  durvektor = floatvektor_fuellen(durvektor, input.mindur,
                                  input.maxdur, input.anzahl);
  transvektor = floatvektor_fuellen(transvektor, input.mintrans,
                                    input.maxtrans, input.anzahl);
  ampvektor = floatvektor_fuellen(ampvektor, input.minamp,
                                  input.maxamp, input.anzahl);

  csound_score(input, samplevektor, durvektor, transvektor, ampvektor);

  return 0;
}



/************************
 * Parameter_Zuweisung: *
 ************************/

void Parameter_Zuweisung(INPUT * in)
{
  int SampleAnzahl=0;
  double SndAusschnittsDauer=0.0;
  double avgDur=0.0;

  /* Default-Werte setzen */
  if(!in->mintrans) in->mintrans = 0.7; /* 1.0=orig. Transp. */
  if(!in->maxtrans) in->maxtrans = 1.2; /* 1.0=orig. Transp. */
```

```
if(!in->minamp) in->minamp = 50; /* [db] */
if(!in->maxamp) in->maxamp = 88; /* [db] */

if(!in->pan) in->pan = 0.5; /* 0.0=links, 1.0=rechts */

/* seed fuer frand (-> int_zufall, float_zufall) */
if(!in->seed) in->seed = 12345;
srand(in->seed);

if(!in->samplerate) in->samplerate = 44100;

/* Dauer des SoundFile-Ausschnitts berechnen
 * (-> fuer mindur/maxdur Default-Werte) */
SampleAnzahl = in->maxsample - in->minsample;
SndAusschnittsDauer = (double) SampleAnzahl / in->samplerate;
if(!in->mindur) in->mindur = SndAusschnittsDauer * 0.1; /* [sec] */
if(!in->maxdur) in->maxdur = SndAusschnittsDauer * 0.3; /* [sec] */

if(!in->attack) in->attack = in->mindur*0.1;
if(!in->release) in->release = in->mindur*0.1;

/* Gesamt-Event-Anzahl berechnen */
/* Mittelwert von mindur & maxdur */
avgDur = (in->mindur+in->maxdur)/2;
in->anzahl = (int)(in->TotalDur / avgDur) * in->streamanzahl;

/* maxsample neu berechnen
 * -> maxdur [in Samples] von maxsample abziehen */
in->maxsample = in->maxsample - (int)(in->maxdur*in->samplerate);



/* Parameter-TestAusgabe: (auf stderr) */
fprintf(stderr,"\n");

fprintf(stderr,"Snd-Ausschnitts-Dauer: %f s", SndAusschnittsDauer);

fprintf(stderr,"\n");

fprintf(stderr,"\nargv[1] MinSample:         %d", in->minsample);
fprintf(stderr,"\nargv[2] MaxSample:         %d", in->maxsample);
fprintf(stderr,"\nargv[3] GrainstreamNumber: %d", in->streamanzahl);
fprintf(stderr,"\nargv[4] StartTime:         %f s", in->start);
fprintf(stderr,"\nargv[5] TotalDur:          %f s", in->TotalDur);

fprintf(stderr,"\n");
```

```
  fprintf(stderr,"\nargv[6] Min-Transposition: %f", in->mintrans);
  fprintf(stderr,"\nargv[7] Max-Transposition: %f", in->maxtrans);

  fprintf(stderr,"\nargv[8] Min-Amplitude:     %f dB", in->minamp);
  fprintf(stderr,"\nargv[9] Max-Amplitude:     %f dB", in->maxamp);

  fprintf(stderr,"\nargv[10] Pan-Position:     %f", in->pan);

  fprintf(stderr,"\nargv[11] Random-Seed:      %d", in->seed);

  fprintf(stderr,"\nargv[12] Samplerate:       %d", in->samplerate);

  fprintf(stderr,"\nargv[13] Min-Duration:     %f s", in->mindur);
  fprintf(stderr,"\nargv[14] Max-Duration:     %f s", in->maxdur);

  fprintf(stderr,"\nargv[15] Attack:           %f s", in->attack);
  fprintf(stderr,"\nargv[16] Release:          %f s", in->release);

  fprintf(stderr,"\n");

  fprintf(stderr,"\nEvent-Anzahl: %d", in->anzahl);
  fprintf(stderr,"\n\n");
}



/************************************************************
 * Vektoren mit Hilfe von Zufalls-Transformationen fuellen: *
 ************************************************************/

int int_zufall(int min, int max)
{
  return ((max-min)*frand()) + min;
}



double float_zufall(double min, double max)
{
  return ((max-min)*frand()) + min;
}



int * intvektor_fuellen(int vektor[], int min, int max, int anzahl)
{
  int index=0;
  vektor = malloc(anzahl * sizeof(int));
```

```
  if(vektor == NULL){
    fprintf(stderr,"\n## Kein Speicher fuer 'intvektor'! ##\n");
    exit(1);
  }

  for(index=0; index<anzahl; index++){
    vektor[index] = int_zufall(min, max);
    /*fprintf(stderr,"intvektor[%d]= %d\n", index, vektor[index]);*/
  }
  return vektor;
}



double * floatvektor_fuellen(double vektor[], double min,
                             double max, int anzahl)
{
  int index=0;
  vektor = malloc(anzahl * sizeof(double));
  if(vektor == NULL){
    fprintf(stderr,"\n## Kein Speicher fuer 'floatvektor'! ##\n");
    exit(1);
  }

  for(index=0; index<anzahl; index++){
    vektor[index] = float_zufall(min, max);
    /*fprintf(stderr,"floatvektor[%d]= %f\n", index, vektor[index]);*/
  }
  return vektor;
}



/*****************************
 * csound-sco-Datei erzeugen: *
 *****************************/

void csound_score (INPUT in, int samplevektor[], double durvektor[],
                   double transvektor[], double ampvektor[])
{
  int index=0, streamcounter;


  if(samplevektor!=NULL && durvektor !=NULL && transvektor!=NULL) {

    /* erste Event-Zeile */
    fprintf(stdout,"i 1  %f  %f  %f  %d  %f  %f %f %f\n",
            in.start, durvektor[0], ampvektor[0], samplevektor[0],
```

```
            transvektor[0], in.attack, in.release, in.pan);


    streamcounter=1;
    for(index=1; index<in.anzahl; index++){
      if(streamcounter>=in.anzahl/in.streamanzahl){
        /* neuen Stream bei 'in.start' beginnen */
        fprintf(stdout,
                "i .  %f  %f  %f  %d  %f      .        .        .\n",
                in.start, durvektor[index], ampvektor[index],
                samplevektor[index], transvektor[index]);
        streamcounter=1;
      }
      else{
        /* Event-Start mit '+' definieren, also unmittelbar
         * an das vorhergehende anknuepfen */
        fprintf(stdout,
                "i .   +  %f  %f  %d  %f         .        .        .\n",
                durvektor[index], ampvektor[index],
                samplevektor[index], transvektor[index]);
        streamcounter++;
      }
      /*fprintf(stderr,"%d  ",streamcounter);*/
    }
  }
  else fprintf(stderr,"\n## Keine Daten vorhanden! ##\n");
}

/* EOFrank */
```

B.4 Plugin: pvCross

B.4.1 plugin_pvCross_config.inc.php

```
<?php // plugin_pvCross_config.inc.php

// zusaetzliche Plugin-interne (lokale) Variablen
// (also Var's, die nur innerhalb des Plugins Gueltigkeit haben)

$conf[usr_pvL] = "$usr[filename]-L.pv";
$conf[usr_pvR] = "$usr[filename]-R.pv";

$conf[liFe_pvL] = "$liFe[filename]-L.pv";
$conf[liFe_pvR] = "$liFe[filename]-R.pv";

// $conf[orc] = "$usr[filename].orc";
// $conf[sco] = "$usr[filename].sco";
$conf[orc] = "$config[tmpdir]/pvCross.orc";
$conf[sco] = "$config[tmpdir]/pvCross.sco";

// Name der neu erzeugten Datei wird zurueckgegeben
$conf[newliFe]  = "$liFe[filename]_new.wav";

$conf[liFe_channels] = $liFe[wav_stat][channels];
$conf[usr_channels]  = $usr[wav_stat][channels];

$conf[liFe_dur]  = $liFe[wav_stat][dur];  // [sec]
$conf[usr_dur]   = $usr[wav_stat][dur];  // [sec]

// $conf[growing]   = $config[timestamp] * 0.00000000003;
$conf[growing]   = $config[timestamp] * 0.0000000002;

$file_durations = $conf[liFe_dur] + ($conf[usr_dur] * 0.0001);
$conf[TotalDur] = $file_durations + $conf[growing];  // [sec]

// sr, ksmps & kr -> csound.orc
$conf[sr]    = $config[sr];
$conf[ksmps] = 100;
$conf[kr]    = $conf[sr]/$conf[ksmps];  // 44100/100 = 441


$conf[verbose] = $verbose;

if ($conf[verbose] > 1) { echo "\n\$conf: "; print_r($conf); }

// EOFrank
?>
```

B.4.2 plugin_pvCross.php

```
<?php // plugin_pvCross.php

// ============================================================
// Main-Function
// ============================================================
function plugin_pvCross($liFe, $usr) {
    global $bin, $config, $verbose;

    include("./plugin_pvCross_config.inc.php");


    // pvanal von $usr[wavL,wavR] & $liFe[wavL,wavR]
    // exec_output wird zurueckgeliefert

    // Linker Kanal
    $pvOutput[] = pvanal($usr[wavL], $conf[usr_pvL]);
    // Rechter Kanal (falls existent)
    if ($conf[usr_channels] == 1) { // (also Mono)
        $conf[usr_pvR] = $conf[usr_pvL];
    } else {
        $pvOutput[] = pvanal($usr[wavR], $conf[usr_pvR]);
    }

    // Linker Kanal
    $pvOutput[] = pvanal($liFe[wavL], $conf[liFe_pvL]);
    // Rechter Kanal (falls existent)
    if ($conf[liFe_channels] == 1) { // (also Mono)
        $conf[liFe_pvR] = $conf[liFe_pvL];
    } else {
        $pvOutput[] = pvanal($liFe[wavR], $conf[liFe_pvR]);
    }


    if ($conf[verbose] > 2) {
        echo "\n\$pvOutput: "; print_r($pvOutput);
    }


    // orc+sco
    generate_pvCross_orc($conf);
    generate_pvCross_sco($conf);


    // csound
    $cs_args = "$conf[newliFe] $conf[orc] $conf[sco]";

    $csound_output[0] = "$bin[csound] $cs_args $bin[stderr2stdout]\n";
```

```
    $csound_output[] = `$bin[csound] $cs_args $bin[stderr2stdout]`;


    if ($conf[verbose]) {
        echo "\n\$csound_output: "; print_r($csound_output);
    }


    // temporaere Dateien loeschen
    $unlinkArray = array(
//         $conf[orc],  $conf[sco],
        $conf[usr_pvL],  $conf[usr_pvR],
        $conf[liFe_pvL], $conf[liFe_pvR]
        );
    foreach ($unlinkArray as $tmpFILE) {
        if (is_file($tmpFILE)) { unlink($tmpFILE); }
    }


    // mit sox normalisieren
    // normalize($conf[newliFe]);


    echo "\n\t$conf[newliFe] generated\n";

    // Plugin-Ende, neu erzeugte Datei zurueckgeben
    return $conf[newliFe];

} // end function



// ============================================================
// Sub-Function
// ============================================================
function generate_pvCross_orc($conf) {

    if ($conf[verbose] > 1) {
        echo "\n\t===[   generate_pvCross_orc()   ]===\n";
    }

    // Gesamt=180, indx: 0   1   2   3   4   5  6  7  8  9 10
    $chromosom = array (15, 13, 12, 11, 11, 10, 9, 8, 9, 8, 8,
                        7, 6, 6, 6, 6, 5, 5, 4, 4, 4, 3, 10);
              // indx: 11 12 13 14 15 16 17 18 19 20 21  22

    // zufaellige Neuordnung (Rekombination) der Array-Elemente
    shuffle($chromosom);
```

```
    if ($conf[verbose] > 1) {
        echo "\n\$chromosom: ";
        print_r($chromosom);
        echo "\n";
    }

    $ChromosomAnzahl = sizeof ($chromosom);

    $sum=0;
    // Summe aller $chromosom-Werte errechnen
    foreach ($chromosom as $value) { $sum = $sum + $value; }

    if ($conf[verbose] > 1) {
        echo "\n\$ChromosomAnzahl = $ChromosomAnzahl\n\$sum = $sum\n\n";
    }

    $x=0;
    foreach ($chromosom as $indx => $wert) {
        $x = $x + $wert;
        // Werte aufsteigend von 0 bis 1 (letzter = 1, weil $x=$sum)
        $ChromoIndx[$indx] = ($x/$sum);
        if ($conf[verbose] > 1) {
            echo "\$x = $x , ";
            echo "\$ChromoIndx[$indx] = $ChromoIndx[$indx]\n";
        }
    }

    if ($conf[verbose] > 1) {
        echo "\n\n\$ChromoIndx: "; print_r($ChromoIndx);
    }

    $orc_file = fopen ("$conf[orc]", "w");
    fwrite ($orc_file, "
; pvCross.orc

sr     = $conf[sr]
kr     = $conf[kr]
ksmps  = $conf[ksmps]
nchnls = 2


;*******;
 instr 1
;*******;

idur        = p3
ivol        = p4
```

```
iCrossTable = p5

iTabSize = 23   ; Elemente-Anzahl von iCrossTable

iliFeDur = $conf[liFe_dur]
iUserDur = $conf[usr_dur]

kphasor phasor 1/idur ; ein table-Durchgang in p3 sec

kfmod  linseg 1, idur, 1

kcross  tablei  kphasor*iTabSize, iCrossTable
kliFeAmp = 1-kcross ; 0 => liFe
kUserAmp = kcross   ; 1 => user-File

");


// linseg 0,idur/$ChromosomAnzahl,$ChromIndx[0],idur/$ChromosomAnzahl, ... ,1
    fwrite($orc_file, "kindx linseg 0, ");
    for($i=0; $i<$ChromosomAnzahl-1; $i++) {
        fwrite ($orc_file, "idur/$ChromosomAnzahl, $ChromoIndx[$i], ");
    }
    fwrite ($orc_file, "idur/$ChromosomAnzahl, 1");


    fwrite ($orc_file, "

kUserPointer = kindx * iUserDur
kliFePointer = kindx * iliFeDur

     pvbufread kUserPointer, \"$conf[usr_pvL]\"
autL pvcross kliFePointer, kfmod, \"$conf[liFe_pvL]\", kUserAmp, kliFeAmp

     pvbufread kUserPointer, \"$conf[usr_pvR]\"
autR pvcross kliFePointer, kfmod, \"$conf[liFe_pvR]\", kUserAmp, kliFeAmp

alinks  = autL * ivol
arechts = autR * ivol

outs alinks, arechts

endin

");
    fclose ($orc_file);

    echo "\t$conf[orc] generated\n";
```

```
} // end function



// ============================================================
// Sub-Function
// ============================================================
function generate_pvCross_sco($conf) {

    if ($conf[verbose] > 1) {
        echo "\n\t===[   generate_pvCross_sco()   ]===\n";
    }

    // je groesser die Zahl, desto mehr Nachkommastellen
//    $rnd_max = 999999999;
    $rnd_max = 10;

    for ($i=0; $i<23; $i++) {
        // Zufallszahl zwischen 0 und 1
        $rnd_cross[$i] = mt_rand(0,$rnd_max) / $rnd_max;
        // Zufallszahl 0 oder 1
//          $rnd_cross[$i] = mt_rand(0,1);
    }

    if ($conf[verbose] > 1) {
        echo "\n\$rnd_cross: "; print_r($rnd_cross);
    }

    $sco_file = fopen ("$conf[sco]", "w");
    fwrite ($sco_file, "
; pvCross.sco

; zufaellig generierter CrossTable (0 = liFe, 1 = User):
");


    // Zufalls-Cross-Table
    fwrite($sco_file, "f22 0 32 -2 ");
    foreach ($rnd_cross as $x) {  fwrite ($sco_file, "$x ");  }


    fwrite ($sco_file, "

;instr start   dur       vol CrossTable
;             (p3        p4     p5)
 i 1    0  $conf[TotalDur]  1     22
```

```
e

");

    fclose ($sco_file);

    echo "\t$conf[sco] generated\n";

} // end function


// EOFrank
?>
```

B.5 Plugin: template

B.5.1 plugin_template_config.inc.php

```
<?php // plugin_template_config.inc.php

// zusaetzliche Plugin-interne (lokale) Variablen
// (also Var's, die nur innerhalb des Plugins Gueltigkeit haben)

// Name der neu erzeugten Datei wird zurueckgegeben
$conf[newliFe] = "$liFe[filename]_new.wav";

$conf[verbose] = $verbose;

//if ($conf[verbose] > 1) {
    echo "\n\$conf: "; print_r($conf);
//}

// EOF
?>
```

B.5.2 plugin_template.php

```
<?php // plugin_template.php

// =============================================================
// Main-Function
// =============================================================
function plugin_template($liFe, $usr) {
    global $bin, $config, $verbose;

    include("./plugin_template_config.inc.php");

    echo "\navailable variables:\n";
    echo "\n\$config: "; print_r($config);
    echo "\n\$bin: "; print_r($bin);
    echo "\n\$liFe: "; print_r($liFe);
    echo "\n\$usr: "; print_r($usr);

//    normalize($conf[newliFe]);

//    echo "\n\t$conf[newliFe] generated\n";

    // Plugin-Ende, neu erzeugte Datei zurueckgeben
//    return $conf[newliFe];

    // keine weiteren Aktionen -> liFe.wav unangetastet lassen
```

```
    return FALSE;

} // end function

// EOF
?>
```

C Client

C.1 client_config.inc.php

```
<?php // client_config.inc.php

// hier gesetzte Default-Variablen werden dem Server mituebergeben
// (auch wenn das entsprechende --Argument nicht gesetzt werden kann)
$default = array(
    "host"   => "127.0.0.1", // localhost
    "port"   => "12965" // Port: Life -> l=12 i=9 f=6 e=5 --> 12965
    );


// verfuegbare Optionen, die als '--Argument Wert' gesetzt
// werden koennen/duerfen
$options = array("host","port","plugin","ftpuser","email",
                 "verbose","resize");

// Argumente, mit denen die Hilfe ($Usage) aufgerufen werden kann
$help_options = array("--help", "-help", "-h");


$tryHelp = "Try `$argv[0] --help' for more information.\n";

$Usage = "Usage:\t$argv[0] [OPTION]... <URL> [OPTION]...

options:
\t--host HOST\t transmit URL to server on HOST (default: $default[host])
\t--port PORT\t connect to PORT on host (default: $default[port])
\t--plugin PLUGIN\t use PLUGIN for soundprocessing
\t--resize MODE\t use `truncate' or `stretch' to resize the sound-files
\t\t\t (default: stretch)
\t--email ADR\t server sends email to ADR when soundprocessing finished
\t--ftpuser NAME\t use NAME to log in ftp-Server
\t\t\t if this option is not given, anonymous ftp will be used
\t\t\t (URL should certainly start with 'ftp://')
\t--verbose INT\t set verbose level for client AND server
\t\t\t (as INTeger number > 0)
\t--help, -h\t display this help and exit

<URL>:\thttp://some.server.org/file
\tftp://some.server.org/file
\tftp://user:password@some.server.org/file (NOT recommended!)
\tfile:///path/to/local/file
\t/path/to/local/file

Note:\tlocal file has to be on the server's host!
```

```
\tfile upload only available with the Web-client (under construction ;-)


currently supported filetypes: MPEG Layer I/II/III (mp3), ogg-vorbis, wav

";


// EOFrank
?>
```

C.2 client.php

```
#! ../organiSm/php5 -c ../organiSm/php5.ini
<?php // client.php

include("./client_config.inc.php");

// uebergebene Argumente verarbeiten/ueberpruefen
// ($argv[0] -> Programm-Name, $argv[1] -> erstes Argument, ...)
//echo "\$argc: $argc\n\$argv: "; print_r($argv);

// falls Anzahl der Argumente < 2
// (also Programm ohne Argumente aufgerufen wurde)
if ($argc < 2) {
    echo "ERROR: too few arguments\n";
    exit($tryHelp);
}

// 1 -> beim ersten Argument beginnen (Programm-Name ueberspringen)
$i = 1;
while ($i < $argc) {
    if (preg_match("/^--?([-a-zA-Z]+)/", $argv[$i], $match)) {
        //echo "\$match[1]: $match[1]\n";
        $option_orig = $argv[$i];
        $option_name = $match[1]; // Name ohne '--'

        // naechstes Argument, welches der zu setzende Wert sein sollte
        $i++;
        $value = $argv[$i];
        $option[$option_name] = $value;

        if (in_array($option_orig, $help_options)) {
            exit($Usage);
        }
        if (!in_array($option_name, $options)) {
```

```
            echo "ERROR: unknown option ‘$option_orig’\n";
            exit($tryHelp);
        }
        if (empty($value)) {
            echo "ERROR: missing value for ‘$option_name’\n";
            exit($tryHelp);
        }
        echo "option set: ‘$option_name’\t->  ‘$value’\n";
        $i++;
    } else {
        $datei = $argv[$i];
        if ($option[verbose] > 2 ) {
            echo "\n\$datei bekommt den Wert: $argv[$i]\n";
        }
        $i++;
    }
}


// Default-Werte setzen
foreach ($default as $defName => $defWert) {
    if (empty($option[$defName])) { $option[$defName] = $defWert; }
}


// ftp-Passwort abfragen, falls ‘--ftpuser username’
if (!empty($option[ftpuser])) {
    $ftpWARNING  = "WARNING: The password you have to type in,";
    $ftpWARNING .= " appears on the screen!\n";
    $ftpWARNING .= "\t So close this window, after you are done";
    $ftpWARNING .= " (or hit <CTRL>-c to abort now).\n";
    echo $ftpWARNING;
    echo "Password for ‘$option[ftpuser]’: ";
    $input = fscanf(STDIN, "%s");
    echo "[[[ not implemented yet! ]]] Pass: $input[0]\n";
    $option[ftppass] = $input[0];
}


if (empty($datei)) {
    echo "ERROR: missing URL\n";
    exit($tryHelp);
}


if ($option[verbose] > 1) {
    echo "\n\$option: "; print_r($option);
}
```

```
// String, der an Server uebermittelt werden soll
// (+ Optionen, durch Leerzeichen getrennt)

foreach ($option as $optName => $optWert) {
    $option_string .= " $optName $optWert";
}

$transmission = "file $datei" . $option_string;

if ($option[verbose]) {
    echo "\n\$transmission: ‘$transmission’\n\n";
}

$host = $option[host];
$port = $option[port];

// mit Server verbinden
// @ -> keine php-Fehlermeldung ausgeben
//      -> Fehler werden in $errno & $errstr geschrieben
$connection = @stream_socket_client("tcp://$host:$port",
                                    $errno, $errstr);

if (!$connection) {
    echo "ERROR: $errno - $errstr\n";
} else {
    fwrite($connection, "$transmission");
    echo "‘$datei’ transmitted to server ($host:$port)\n";
    fclose($connection);
}

// EOFrank
?>
```