Spaß mit dem Ravensburger Tiptoi® Stift

Quellen

- Homepage von ''tttool''
- Installation von `tttool`

Tiptoi® als MP3 Player

Leider kann der Ravensburger TipToi® Stift keine MP3 Dateien wie ein klassischer MP3-Player abspielen. Ravensbrger begründet das in seinen FAQ wie folgt:
Zitat:
Frage: Kann ich auch Audioinhalte von anderen Abspielgeräten auf den tiptoi® Stift mit Player ziehen?
Antwort: Um eine verlässliche Qualität der Audioinhalte zu gewährleisten, haben wir uns für ein geschlossenes System mit eigenem Dateiformat entschieden. Dies hat den Vorteil, dass keine von Eltern ungewünschten Inhalte auf den Stift gelangen. Daher ist es nicht möglich, eigene Hörbücher und Lieder auf dem tiptoi® Stift zu speichern.\\
Schade - aber nicht zu ändern.
Zum Glück gibt es Joachim “nomeata” Breitner der die Software ''tttool'' erstellt hat. Mit Hilfe dieser Software ist es möglich, eigene Audiodateien z.B. Hörbücher o.ä. auf den TipToi® Stift zu bringen.
Mehr dazu auf seiner Webseite.

Installation von ''tttool''

Die Installation von `tttool` wird u.a. hier verständlich erklärt. Daher gehe ich hier nicht näher darauf ein.
Im Folgenden sind lediglich meine Installationsschritte aufgeführt. Ich arbeite ausschließlich auf einem Ubuntu 16.04 LTS 64bit 😉

```
git clone https://github.com/entropia/tip-toi-reveng.git ~/bin/tttool_src
cd ~/bin/tttool_src
# falls noch nicht installiert:
apt-get install haskell-platform libncurses5-dev
cabal update
cabal install --only-dependencies
cabal install --bindir=.
cd ~/bin
ln -s tttool_src/tttool tttool
```

Ein Shell Script das ALLES macht

Voraussetzungen

Folgende Programmen müssen zur erfolgreichen

Abarbeitung des folgenden Skriptes installiert sein:

- `tttool` ist installiert und im Suchpfad (z.B. `/bin`) enthalten
- Das Grafikmanipulationspaket `ImageMagick` ist min. in Version 7.x installiert. Bis Version 6.8.9.x gibt es leider einen Bug im Tool `convert` welches für die Erzeugung verschiedener Grafikelemente verwendet wird. Der Bug führt zu einer fehlerhaften Grafik.
- Das Programm `oggenc` zum kodieren vom Wave- ins Ogg-Format
- `rec`, ein Kommandozeilen-Tool zum erstellen einer „Stille“-Datei

Automatisch generierte Grafik

Einzig folgende Dateien müssen vom Benutzer bereit gestellt werden:

- die Quelldateien (*.wav) z.B. eines Hörspieles
- eine Bilddatei, z.B. das passende CD Cover (meistens leicht per Google zu finden)
- eine Begrüßungsdatei im Wave- oder Ogg-Format

Die Begrüßungsdatei sollte möglichst nicht länger als ein paar Sekunden. Diese Datei wird beim Aktivieren des Hörbuches, also beim „berühren“ des entsprechenden `Start`-Symbols abgespielt - quasi als akustisches Feedback für Kinder. Damit sie wissen was sie ausgewählt haben.
Ich erstelle diese Begrüßungsdatei höufig aus den ersten Sekunden eines Titelliedes oder dem ersten Track einer Hörbuch-CD.

Skriptaufruf

Gestartet wird das Skript z.B. wie folgt:

```
./gen_gme.sh -w ../tiptoi_test -c "Kokosnuss - Expedition auf dem Nil" -pid 713 -d 2
```

Die Parameter im einzelnen:

- -c: „Comment“ - also ein Kommentar für die YAML-Datei, wird auch als Dateiname für die GME-Datei verwendet. Bei Verwendung von Leerzeichen bitte den Kommentar in Anführungszeichen setzten
- -w: „Working Directory“ oder „Arbeitsverzeichnis“, hier liegt der Ordner `WAV` mit den Quelldateien z.B. eines Hörspieles. Auch hier gilt: bei Verwendung von Leerzeichen bitte den Verzeichnispfad in Anführungszeichen setzten
- -pid: „Product ID“ - eindeutige ID des eingenen/neuen Produktes. Wird benötigt zur eindeutigen Zuordnung der GME-Datei. Die Product-ID darf noch nicht von einem Produkt das sich bereits auf dem TipToi Stift befindet verwendet werden. Anderenfalls gibt es durch die Doppeldeutigkeit Probleme bei der Produktauswahl und bei der Steuerung.

- -d: „Debug Level“ - wenn mal was nicht klappt kann mit diesem Parameter evtl. hilfreiche DEbug-Ausgaben erzeugt werden

Ausgaben des Skriptes

- Eine GME-Datei - sie muss ins Wurzelverzeichnis des TipToi® Stiftes kopiert werden
- Eine *.png Datei - eine Grafikdatei zur „Steuerung“ des TipToi® Stiftes, sie ist mit einer Auflösung von **600 dpi** erstellt und **muss unbedingt in dieser Auflösung gedruckt werden** (z.B. mit Hilfe von GIMP). Anderenfalls kann der TipToi® Stifte die Punktmuster der OIDs nicht fehlerfrei lesen.
- Die *.yaml Datei welche für die Erstellung der *.gme Datei verwendet wurde

Quellcode

gen_gme.sh

```
#!/bin/bash
#
# Input files, supplied by user:
COVER="cover.jpg"
OPENING_SOUND="Titel"

NULLSOUND="silence"

SHEETSIZE="7016x4960"

BUTTON_H_POSITION=4015
BUTTON_V_POSITION=600
BUTTON_H_OFFSET=0
BUTTON_V_OFFSET=170

OID_H_POSITION=4015
OID_V_POSITION=600
OID_H_OFFSET=0
OID_V_OFFSET=170

TEXT_H_POSITION=4250
TEXT_V_POSITION=700
TEXT_H_OFFSET=0
TEXT_V_OFFSET=170

PRODUCT_ID_H_POS=5100
PRODUCT_ID_V_POS=4800

COPYRIGHT_H_POSITION=6000
COPYRIGHT_V_POSITION=4800

COVER_MAX_WIDTH=3300
COVER_MAX_HEIGHT=3300
```

```
COVER_SIZE="${COVER_MAX_WIDTH}x${COVER_MAX_HEIGHT}"

TRACKBUTTON_WIDTH=189
TRACKBUTTON_HEIGHT=142
TRACKBUTTONSIZE="${TRACKBUTTON_WIDTH}x${TRACKBUTTON_HEIGHT}"
CONTROLBUTTON_WIDTH=600
CONTROLBUTTON_HEIGHT=600
CONTROLBUTTONSIZE="${CONTROLBUTTON_WIDTH}x${CONTROLBUTTON_HEIGHT}"
BUTTON_STROKEWIDTH=12
BUTTON_FONT_SIZE=120
TEXT_FONT_SIZE=$BUTTON_FONT_SIZE

PRODUCT_ID=""
STARTSCRIPTCODE=7000
COMMENT=""

TRACKS=0
COUNTER=0

DEBUG_MAX_LEVEL=3
DEBUG_LEVEL=0

WORKFOLDER=""
MESSAGE_METHOD=""
TRACKLIST="tracklist.txt"

# special folders to use:
#TEMPLATES="templates"
OIDS="oids"
OGGS="ogg"
WAVES="WAV"

# Output file for printing
OUTPUT=""

# tttool config file to generate:
YAML_FILE=""



# Functions needed to do all jobs ;-)

check_tools ()
{
command -v $1 >/dev/null 2>&1 || { echo "The required tool $1 is not
installed. Aborting." >&2; exit 1; }

}


debug ()
```

```
# Print debug output if needed
{
MESSAGE=$1
DEBUG_MESSAGE_LEVEL=$2
LINEFEED=$3

if [ $DEBUG_LEVEL -le -1 ] || [ $DEBUG_LEVEL -gt $DEBUG_MAX_LEVEL ] ;
then
   echo "Error - wrong debug level selected - EXIT!"
   exit 1
else
  if [ $((DEBUG_MESSAGE_LEVEL)) -le $((DEBUG_LEVEL)) ]; then
    echo $LINEFEED "$MESSAGE"
  fi
fi
}

desktop_message ()
{
MESSAGE=$1

if [ ! -z `command -v notify-send` ]; then
   notify-send "$MESSAGE"
else
  if [ ! -z `command -v zenity` ]; then
    zenity  --notification  --text "$MESSAGE"
  else
    echo "$MESSAGE"
  fi
fi
}


cleanup ()
# Clean up all temporarily generated files
{
debug "Cleaning up $1 ... " 1 -n
  case $1 in
        output)
        if [ -f $WORKFOLDER/$OUTPUT ]; then
          debug " - \"$WORKFOLDER/$OUTPUT\" - " 2 -n
          rm $WORKFOLDER/$OUTPUT
          rm $WORKFOLDER/$YAML_FILE
          rm $WORKFOLDER/*.gme
        else
          debug " - no \"$WORKFOLDER/$OUTPUT\" file found - " 2 -n
        fi
     ;;
      tracklist)
        if [ -e $WORKFOLDER/$TRACKLIST ]; then
          debug " - \"$WORKFOLDER/$TRACKLIST\" - " 2 -n
```

```
          rm $WORKFOLDER/$TRACKLIST
        else
          debug " - no \"$WORKFOLDER/$TRACKLIST\" file found - " 2 -n
        fi
      ;;
      cover)
        if [ -e $WORKFOLDER/cover_resized.jpg ]; then
          debug " - \"$WORKFOLDER/cover_resized.jpg\" - " 2 -n
          rm $WORKFOLDER/cover_resized.jpg
        else
          debug " - no \"$WORKFOLDER/cover_resized.jpg\" file found - "
2 -n
        fi
      ;;
      ogg)
        if [ -d $WORKFOLDER/$OGGS ]; then
          debug " - \"$WORKFOLDER/$OGGS\" folder - " 2 -n
          rm $WORKFOLDER/$OGGS/*.ogg
          rmdir $WORKFOLDER/$OGGS
        else
          debug " - no \"$WORKFOLDER/$OGGS\" folder found - " 2 -n
        fi
      ;;
      oids)
        if [ -d $WORKFOLDER/$OIDS ]; then
          debug " - \"$WORKFOLDER/$OIDS\" folder - " 2 -n
          rm $WORKFOLDER/$OIDS/*.png
          rmdir $WORKFOLDER/$OIDS
        else
          debug " - no \"$WORKFOLDER/$OIDS\" folder found - " 2 -n
        fi
      ;;
      *)
      ;;
  esac
debug " done" 1
}

mk_ogg_files()
{
  if [ $DEBUG_LEVEL -le 1 ]; then
    debug "Makeing *.ogg files ..." 1 -n
  else
    debug "Makeing *.ogg files ..." 1
  fi

  if [ ! -d "$WORKFOLDER/$OGGS" ]; then
    mkdir -p $WORKFOLDER/$OGGS
  fi

  if [ -d "$WORKFOLDER/$WAVES" ]; then
```

```
  for wavefile in $WORKFOLDER/$WAVES/*.wav
    do
      COUNTER=$((COUNTER+1))
      debug "File to prepare: $wavefile ... " 2 -n
      basename "$wavefile" | cut -f2 -d'/' | cut -f1 -d'.' | sed -e
's/_/ /g' >> $WORKFOLDER/$TRACKLIST
      oggenc -Q -b 128 "${wavefile}" -o $WORKFOLDER/$WAVES/$(printf
"Track_%02d.ogg" $COUNTER)
      mv "$WORKFOLDER/$WAVES/$(printf "Track_%02d.ogg" $COUNTER)"
$WORKFOLDER/$OGGS/
      debug " done" 2
    done
  else
    echo "Source folder \"$WORKFOLDER/$WAVES\" for wave files not found
-> EXIT!"
    exit 1
  fi

  # save number of tracks
  TRACKS=$COUNTER

  if [ ! -f $WORKFOLDER/${NULLSOUND}.ogg ]; then
    debug "File \"$WORKFOLDER/${NULLSOUND}.ogg\" not found - generating
it" 2
    rec $WORKFOLDER/${NULLSOUND}.wav trim 0 0.1 2> /dev/null
    oggenc -Q -b 128 $WORKFOLDER/${NULLSOUND}.wav > /dev/null
    rm $WORKFOLDER/${NULLSOUND}.wav
  fi
  if [ -f $WORKFOLDER/$OGGS/${NULLSOUND}.ogg ]; then
    rm $WORKFOLDER/$OGGS/${NULLSOUND}.ogg
  fi
  cp $WORKFOLDER/${NULLSOUND}.ogg $WORKFOLDER/$OGGS/${NULLSOUND}.ogg

  if [ ! -f $WORKFOLDER/${OPENING_SOUND}.wav ] && [ ! -f
$WORKFOLDER/${OPENING_SOUND}.ogg ] ; then
    echo "Error: Opening sound \"${OPENING_SOUND}.wav\" or
\"${OPENING_SOUND}.ogg\" in \"$WORKFOLDER\" not found -> EXIT!"
    exit 1
  else
    if [ -f $WORKFOLDER/${OPENING_SOUND}.wav ] && [ ! -f
$WORKFOLDER/${OPENING_SOUND}.ogg ]; then
      debug "Encoding \"${OPENING_SOUND}.wav\" ... " 2 -n
      oggenc -Q -b 128 $WORKFOLDER/${OPENING_SOUND}.wav > /dev/null
      debug " done" 2
    fi
  fi
  cp $WORKFOLDER/${OPENING_SOUND}.ogg $WORKFOLDER/$OGGS
  debug " done" 1
}

mk_yaml_file()
```

```
{
  if [ $DEBUG_LEVEL -le 1 ]; then
    debug "Generating \"$WORKFOLDER/$YAML_FILE\" ... " 1 -n
  else
    debug "Generating \"$WORKFOLDER/$YAML_FILE\" ... " 1
  fi

  if [ -f $WORKFOLDER/$YAML_FILE ]; then
    rm $WORKFOLDER/$YAML_FILE
  fi

  echo "product-id: $PRODUCT_ID" >> $WORKFOLDER/$YAML_FILE
  echo "media-path: ${OGGS}/%s"  >> $WORKFOLDER/$YAML_FILE
  echo comment: $COMMENT  >> $WORKFOLDER/$YAML_FILE
  echo "init: \$track := 0" >> $WORKFOLDER/$YAML_FILE
  echo "welcome: `eval basename $WORKFOLDER/$OPENING_SOUND | cut -f1 -
d'.'`" >> $WORKFOLDER/$YAML_FILE
  echo "scripts:" >> $WORKFOLDER/$YAML_FILE
  echo "  play:" >> $WORKFOLDER/$YAML_FILE
  COUNTER=0
  echo "  - \$track == $COUNTER? \$track := $((COUNTER+1)) J(play)" >>
$WORKFOLDER/$YAML_FILE
  COUNTER=$((COUNTER+1))

  while [ $COUNTER -le $TRACKS ]; do
    echo "  - \$track == $COUNTER? P($(printf "Track_%02d" $COUNTER))
\$track := $((COUNTER+1)) J(play)" >> $WORKFOLDER/$YAML_FILE
    COUNTER=$((COUNTER+1))
  done
  echo "  - \$track == $COUNTER? P(`eval basename
$WORKFOLDER/$NULLSOUND | cut -f1 -d'.'`) \$track := 0 J(play)" >>
$WORKFOLDER/$YAML_FILE
  echo "  previous:" >> $WORKFOLDER/$YAML_FILE
  echo "  - \$track > 2? P(`eval basename $WORKFOLDER/$NULLSOUND | cut
-f1 -d'.'`) \$track -= 2 J(play)" >> $WORKFOLDER/$YAML_FILE
  echo "  next:" >> $WORKFOLDER/$YAML_FILE
  echo "  - P(`eval basename $WORKFOLDER/$NULLSOUND | cut -f1 -d'.'`)
J(play)" >> $WORKFOLDER/$YAML_FILE
  echo "  stop:" >> $WORKFOLDER/$YAML_FILE
  echo "  - P(`eval basename $WORKFOLDER/$NULLSOUND | cut -f1 -d'.'`)
\$track := 0" >> $WORKFOLDER/$YAML_FILE

  COUNTER=1
  while [ $COUNTER -le $TRACKS ]; do
    echo "  $(printf "track_%02d" $COUNTER):" >> $WORKFOLDER/$YAML_FILE
    echo "  - P(`eval basename $WORKFOLDER/$NULLSOUND | cut -f1 -d'.'`)
\$track := $COUNTER J(play)" >> $WORKFOLDER/$YAML_FILE
    COUNTER=$((COUNTER+1))
  done

  echo "scriptcodes:" >> $WORKFOLDER/$YAML_FILE
```

```
  echo "  play: $((STARTSCRIPTCODE+1))" >> $WORKFOLDER/$YAML_FILE
  echo "  previous: $((STARTSCRIPTCODE+2))" >> $WORKFOLDER/$YAML_FILE
  echo "  next: $((STARTSCRIPTCODE+3))" >> $WORKFOLDER/$YAML_FILE
  echo "  stop: $((STARTSCRIPTCODE+4))" >> $WORKFOLDER/$YAML_FILE

  COUNTER=1
  while [ $COUNTER -le $TRACKS ]; do
    echo "  $(printf "track_%02d" $COUNTER):
$((STARTSCRIPTCODE+4+$COUNTER))" >> $WORKFOLDER/$YAML_FILE
    COUNTER=$((COUNTER+1))
  done
  debug " done" 1
}

mk_oids()
{
  if [ $DEBUG_LEVEL -le 1 ]; then
    debug "Generating OIDs ..." 1 -n
  else
    debug "Generating OIDs ..." 1
  fi

  if [ ! -d $WORKFOLDER/$OIDS ]; then
    mkdir -p $WORKFOLDER/$OIDS
  fi
  debug "Using YAML-File: $WORKFOLDER/$YAML_FILE" 2
  if [ $DEBUG_LEVEL -le 1 ]; then
    tttool --dpi 600 --code-dim 8x6 oid-codes $WORKFOLDER/$YAML_FILE >
/dev/null
  else
    tttool --dpi 600 --code-dim 8x6 oid-codes $WORKFOLDER/$YAML_FILE
  fi
  mv *track*.png $WORKFOLDER/$OIDS
  if [ $DEBUG_LEVEL -le 1 ]; then
    tttool --dpi 600 --code-dim 25x25 oid-codes $WORKFOLDER/$YAML_FILE
> /dev/null
  else
    tttool --dpi 600 --code-dim 25x25 oid-codes $WORKFOLDER/$YAML_FILE
  fi
  rm *track*.png
  mv *.png $WORKFOLDER/$OIDS
  debug " done" 1
}

mk_gme()
{
  debug "Generate GME-file ... " 1 -n
  tttool assemble $WORKFOLDER/$YAML_FILE
  debug " done" 1
}
```

```
mk_base_image()
{
  debug "Generate printable image ..." 1 -n
  convert $WORKFOLDER/$COVER -resize $COVER_SIZE\!
$WORKFOLDER/cover_resized.jpg
  convert -size $SHEETSIZE xc:white $WORKFOLDER/$OUTPUT

  convert -size $CONTROLBUTTONSIZE xc:white -fill white -stroke black -
strokewidth 30 -draw "roundrectangle 15,15 585,585 30,30\
stroke-linecap round line 150,150 450,300\
stroke-linecap round line 450,300 150,450\
stroke-linecap round line 150,450 150,150" $WORKFOLDER/play.png

  convert -size $CONTROLBUTTONSIZE xc:white -fill white -stroke black -
strokewidth 30 -draw "roundrectangle 15,15 585,585 30,30\
stroke-linecap round line 150,150 450,150\
stroke-linecap round line 450,150 450,450\
stroke-linecap round line 450,450 150,450\
stroke-linecap round line 150,450 150,150" $WORKFOLDER/stop.png

  convert -size $CONTROLBUTTONSIZE xc:white -fill white -stroke black -
strokewidth 30 -draw "roundrectangle 15,15 585,585 30,30\
stroke-linecap round line 150,150 200,150\
stroke-linecap round line 200,150 200,300\
stroke-linecap round line 200,300 450,150\
stroke-linecap round line 450,150 450,450\
stroke-linecap round line 450,450 200,300\
stroke-linecap round line 200,300 200,450\
stroke-linecap round line 200,450 150,450\
stroke-linecap round line 150,450 150,150" $WORKFOLDER/previous.png

  convert -size $CONTROLBUTTONSIZE xc:white -fill white -stroke black -
strokewidth 30 -draw "roundrectangle 15,15 585,585 30,30\
stroke-linecap round line 150,150 400,300\
stroke-linecap round line 400,300 400,150\
stroke-linecap round line 400,150 450,150\
stroke-linecap round line 450,150 450,450\
stroke-linecap round line 450,450 400,450\
stroke-linecap round line 400,450 400,300\
stroke-linecap round line 400,300 150,450\
stroke-linecap round line 150,450 150,150" $WORKFOLDER/next.png

  convert -size $CONTROLBUTTONSIZE xc:white -fill white -stroke black -
strokewidth 30 -draw "roundrectangle 15,15 585,585 30,30"\
                                                    -stroke black -
strokewidth 50 -draw "stroke-linecap round ellipse 300,300 200,200
305,235"\
                                                    -stroke black -
strokewidth 50 -draw "stroke-linecap round line 300,65 300,180"
$WORKFOLDER/on.png
```

```
  composite -geometry  +600+1200 $WORKFOLDER/cover_resized.jpg
$WORKFOLDER/$OUTPUT $WORKFOLDER/$OUTPUT

  composite -geometry  +600+600 $WORKFOLDER/on.png $WORKFOLDER/$OUTPUT
$WORKFOLDER/$OUTPUT
  rm $WORKFOLDER/on.png
  composite -geometry  +600+600 $WORKFOLDER/$OIDS/oid-${PRODUCT_ID}-
START.png $WORKFOLDER/$OUTPUT $WORKFOLDER/$OUTPUT

  composite -geometry  +1275+600 $WORKFOLDER/play.png
$WORKFOLDER/$OUTPUT $WORKFOLDER/$OUTPUT
  rm $WORKFOLDER/play.png
  composite -geometry  +1275+600 $WORKFOLDER/$OIDS/oid-${PRODUCT_ID}-
play.png $WORKFOLDER/$OUTPUT $WORKFOLDER/$OUTPUT

  composite -geometry  +1950+600 $WORKFOLDER/stop.png
$WORKFOLDER/$OUTPUT $WORKFOLDER/$OUTPUT
  rm $WORKFOLDER/stop.png
  composite -geometry  +1950+600 $WORKFOLDER/$OIDS/oid-${PRODUCT_ID}-
stop.png $WORKFOLDER/$OUTPUT $WORKFOLDER/$OUTPUT

  composite -geometry  +2625+600 $WORKFOLDER/previous.png
$WORKFOLDER/$OUTPUT $WORKFOLDER/$OUTPUT
  rm $WORKFOLDER/previous.png
  composite -geometry  +2625+600 $WORKFOLDER/$OIDS/oid-${PRODUCT_ID}-
previous.png $WORKFOLDER/$OUTPUT $WORKFOLDER/$OUTPUT

  composite -geometry  +3300+600 $WORKFOLDER/next.png
$WORKFOLDER/$OUTPUT $WORKFOLDER/$OUTPUT
  rm $WORKFOLDER/next.png
  composite -geometry  +3300+600 $WORKFOLDER/$OIDS/oid-${PRODUCT_ID}-
next.png $WORKFOLDER/$OUTPUT $WORKFOLDER/$OUTPUT

  convert $WORKFOLDER/$OUTPUT -font Times-Roman -pointsize
$TEXT_FONT_SIZE -draw "fill black text
+$PRODUCT_ID_H_POS+$PRODUCT_ID_V_POS 'Product ID: $PRODUCT_ID"
$WORKFOLDER/$OUTPUT
  convert $WORKFOLDER/$OUTPUT -font Times-Roman -pointsize
$TEXT_FONT_SIZE -draw "fill black text
+$COPYRIGHT_H_POSITION+$COPYRIGHT_V_POSITION 'Created by ChrvTh"
$WORKFOLDER/$OUTPUT
  debug "  done" 1
}

mk_trackbuttons()
{
  COUNTER=1
  debug "Generating track buttons and adding them to printable image
..." 1 -n
  while [ $COUNTER -le $TRACKS ]; do
```

```
    debug "COUNTER: $COUNTER" 3
    debug "TRACKS: $TRACKS" 3
    debug "BUTTON H: $BUTTON_H_POSITION" 3
    debug "BUTTON V: $BUTTON_V_POSITION" 3

    convert -size $TRACKBUTTONSIZE xc:white -fill white -stroke black -
strokewidth $BUTTON_STROKEWIDTH -draw "roundrectangle 0,0
$TRACKBUTTON_WIDTH,$TRACKBUTTON_HEIGHT 30,30" $WORKFOLDER/$(printf
"track_%02d" $COUNTER)_button.png
    convert $WORKFOLDER/$(printf "track_%02d" $COUNTER)_button.png -
font Arial -pointsize $BUTTON_FONT_SIZE -draw "fill black text +30+115
'$(printf "%02d" $COUNTER)" $WORKFOLDER/$(printf "track_%02d"
$COUNTER)_button.png

    composite -geometry  +$BUTTON_H_POSITION+$BUTTON_V_POSITION
$WORKFOLDER/$(printf "track_%02d" $COUNTER)_button.png
$WORKFOLDER/$OUTPUT $WORKFOLDER/$OUTPUT
    rm $WORKFOLDER/$(printf "track_%02d" $COUNTER)_button.png
    composite -geometry  +$OID_H_POSITION+$OID_V_POSITION
$WORKFOLDER/$OIDS/oid-$PRODUCT_ID-$(printf "track_%02d" $COUNTER).png
$WORKFOLDER/$OUTPUT $WORKFOLDER/$OUTPUT
    rm $WORKFOLDER/$OIDS/oid-$PRODUCT_ID-$(printf "track_%02d"
$COUNTER).png
    convert $WORKFOLDER/$OUTPUT -font Times-Roman -pointsize
$TEXT_FONT_SIZE -draw "fill black text
+$TEXT_H_POSITION+$TEXT_V_POSITION '`eval sed '$COUNTER!d'
$WORKFOLDER/$TRACKLIST`'" $WORKFOLDER/$OUTPUT

    COUNTER=$((COUNTER+1))
    BUTTON_H_POSITION=$((BUTTON_H_POSITION+$BUTTON_H_OFFSET))
    BUTTON_V_POSITION=$((BUTTON_V_POSITION+$BUTTON_V_OFFSET))
    OID_H_POSITION=$((OID_H_POSITION+$OID_H_OFFSET))
    OID_V_POSITION=$((OID_V_POSITION+$OID_V_OFFSET))
    TEXT_H_POSITION=$((TEXT_H_POSITION+$TEXT_H_OFFSET))
    TEXT_V_POSITION=$((TEXT_V_POSITION+$TEXT_V_OFFSET))
  done
  debug " done" 1
}

## Main

while [ $# -ne 0 ]; do
  case $1 in
    -c)
      shift
      if [ ! "$1" ]; then
        echo "No comment given -> EXIT!"
        exit 1
      else
        case "$1" in
           *\ * )
```

```
                COMMENT=`eval echo "$1" | sed -e 's/ /_/g'`
                YAML_FILE=${COMMENT}.yaml
                OUTPUT="${COMMENT}.png"
                ;;
              *)
                COMMENT="$1"
                YAML_FILE="$1.yaml"
                OUTPUT="$1.png"
                ;;
        esac
      fi
      shift
      ;;
    -d)
      shift
      if [ "$1" -eq "$1" ] 2>/dev/null ; then
        DEBUG_LEVEL=$1
      else
        echo "No debug level given! -> EXIT!"
        exit 1
      fi
      shift
      ;;
    -h)
      echo "usage: $0 [ARGUMENTS] "
      echo "-c [TEXT]: Comment which is used in gme-file"
      echo "-d [n]: Debug output with level n (0...3)"
      echo "-h - this help ;-)"
      echo "-pid [nnn] - Product ID to use e.g. 701 "
      echo "-w [path to working folder] - Working folder. All
operations will be performed in this folder. The source folder for wave
files will be expected here."
      echo ""
      echo ""
      echo ""
      echo ""
      exit 0
      ;;
    -pid)
      shift
      if [ -n "$1" ]; then
        if [ "$1" -eq "$1" ] 2>/dev/null ; then
          PRODUCT_ID="$1"
        else
          echo "Error - no product ID given -> EXIT!"
          exit 1
        fi
      else
        echo "Error - no product ID given -> EXIT!"
        exit 1
      fi
```

```
      shift
      ;;
     -w)
      shift
      if [ -e "$1" ] || [ ! "$1" ] ; then
        WORKFOLDER=$1
        shift
      else
        echo "Given work folder does not exist or is not accessible ->
EXIT!"
      exit 1
      fi
      ;;
     *)
      echo "Parameter \"$1\" unkown -> exit!"
      shift
      exit 1
      ;;
  esac
done



if [ $DEBUG_LEVEL -le 1 ]; then
  echo -n "Generating $COMMENT.gme ... "
else
  echo "Generating $COMMENT.gme ... "
fi

if [ -z $PRODUCT_ID ]; then
  echo "Error - no product ID given -> EXIT!"
  exit 1
else
  debug "Product-ID: $PRODUCT_ID" 1
fi

if [ -z $COMMENT ]; then
  echo "Error - no comment given -> EXIT!"
  exit 1
else
  debug "Comment: $COMMENT" 1
  debug "Output: $OUTPUT" 1
  debug "YAML: $YAML_FILE" 1
fi

check_tools tttool
check_tools oggenc
check_tools composite
check_tools convert
check_tools rec
check_tools notify-send
```

```
check_tools zenity

cleanup cover
cleanup tracklist
cleanup output
cleanup ogg
cleanup oids

mk_ogg_files
mk_yaml_file
mk_oids
mk_gme
mk_base_image
mk_trackbuttons

desktop_message "GME file generated"

cleanup cover
cleanup tracklist
cleanup templates
cleanup ogg
cleanup oids

echo "done!"
echo ""
echo "GME file generated - please copy ${COMMENT}.gme to your TipToi device and print out $OUTPUT by using 600 dpi printing resolution!"
echo ""
```

Nicht vergessen `gen_gme.sh` mit `chmod +x gen_gme.sh` auch ausführbar zu machen