```
//=============================================================================
//  MuseScore Plugin
//
//  This plugin reads a chromatic TAB staff and translates it into the
//  diatonic+ TAB for the Mountain Dulcimer.
//
//  **** This is still a work in process, so expect surprises ****
//
//  **** Please read the README file before using ****
//
//  This plugin created by Jeffrey Rocchio and is free for anyone to use or
//  modify.
//
//   NOTE: Sections of code warranting verbose documentation are noted with
//   references, in the form "See CD-##," which are rows in a table in
//   a LibreOffice document which can be found in my github repository as
//   Chromo-to-Diatonic-Dulcimer-TAB_CodeDocumentation.odt.
//
//  This plugin created by Jeffrey Rocchio and is free for anyone to use or
//  modify.
//
//   NOTE: This file was edited with the KDE Kate editor, indentation is with
//   tabs, 4 spaces equlivant per tab.
//=============================================================================

//-----------------------------------------------------------------------------
//  1.0: 05/09/2021 | Initial Release
//-----------------------------------------------------------------------------


import QtQuick 2.0
import QtQuick.Dialogs 1.1
import MuseScore 3.0

MuseScore {
      title: "Mtn Dulcimer.Chromo TAB 2 Diatonic"
      version:  "1.0"
      description: "Modifies existing Mountain Dulcimer chromatic TAB to diatonic fret numbering"
      menuPath: "Plugins.Mtn Dulcimer.Chromo TAB 2 Diatonic"


    QtObject { // oUserMessage
        id: oUserMessage

        //  PURPOSE: Is an error recording and reporting object, used
        //to provide error and warning messages to the user.

        property bool bError: false
        property int iMessageNumber: 0 // <-- Last trapped error
        readonly property var sUserMessage: [
        //  0: <-- All OK, No Error
        "OK",
        //  1:
        "Nothing Selected - please select 1 or more measures to transform and try again.",
        //  2:
        "Selected staff does not appear to be a TAB staff. Please double-check and try again.",
        //  3:
        "Selected staff appears to be a linked staff. Transforming will destroy the linked-to
        staff.
        Please create a non-linked TAB staff, copy the linked TAB staff's content to that,
```

```
select the measure of that you wish to transform and try again.",
//  4:
"Unrecognized element on SMN staff. Don't know how to handle it.",
//  5:
"Selection appears to be invalid. Please clear your selection and try again. Be sure to
select whole measures, do not start with part of a measure. Also, please makre sure your
selection does not span multiple staves.",
//  6:
"Cannot find a valid TAB staff immediately below your selected range.",
//  7:
"An error occurred trying to create tied note.",
//  8:
"ERROR: Infinite Loop in makeTAB()",
//  9:
"**** ERROR: fret number too low in transformNote() - Likely due to not having selected a
TAB staff, please double check and try again. If error persists, please report.",
//  10:
"Transformation complete. \n\n NOTE: half-frets are added as fingering text elements. For
them to be visible the TAB staff must be set to 'show fingering' in [Staff Properties |
Advanced].",
]

function getError() { return bError; }

function setError(iMessageNum) {
    oUserMessage.bError = true;
    oUserMessage.iMessageNumber = iMessageNum;
} // end function setError()

function clearError() {
    bError = false;
    return;
}

function showError(bReset) {
    console.log("", oUserMessage.sUserMessage[oUserMessage.iMessageNumber]);
    if (bReset) {
        oUserMessage.iMessageNumber = 0;
        oUserMessage.bError = false;
    }
} // end function getError()

function popupError() {

    errorDialog.openErrorDialog(qsTranslate("QMessageBox", sUserMessage[iMessageNumber]));
    oUserMessage.iMessageNumber = 0;
    oUserMessage.bError = false;

} // end function popupError()

function popupStatusMessage(iStatusMessage) {
    //  Confession: This object was originally written to provide
    //an error-reporting system. But I later realized that I also
    //needed a way to just inform the user of status at the end
    //of the plugin's run. So I 'bolted on' this "status" function,
    //and it's associated MessageDialog object. So not exactly
    //a clean object here.

    statusDialog.openStatusDialog(qsTranslate("QMessageBox",
    sUserMessage[iStatusMessage]));
```

```
    } // end function popupError()

} // end oUserMessage QtObject

QtObject { // oDebug
    id: oDebug

    //PURPOSE:
    //  Provide services that help in debugging. This is primarily
    //services to create console.log print statements.

    function fnEntry(fnName) {
        console.log("");
        console.log("======== | In function ", fnName, "() |
        ================================\n");
    }

    function fnExit(fnName) {
        console.log("");
        console.log("======== | ", fnName, "() RETURNing to caller ||\n");
    }

    function showObject(oObject) {
        //  PURPOSE: Lists all key -> value pairs of the passed-in
        //object to the console.

        if (Object.keys(oObject).length >0) {
            console.log("");
            console.log("---- Key->Value pairs of object || ", oObject.name, "||");
            Object.keys(oObject)
            .filter(function(key) {
                return oObject[key] != null;
            })
            .sort()
            .forEach(function eachKey(key) {
                console.log("---- ---- ", key, " : <", oObject[key], ">");
            });
        }
        console.log("---- ----");
        console.log("");
    }

} // end QtObject oDebug

QtObject { // oTabStaff
    id: oTabStaff

    //PURPOSE:
    //  Provides methods and state for the TAB staff we are
    //transforming.

    //   Create the offset information array. This array maps chromatic fret
    //numbers to the Mountain Dulcimer diatonic+ fret numbers and
    //half-frets.
    // ============================================ See CD-01 >
    readonly property var aDiatonicFretMap: [
        { offset: 0, halfFret: false },
        { offset: 1, halfFret: true },
        { offset: 1, halfFret: false },
```

```
    { offset: 2, halfFret: true },
    { offset: 2, halfFret: false },

    { offset: 2, halfFret: false },
    { offset: 3, halfFret: true },
    { offset: 3, halfFret: false },
    { offset: 4, halfFret: true },
    { offset: 4, halfFret: false },

    { offset: 4, halfFret: false },
    { offset: 5, halfFret: true },
    { offset: 5, halfFret: false },
    { offset: 6, halfFret: true },
    { offset: 6, halfFret: false },

    { offset: 7, halfFret: true },
    { offset: 7, halfFret: false },
    { offset: 7, halfFret: false },
    { offset: 8, halfFret: true },
    { offset: 8, halfFret: false },

    { offset: 9, halfFret: true },
    { offset: 9, halfFret: false },
    { offset: 9, halfFret: false },
    { offset: 10, halfFret: true },
    { offset: 10, halfFret: false },

    { offset: 11, halfFret: true },
    { offset: 11, halfFret: false },
    { offset: 12, halfFret: true },
    { offset: 12, halfFret: false },
    { offset: 12, halfFret: false }
];

// ========================================== See CD-02 >
readonly property var sHalfFretSymbol: "+"
readonly property var sHalfFretSize: 10 // See notes in CD-02
readonly property var sHalfFretWeight: 1; // Bold text
readonly property var nHalfFretXPosition: 1.4; // position a bit to the right of the
fret#.
readonly property var ihalfFretColor: "#000000" // Black
// readonly property var ihalfFretColor: "#aa0000" // RED, used only in debugging


function transformFret(oNote, oCursor) {
    //    PURPOSE: Takes a chromatic TAB note object and translates it to
    //mountain dulcimer diatonic+ fret number.
    //    RETURNS: The note is transformed directly in the score by
    //of simply updating the note object directly here. When this is
    //done Musescore immediately upates the TAB in the score with the
    //revised fret number.
    // ========================================== See CD-01 >

    var bDEBUG = true;
    bDEBUG = false;

    if(bDEBUG) oDebug.fnEntry(transformFret.name);

    var iChromoFret = oNote.fret;
    if (iChromoFret >= 0) {
```

```
            if(bDEBUG) console.log(" ");
            if(bDEBUG) console.log("---- Input Note: pitch <", oNote.pitch, "> string <",
            oNote.String, "> chromoFret <", iChromoFret, "> | tpc <", oNote.tpc, "> tpc1 <",
            oNote.tpc1, "> tpc2 <", oNote.tpc2, ">");
            oNote.string = oNote.string; // In reality this should never change.
            oNote.fret = iChromoFret - aDiatonicFretMap[iChromoFret].offset;
            oNote.pitch = oNote.pitch - aDiatonicFretMap[iChromoFret].offset;
            oNote.tpc1 = oTPC.wrapTpcWithOffset(oNote.tpc1,
            aDiatonicFretMap[iChromoFret].offset);
            oNote.tpc2 = oTPC.wrapTpcWithOffset(oNote.tpc2,
            aDiatonicFretMap[iChromoFret].offset);
            if(bDEBUG) console.log("\n---- New Values: pitch <", oNote.pitch, "> string <",
            oNote.string, "> diaFret <", oNote.fret, "> | tpc <", oNote.tpc, "> tpc1 <",
            oNote.tpc1, "> tpc2 <", oNote.tpc2, ">");
            oNote.play = false; // Turn note 'Play' off (since it's pitch is now really a
            fake-out).
            if (aDiatonicFretMap[iChromoFret].halfFret) oTabStaff.addHalfFret(oNote);
        }
        else {
            console.log("**** ERROR: fret number too low ****");
            oUserMessage.setError(9);
            return false;
        }

        if(bDEBUG) oDebug.fnExit(transformFret.name);
        return true;

    } // end transformFret()

    function addHalfFret(oNote) {
        //PURPOSE: Places a half-fret symbol next to the note.

        var bDEBUG = true;
        bDEBUG = false;

        if(bDEBUG) oDebug.fnEntry(addHalfFret.name);

        var half = newElement(Element.FINGERING);
        half.text = sHalfFretSymbol;
        half.fontStyle = sHalfFretWeight;
        half.color = ihalfFretColor;
        oNote.add(half);
        //Here's an odd thing - you have to set the x,y offset
        //properties **after** you have added the fingering object
        //to the note. If you set them prior to adding, like I did
        //for the color above, they don't seem to have any effect.
        //I personally cannot explain this difference in behavior.
        half.autoplace = false; // Must set this for alignment and offsets to be effective.
        half.align = "VMASK"; // 'VMASK' gives me left-aligned, vertically centered, which is
        what I want.
        half.offsetX = nHalfFretXPosition;
        if(bDEBUG) oDebug.showObject(half);

    if(bDEBUG) oDebug.fnExit(addHalfFret.name);

    } // end addHalfFret()

} // end QtObject oTabStaff

QtObject { // oTPC
```

```
id: oTPC

//PURPOSE:
//  Provide an abstraction of the Tonal Pitch Class properties.
//See CD-03 for why this is needed.

// ========================================== See CD-03 >
readonly property var tpcPerPitch: [
    [ 26, 14,  2 ], //B# , C , Dbb
    [ 33, 21,  9 ], //B##, C# , Db
    [ 28, 16,  4 ], //C##, D , Ebb
    [ 23, 11, -1 ], //D# , Eb, Fbb
    [ 30, 18,  6 ], //D##, E , Fb
    [ 25, 13,  1 ], //E# , F , Gbb
    [ 32, 20,  8 ], //E##, F#, Gb
    [ 27, 15,  3 ], //F##, G , Abb
    [ 22, NaN, 10 ], //G#,   , Ab
    [ 29, 17,  5 ], //G##, A , Bbb
    [ 24, 12,  0 ], //A# , Bb, Cbb
    [ 31, 19,  7 ]  //A##, B , Cb
];

function findTPCidx(tpc) {
    // ========================================== See CD-03 >

    var bDEBUG = true;
    bDEBUG = false;

    if(bDEBUG) oDebug.fnEntry(findTPCidx.name);

    if (bDEBUG) console.log(" ");
    if (bDEBUG) console.log("---- Entering 1st for loop, tpc passed in is <", tpc, ">
    ----: ");
    for (var tpcIdx = tpcPerPitch.length; tpcIdx-- > 0; ) {
        for (var tpcI = tpcPerPitch[tpcIdx].length; tpcI-- > 0; ) {
            if (tpcPerPitch[tpcIdx][tpcI] == tpc) {
                return { idx: tpcIdx, column: tpcI };
            }
        }
    }
    if (bDEBUG) console.log(" ");

    if(bDEBUG) oDebug.fnExit(findTPCidx.name);

} //end findTPCidx()

function wrapTpcWithOffset(tpc, offset) {
    // ========================================== See CD-03 >

    var bDEBUG = true;
    bDEBUG = false;
    if(bDEBUG) oDebug.fnEntry(wrapTpcWithOffset.name);

    var tpcIdx = findTPCidx(tpc);
    tpcIdx.idx -= offset;
    if (tpcIdx.idx >= tpcPerPitch.length) {
        tpcIdx.idx -= tpcPerPitch.length;
    }
    else if (tpcIdx.idx < 0) {
        tpcIdx.idx += tpcPerPitch.length;
```

```
        }
        if (tpcPerPitch[tpcIdx.idx][tpcIdx.column] == NaN) {
            tpcIdx.column += 1;
        }

        if(bDEBUG) oDebug.fnExit(wrapTpcWithOffset.name);
        return tpcPerPitch[tpcIdx.idx][tpcIdx.column];

    } // end wrapTpcWithOffset()



} // end QtObject oTPC

function assessValidity(oCursor) {
    //   PURPOSE: Prior to attempting any transformation, see
    //if it appears that the user has actually specified a
    //valid staff to perform the transform on.
    //We want to check for:
    //   1. Is there a selection - one or more chords/measures?
    //   2. Is it even a TAB staff?
    //   3. Is the staff 'Linked' to another staff (if so, we
    //don't want to operate on it as it will totally destroy
    //the arrangement on whatever staff it is Linked to.)
    //Unfortunally it is not possible to determine this in
    //plugin code, so I am NOT actually testing for this
    //condition.
    //   RETURNS: true if all is well, otherwise returns false
    //and sets an integer that is used as an index into an error
    //message string array.

    var bDEBUG = true;
    bDEBUG = false;

    if(bDEBUG) oDebug.fnEntry(assessValidity.name);

                // We'll start with a presumption that user selection is valid.
    var bValid = true;

    if ((curScore.selection.elements.length==0) || (!curScore.selection.isRange)) {
        oUserMessage.setError(1); //<- No selection.
        bValid = false;
    }
                //There is a selection, check if it spans more than one stave
    if(bValid) {
        if ((curScore.selection.endStaff-curScore.selection.startStaff) > 1) {
            oUserMessage.setError(5); //<- More than one stave in selection range.
            bValid = false;
        }
    }

                //Selection looks valid, now check if this is a TAB staff.
                // ========================================== See CD-04 >
    if(bValid) {
                //Rewind puts cursor on staff containing the selection range.
        oCursor.rewind(Cursor.SELECTION_START);
        if(!oCursor.element.staff.part.hasTabStaff) {
            oUserMessage.setError(2);
            bValid = false;
        }
```

```
    }

    if(bDEBUG) oDebug.fnExit(assessValidity.name);

    return bValid;
} // end assessValidity()

function transformChord(oChord, oCursor) {
    //   PURPOSE: Given a chord object, transform all of it's notes
    //from chromatic fret numbers to Mtn Dulcimer diatonic+ fret
    //numbers. This includes adding the text elements for a '+'
    //character if the diatonic+ fret is a half-fret.
    //This is where the real work occurs.
    //   NOTE: This function does not advance the cursor.


    var bDEBUG = true;
    bDEBUG = false;

    if(bDEBUG) oDebug.fnEntry(transformChord.name);

    var oNotesArray;
    var iNumOfNotes;

    oNotesArray = oChord.notes;
    iNumOfNotes = oNotesArray.length;
    for (var i=0; i<iNumOfNotes; i++) {
        if(!oTabStaff.transformFret(oNotesArray[i], oCursor)) return false;
    }

    if(bDEBUG) oDebug.fnExit(transformChord.name);

    return true;

} // end transformChord

function transformTAB(oCursor) {
    //    PURPOSE: Walk through the TAB staff and transform all notes from
    //chromatic to Mountain Dulcimer diatonic+ fret numbers (with those
    //half-fret symbols as well, as needed.)
    //   ASSUMES: We have successfully passed the assessValidity() tests.

    var bDEBUG = true;
    bDEBUG = false;

    if(bDEBUG) oDebug.fnEntry(transformTAB.name);

    var bNoErrors = true;

    var endTick = 0;
    var oChord;
    var oNotesArray;
    var iNumOfNotes;

                //   Set up markers to sense the end of our work.
    oCursor.rewind(Cursor.SELECTION_END);
    if (oCursor.tick === 0) {
                //   (This happens when the selection includes
                //the last measure of the score. rewind(2) goes
                //behind the last segment (where there's none)
```

```
                //and sets tick=0)
            endTick = curScore.lastSegment.tick + 1;
        } else {
            endTick = oCursor.tick;
        }

                //   OK, now iterate through the score until
                //we reach the end of the selection or the score.
        oCursor.rewind(Cursor.SELECTION_START);

        curScore.startCmd(); // <--** We set this to mark an 'undo' point for user.
        var iWatchDog = 5000;
        while (oCursor.segment && oCursor.tick < endTick) {
            if (oCursor.element.type === Element.CHORD) {
                oChord = oCursor.element;
                if(!transformChord(oChord, oCursor)) { // Note that this funct does not advance
                cursor.
                    bNoErrors=false;
                    break;
                }
            }
            oCursor.next();
            iWatchDog--;
            if (iWatchDog<=0) {
                console.log("**** ERROR: WatchDog Exceeded, Infinte Loop ****");
                bNoErrors = false;
                break;
            }
        } // Bottom of while loop
        curScore.endCmd(); // <--** Marks the end of the 'undo' point.

        if(bDEBUG) oDebug.fnExit(transformTAB.name);

        return bNoErrors;

    } // end of transformTAB()


    onRun: {
        console.log("********** RUNNING **********");

        var oCursor = curScore.newCursor()

        assessValidity (oCursor);
        if (oUserMessage.getError()) {
            oUserMessage.showError();
            oUserMessage.popupError();
        }
        else { // We have a valid TAB staff & selection to work on
            transformTAB(oCursor);
            if (oUserMessage.getError()) {
                oUserMessage.showError();
                oUserMessage.popupError(); // Houston, we have a problem....
            }
            else {
                oUserMessage.popupStatusMessage(10); // All done, and warn user re fingering text.
            }
        }

        console.log("********** QUITTING **********");
```

```
        Qt.quit();

    } //END OnRun



    //==== PLUGIN USER INTERFACE OBJECTS ============================================

    MessageDialog {
        id: errorDialog
        visible: false
        title: qsTr("Error")
        text: "Error"
        onAccepted: {
            Qt.quit()
        }
        function openErrorDialog(message) {
            text = message
            open()
        }
    }

    MessageDialog {
        id: statusDialog
        visible: false
        title: qsTr("Plugin Status")
        text: "Plugin Status"
        onAccepted: {
            Qt.quit()
        }
        function openStatusDialog(message) {
            text = message
            open()
        }
    }


} // END Musescore
```