EMDA 簡略化の為のツール作成

福知山公立大学 情報学部情報学科

32245105　和田想

指導教員　橋田光代　准教授

提出日　2026 年 1 月 30 日

改訂日　2026 年 1 月 30 日

— 目次 ———

— 図目次 ———

— 表目次 ———

1. はじめに

ツールとは，目的を達成するための道具として用いられるコンピュータプログラムのことである．今回達成したい目的というのは EMDA 分析の簡略化，EMDA 分析とは対象である音楽や，物語の根幹となる部分を軸に木構造を作成する分析方法であり，その木構造の簡易作成が目的である．この研究を始めた経緯として，私が 2 年生の時に EMDA 分析を行った際，PowerPoint で木構造を作成したのだが，膨大な時間がかかってしまい分析の進行が遅くなってしまった事から，この作業を簡単にすれば EMDA 分析自体にも触れやすくなるのではないかと考えたため，今回のツール作成に至った．開発環境としては，Python に標準搭載されている GUI ライブラリである Tkinter を使用した．

本稿では，EMDA の前身となった GTTM，そして EMDA の説明を踏まえた上で，プログラムの制作過程と最終的な機能，性能評価について述べていく．

2. EMDA について

2.1 音楽理論 GTTM とは

音楽理論とは，音イベントを時系列で構造解析する技術の事である．この技術を用いることで，時間の進行によって生じる音イベントをグループ分けした上で，重要な音を見つけることが出来る．Generative Theory of Tonal Music(GTTM) は，作曲家・音楽学者の Lerdahl と，言語学者の Jackendaff によって 1983 年に提案された音楽理論である．

特徴としては楽曲の簡約や木構造による表現が挙げられ，構造解析は，音楽の認知に必要とされる 4 つのサブ理論から成る．1 つ目はグルーピング構造であり，楽曲を楽曲中の動機や楽節といった単位のグループに分節し，分節された各グループの階層構造を定めるもの，2 つ目は拍節構造であり，各拍節レベルにおける強拍と弱拍を定めるもの．3 つ目がタイムスパン簡約であり，グルーピング構造の解析と拍節構造の解析をもとに重要な音の選出を繰り返し，ボトムアップに木構造 (タイムスパン木) を作っていくものである．[1](図 1)

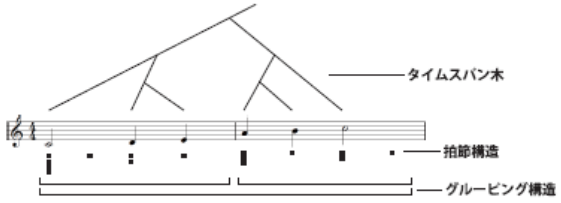


図 1: GTTM のタイムスパン木

タイムスパン木の構造では，枝 (branch) が幹 (stem) の従属部になっており，幹が枝より重要な音であることを表している．4 つ目に延長的簡約という，機能和声に基づく緊張-弛緩構造をトップダウンに分析するものがある．

2.2 EMDA

今回ツールを作成する目的である EMDA(Emotion Movement Design Annotator)[2] は，GTTM を音楽だけでなく様々な作品に応用したものである．対象である音楽や物語の根幹となる部分を軸に木構造を作成し，それまでのメロディや事象が何に繋がっているのか，何が重要なのかを可視化したものである．GTTM と対比しながら考えると，1 つのイベントが音符 1 つ分であり，それらをグループ分けした上で，それぞれの重要度に応じて幹と枝の従属関係を構築することにより，木構造を作成していくものになっており，この時のグループ分けの基準やどのイベントがより重要なのかは，対象の作品や分析する人物によって異なる．この分析をする中で必要ではあるが，時間がかかってしまう木構造の作成という部分に焦点を当て，ツールを作成した．

3. 制作過程

3.1 「木」部分の作成

今回，最終的にツールで作成したいものは以下のようなものである．(図 2)

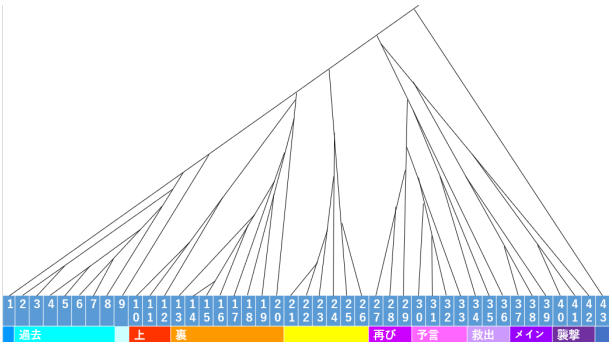


図 2: 以前作成した木構造

作成するにあたって，上部の木構造部分と下部のイベント部分で分けて考えることにし，順序としては木構造部分を作成した上で，テキストボックスをそれに連動させる形で追加するという手順で行った．

木構造を作成する上で，必要な機能として自由に線を動かす機能が真っ先に思い浮かび，その部分から手を付けることにした．最初に，Tkinter のキャンバスに描いた左右 2 本の線分の「上端」だけをドラッグで動かせるようにし，線同士が交差または十分近づいたら，ドラッグした側の上端を相手線の中点へスナップして結合状態にしつつ，子側を中点から一定距離以上離して離すと結合解除できるようにした．(図 3)

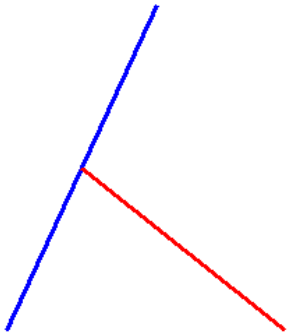


図 3: 2 本線の木構造

木構造自体はこの動作を繰り返していけば形になると考えた．次に，線の数を 2 本から 6 本に増やして動作させたが，この時の線はそれぞれの中点にしかスナップしないような仕組みになっていたため，1 つの幹に対して枝が 2 本吸着する形になった時に同位置に吸着するという問題が発生した．(図 4)

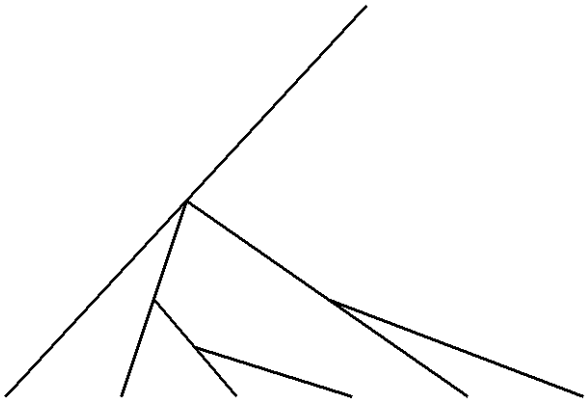


図 4: 中点に吸着する 6 本線の木構造

その問題を解決するべく，線を吸着させる部分を線の中点ではなく，t 値 (全ての線の数-1) で等間隔にスロットを作り，既に使用しているスロットには他の線が吸着しないようにした．(図 5)

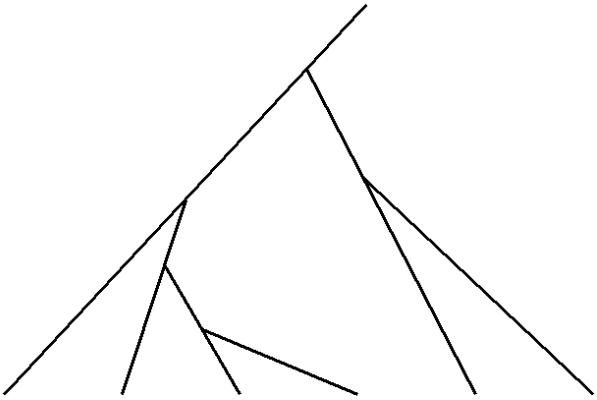


図 5: 各スロットに吸着する 6 本線の木構造

これにより，木構造を作成するための基礎的な機能は実装出来たと言えるが，とても便利といえるものではない為，さらに機能を追加した．

まず，イベントの数は作品によって変動する為，自由に線の本数を変えられる機能を追加した．テキストボックス内に本数を入力することで瞬時にその本数になる機能と，ボタンを押すことにより，右端 (一番新しい線) から線を追加，削除することが出来る機能の 2 種類である．また，PowerPoint で作業していた時に煩わしかった作業の 1 つである，イベント間にイベントを追加，削除するという機能を追加し，最終的には以下の図のようになった．(図 6)

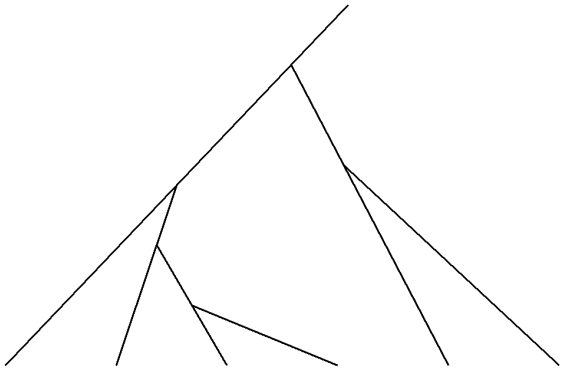


図 6: 木構造部分完成時のウィンドウ

各種ボタンを押すことによって，それぞれの機能が使えるようになっている．これにより，木構造部分だけなら本数が自由に変えられるツールが出来たと言える．

3.2 「イベント」部分の作成

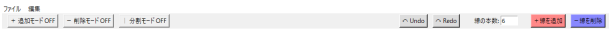
次に，作成した木構造部分に連動させながらテキストボックスを追加する段階に入った．まずは線分同士の間にテキストボックスを入れ込むような形で作成した．(図 7)

図 7: 上段テキストボックス追加

テキストボックスとその左上の角から伸びている線分が連動している仕組みになっており，例えば左から 2 番目の線を削除した時，共に左から 2 番目のテキストボックスが消えるようになっている．

最初に例として挙げた木構造を再現するのなら，それぞ

れのイベントをグループ分けするためのテキストボックスの追加も必要になってくる．グループ分けという目的に沿ったものにする為，初期状態は全てのイベントを包括する大きなグループがあり，それを分割させていくことで，イベントのグループ分けを再現しようと考えた．この考えのもと作成したのが以下の画面になる．(図 8)(図 9)



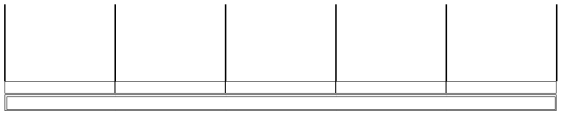


図 8: 下段グループ列分割前

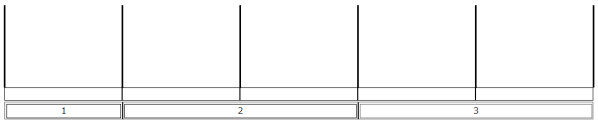


図 9: 下段グループ列分割後

このように，イベントテキストボックスの下部に 1 つの大きなテキストボックスを配置し，それを分割することでグループ分けとした．通し番号については分かりやすいように手動で入力したものである．他にもいくつか細かい修正や機能を追加した時の初期ウィンドウが以下の画像になる．(図 10)

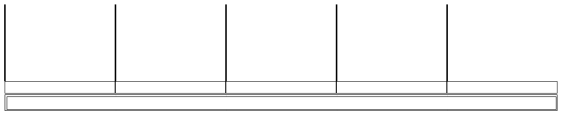


図 10: 下段グループ列追加後ウィンドウ

3.3 ユーザーフィードバック

ここまで作成してきたツールを実際に使ってもらい，不便な点や改善点を挙げてもらうテストを行った．対象ユーザーは同学年のゼミ生 5 人と顧問 1 人の計 6 人，調査方法はまず以下の画像 (図 11) と同じ木構造をツールを使用して作成してもらい，

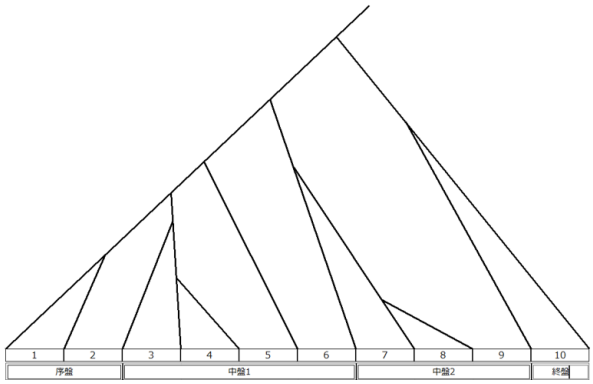


図 11: 調査用木構造 (編集前)

この木構造を作成した後にイベント「6」とイベント「9」を削除，イベント「7」とイベント「8」の間にイベント「!」を追加，グループの分割位置を変える，といった追加の操作指示をすることで，木構造を作成する時に使うであろう機能をすべて使ってもらうという形で調査を行った．最終的に作成した木構造は以下のものである．(図 12)

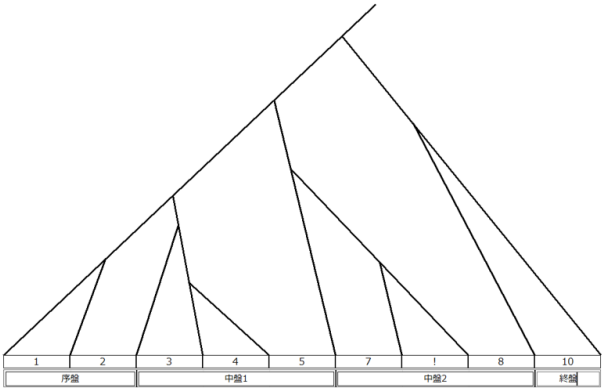


図 12: 調査用木構造 (編集後)

この木構造を作成する上での感想や改善点を挙げてもらい，それを元にプログラムに修正を加えていった．フィードバックとしては，直感的に操作が分かりづらい箇所がある，判定の拾い方次第では不具合のような挙動になることがある，などの自分で操作していた時には気づけなかった意見があり，テストプレイの重要性を実感した．

4. コード解説

4.1 概要（このツールでできること）

本プログラムは，Tkinter を用いて GUI を構築し，キャンバス上に複数の線（Line）を配置・編集するツールである．ユーザは線の上端をドラッグして形状を変えたり，線同士を吸着（スナップ）させて親子関係（階層構造）を作ることができる．

さらに，本ツールは「線と線の間（区間）」ごとにテキスト入力欄を自動生成し，区間に対応する説明文などを入

力できる．加えて下段には複数区間をまとめる「グループ行」を設け，境界をクリックすることで区間の分割・結合（グルーピング）を行える．

操作の履歴は Undo/Redo に対応し，状態は JSON として保存・読み込みが可能である．また，資料作成向けにキャンバス領域のみを画像として保存するスクリーンショット機能も実装している．

4.2 全体構成（クラス設計）

実装は大きく Line と SnapHierarchyApp の 2 クラスに分かれる．

- **Line クラス**：線 1 本の「幾何（座標）」「描画」「親子スナップ（追従）」を担当する．
- **SnapHierarchyApp クラス**：UI 全体，イベント処理（クリック・ドラッグ・ホイール等），線の追加削除，テキスト欄の再構築，Undo/Redo，保存/復元を担当する．

以降では，設計上の重要点（右端の隠し線・座標系）→ Line → SnapHierarchyApp の順に説明する．

4.3 座標系とビューポート（ズーム・パンの前提）

本ツールでは，線の X 位置を「ピクセル」ではなく **0〜1 の比率（世界座標）** として保持する．この設計により，ウィンドウサイズ変更やズーム操作後でも，再描画によって配置を保ちやすい．

世界座標と画面座標（px）の変換は Line.map_x()/unmap_x() および Line.map_y()/unmap_y() で行う．表示範囲（viewport）は X 方向が view_min, view_max，Y 方向が view_y_min, view_y_max で表される．

- ズーム：ホイール操作で，マウス位置を基準（アンカー）に X/Y 同倍率で拡大縮小する（zoom_at_worldxy()）．
- パン：中ボタン/右ボタンのドラッグで，表示範囲を平行移動する（pan_by_xy()）．
- 縦パン：Shift+ホイールで Y 方向のみパンする．

極端な拡大で表示幅が 0 に近づくことを防ぐため，最小幅 _min_span（Y は _min_span_y）を設けている．

4.4 右端の隠し線（区間を作るための設計）

本コードの重要な設計ポイントとして，内部的に「右端の隠し線」を常に 1 本保持する点が挙げられる．線の配列 self.lines には **内部的な線の本数** が入り，実際に描画するのは self.lines[:-1]（最後の 1 本を除いたもの）のみである．

この設計を採用する理由は，テキスト入力欄を「線と線の間（区間）」に生成しているためである．区間を N 個作るには境界となる線が $N+1$ 本必要になるが，画面上で見せたい「線の本数」は N 本にしたい．そこで，**見えない右端の線**を 1 本追加で持ち，次を成立させている．

- 画面に見える線の本数：|self.lines| − 1
- 上段のテキスト欄の数：|self.lines| − 1（＝見える線と同数）

つまり「見える線 1 本につき，その右側区間に対応する入力欄が 1 つある」という対応関係を作りやすくなる．右端の隠し線は**描画しない**が，区間幅の計算や下段グループ行の右端境界として使われる．

4.5 Line クラス（線 1 本のモデルとスナップ）

4.5.1 Line が保持する情報

Line は線 1 本分の状態を保持するクラスである．主なコードを役割ごとに示す．

- **識別子**：line_id（保存/復元やテキスト対応付けに使う永続 ID）
- **世界座標（比率）**：base_ratio（根元 X），base_y_ratio（根元 Y），top_ratio_x, top_ratio_y（上端の世界座標）
- **画面座標（px）**：base_x，base_y，top_x，top_y
- **親子関係（スナップ）**：parent, children, snap_t, used_t_values
- **描画 ID**：id（Tkinter Canvas 上のオブジェクト ID）

根元は base_y_ratio=0.9 に固定され，キャンバス下側（90%付近）に揃えて配置される．上端はドラッグやスナップによって変化し，形状（傾きや長さ）を表現する．

4.5.2 世界座標と画面座標の変換

map_x() は世界座標の X（0〜1）を，view_min/view_max の範囲に応じて画面 X に写像する．逆に unmap_x() は画面 X から世界座標 X へ戻す．Y 方向も同様に map_y()/unmap_y() が view_y_min/view_y_max を用いて写像を行う．

この変換を統一しておくことで，ズーム・パン後でも「世界座標としての配置」を保った再描画ができる．

4.5.3 描画と幾何更新（update_base_position / draw）

update_base_position() はキャンバスサイズに基づき，根元（base）の画面座標を更新する．親を持たない線（parent is None）の場合，上端については以下の方針で決める．

- 既に上端比率（top_ratio_x/y）を持っている場合：比率から上端を復元する
- それ以外：初期状態として「真上に length だけ伸びる線」にする

描画そのものは draw() で行い，既存の線 ID があれば削除してから再生成する．

4.5.4 ドラッグによる形状変更（move）

ドラッグ中は move(dx,dy) が呼ばれ，上端座標を (dx, dy) だけ移動する．その後，

(1) update_top_ratios() により，上端位置を世界座標比率に更新する
(2) draw() により線を描き直す

(3) 子線があれば follow_parent() で追従させる

という流れで，見た目と内部状態（比率）を同期させている．

4.5.5 親子スナップと追従（follow_parent / detach_from_parent）

線同士を吸着させると，子線は親線の線分上の位置 t に固定される．この t が snap_t であり，follow_parent() は

$$\text{top} = \text{base(parent)} + t \cdot (\text{top(parent)} - \text{base(parent)})$$

により子線の上端位置を更新する．これにより親線の形状が変わっても，子線は同じ割合位置に追従できる．

また，同じ親線に複数の子線が吸着するとき，同一点に重ならないよう親側で使用済みの t を used_t_values に記録する．吸着候補の t は generate_snap_t_values(count) により等間隔で生成される．

親からの解除は detach_from_parent() で行う．このとき，親の children から自分を外し，used_t_values から自分の t も解放する．

4.6 SnapHierarchyApp クラス（UI 全体とイベント処理）

4.6.1 UI の構成

SnapHierarchyApp はツール全体の状態と UI を管理する．主な UI は次の通りである．

- 上部コントロール：追加/削除/分割モード切替，表示範囲入力（%），線の本数指定，Undo/Redo など
- キャンバス：線の描画とドラッグ操作の中心領域（スクリーンショットはここだけを保存）
- 上段テキスト行：各区間（線と線の間）に 1 つずつ Entry を配置
- 下段グループ行：複数区間をまとめる Entry を配置し，境界クリックで分割/結合

4.6.2 イベントバインド（入力と挙動の対応）

主なバインドは以下である．

- 左クリック：ドラッグ開始判定（on_press），ドラッグ（on_drag），リリース（on_release）
- 中/右ドラッグ：パン開始・移動・終了（on_pan_start/drag/end）
- ホイール：ズーム（on_wheel），Shift+ホイールは縦パン
- <Configure>：リサイズ時に再描画（on_resize）
- Ctrl+Z/Ctrl+Y：Undo/Redo

なお左右キーは「入力欄フォーカス移動」にも用いるため，Entry 上では _focus_prev/_focus_next を優先し，キャンバス操作（パン）と干渉しないようにしている．

4.6.3 再描画の中心処理（redraw_all）

キャンバスの表示更新は redraw_all() が中心となる．処理は概ね次の順である．

(1) 全線（隠し線含む）について根元座標を更新（update_base_position）
(2) 見える線（self.lines[:-1]）のみ描画（draw）
(3) 親子関係がある線は follow_parent() により追従位置を反映
(4) モードに応じてガイドを表示（追加ガイド/削除ガイド）
(5) 上段・下段のテキスト欄を作り直す（draw_text_rows_and_group_row）

上段/下段の Entry は「線の増減やリサイズ」で位置や幅が変わるため，再描画のたびに**破棄して作り直す**方式を採用している．その代わり，後述の「ID ベース保存」によってテキスト内容がずれないようにしている．

4.6.4 上段：区間テキスト（ID でずれを防ぐ）

上段の入力欄は「隣り合う 2 本の線の間」を 1 区間として，区間ごとに Entry を 1 つ配置する．内部的には i=0..len(self.lines)-2 について line[i] と line[i+1] の間に枠を作る．右端には隠し線があるため，見える線の本数と同数の区間が作れる．

線の追加・削除で self.lines のインデックスが変わると，単純な配列対応ではテキストがずれる．そこで本ツールでは，区間テキストを「左側の線の永続 ID（left_line_id）」で管理する．

- 保存：_top_text_map[left_line_id] = text
- 復元：Entry 生成時に left_line_id を見てテキストを戻す

また，左右矢印キーで入力欄間を移動できるよう，Entry を順序リスト（top_entries）に保持し，_focus_prev/_focus_next によりフォーカス移動を行う．これによりマウス操作の手間を減らし，連続入力がしやすくなる．

4.6.5 下段：グループ行（区間のまとめと境界操作）

下段のグループ行は，複数区間をまとめて一つのグループとして扱うための入力欄である．分割位置（境界）は group_boundaries に「線のインデックス」で保持する．

- 境界の候補：左端（0）と右端（隠し線）を除いた線位置
- 境界の有効範囲：1〜len(self.lines)-2（端は境界にしない）

分割モード ON のとき，下段をクリックすると「最も近い境界候補」を探し，近ければ（一定距離以内）その境界を **追加または削除** することで分割/結合を行う．この処理が _handle_split_click_local() である．また，線が少なすぎる場合（内部線が 3 本未満）には分割自体に意味がないため，何もしないよう条件分岐している．

下段 Entry の生成は _rebuild_group_row() が担う．境界列 group_boundaries から区間の開始点列 starts を作り，各区間（グループ）に対応する Entry を幅いっぱいに配置する．

下段テキストも上段と同様に left_line_id をキーに保存する．そのため，境界変更で Entry が作り直されても，どのグループの文字かが保たれやすい．

4.6.6 モード切替（追加・削除・分割）

本ツールには 3 つのモードがある．

4.6.6.1 追加モード（add_mode）

追加モードでは，線の下側に赤い「追加ガイド」を表示する（show_insert_guides()）．ユーザがガイドをクリックすると insert_line_at_index() が呼ばれ，右端の隠し線の手前に新しい線を挿入する．このとき，境界配列 group_boundaries も挿入位置以降を右に 1 つずらして整合性を保つ．

4.6.6.2 削除モード（delete_mode）

削除モードでは，線の上端に青いガイド（円）を表示する（show_delete_guides()）．上端付近をクリックするとその線を削除する．削除時には以下も同時に更新する．

- 下段のテキスト対応（_group_text_map）から削除線の ID を除去
- 親子スナップ関係の解除（detach_from_parent）
- 境界 group_boundaries のシフト（右側の境界を 1 つ左へ）

また削除モード中は，マウス位置に近い線を検出し，対応する上段入力欄をハイライトすることで「どれが削除対象になるか」を視覚的に示す．

4.6.6.3 分割モード（split_mode）

分割モードは下段グループ行の編集にのみ関係し，追加/削除モードとは独立して ON/OFF できる．ON の間は下段の背景色を変え，境界クリックによる分割/結合を有効化する．

4.6.7 線のドラッグとスナップ（階層構造の生成）

線の階層構造（親子関係）は，線の上端をドラッグし，他の線分へ近づけて離すことで作る．主要な流れは次の通りである．

(1) クリック位置が上端に十分近い線をドラッグ対象にする（on_press）
(2) ドラッグ中は Line.move(dx,dy) で上端を更新し，随時再描画する（on_drag）
(3) リリース時，対象線の上端が他線の線分に近いか判定する（is_near）
(4) 近ければ，親候補線上の等間隔スロット t のうち，未使用で最も近いものを選んで吸着する
(5) 近くなければ親子関係を解除する（detach_from_parent）

スナップ判定は「点（ドラッグ線の上端）」と「線分（親候補の根元〜上端）」の距離で行い，閾値以内なら吸着可能とする．

また，階層構造が循環（親を辿ると自分に戻る）すると追従が破綻するため，detect_cycle() で循環を検出した場合はその親候補を無視する．

スナップ位置 t の候補は generate_snap_t_values(len(self.lines)-1) により等間隔に生成される．used_t_values によって同じ t に複数の子が吸着しないよう制約をかけている．

4.6.8 Undo/Redo と状態復元（is_restoring の役割）

Undo/Redo は undo_stack と redo_stack に状態スナップショット（辞書）を積むことで実現している．状態変更前に push_undo_state() を呼び，現在状態を Undo スタックへ保存する．Undo 実行時は「現在状態を Redo へ退避」してから過去状態を復元し，Redo はその逆を行う．

復元処理 restore_state() の間は is_restoring=True とし，再描画に伴う _snapshot_current_texts() が「途中の UI」を誤って保存してしまわないようにしている．復元が完了したら is_restoring=False に戻す．

さらに，起動直後の状態を initial_state として保存しておき，編集メニューから「初期状態にリセット」できるようにしている．この操作も Undo 可能とするため，実行前に現在状態を Undo へ積んでから復元している．

4.6.9 JSON 保存形式（線・テキスト・グループ・表示範囲）

本ツールの状態を復元するには，線の幾何と親子関係だけでなく，上段/下段テキスト，グループ境界，表示範囲（viewport）も保存する必要がある．保存対象の主な要素を表 1 に示す．

表 1: 状態 (JSON) に含める要素

カテゴリ	項目（キー）	内容
線（Line）	lines[i].id（line_id）	永続 ID（親子・テキスト対応付けに使用）
線（Line）	base_ratio	根元 X の世界座標比率
線（Line）	top_ratio_x, top_ratio_y	上端位置（世界座標比率）
線（Line）	parent_id	親線の line_id（無ければ null）
線（Line）	snap_t	親線分上の吸着位置 t
上段テキスト	top_texts	left_line_id をキーにした辞書
下段（グループ）	group.boundaries	境界のインデックス配列
下段（グループ）	group.texts_by_left_id	left_line_id に対応するテキスト辞書
表示範囲（viewport）	view.xmin/xmax	X 方向の表示範囲（世界座標）
表示範囲（viewport）	view.ymin/ymax	Y 方向の表示範囲（世界座標）

ここで，JSON では辞書のキーが文字列になるため，保存時は left_line_id を文字列化して保持し，復元時に整数へ戻している．また復元処理では，古い形式のデータ（例：

view.min/max や group.texts）にも対応するため，キーが無い場合は互換処理を行うよう実装している．

4.6.10　スクリーンショット機能（キャンバスだけ保存）

スクリーンショット機能は，GUI 全体ではなくキャンバス領域のみを画像として保存する．winfo_rootx/y と winfo_width/height からキャンバスの絶対座標とサイズを取得し，その矩形（bbox）だけを Pillow の ImageGrab でキャプチャする．

保存形式は PNG/JPEG を選択でき，JPEG の場合は RGB 変換して保存する．Pillow が未導入の場合はエラーメッセージを表示し，pip install pillow を促す．

5.　ツールの機能一覧

ここまでどのようにツールを作成してきたか，プログラムをどのように組んだのかについて説明してきた．これらを踏まえた上で，最終的にどのような機能が実装できたのかを紹介していく．プログラムを起動した直後の初期画面は以下のようになっている．(図 13)

このように，本プログラムの画面は上部の操作パネルと下部のキャンバス領域から構成されている．操作パネルには，画像保存，モード切り替え，表示範囲指定，線数の操作ボタンを配置している．キャンバス領域には，複数の線分と，線分に対応する上段テキスト入力欄，線群の区間に対応する下段グループ入力欄を配置している．これらにどのような機能があるか，操作方法も踏まえながら挙げていく．

図 13: ツール起動時の初期画面

5.1　線の基本操作

各線は下端（根元）と上端（先端）からなる．下端は画面下部に固定され，上端のみを移動可能である．変更したい線の上端を左クリックし，押下したままマウスを移動すると上端を任意位置へドラッグ出来る．

木構造を作成するには，線の親子関係を作る必要がある．方法としては，子にしたい線の上端をドラッグし，親候補の線分上 (t 値でされたスロット) へ上端を近づけ，十分近い状態でマウスボタンを離すと，子線上端が親線へ吸着し，親子関係が作られる．この時，子線は親線に追従するようになる．

子線を再度ドラッグし，親線から離れた位置で離すと親子関係が解除される．解除後，その線は単独の線として扱われ，親に追従しない．

5.2　上段テキストボックス

対象のテキストボックスをクリックし，通常のテキスト入力として文字列を編集できる．また，入力中に左右キーを用いることで，隣接するテキストボックスへフォーカスを移動できる．また，テキストボックスはそれぞれ左上の線と連動している．(図 14)

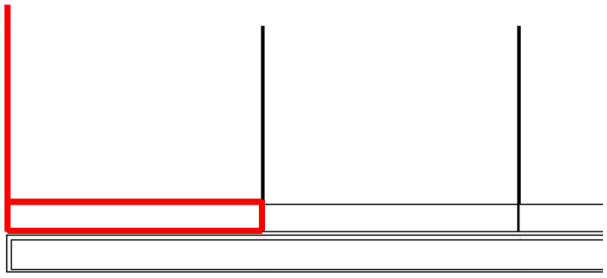


図 14: 線分とそれに対応したテキストボックス

5.3　下段グループ列

上段テキストボックスの下部にグループ入力欄が配置されている．下段グループ列は，線全体をいくつかの区間に分割し，各区間へ代表名やカテゴリ名を付ける用途を想定している．

下段グループの区間分割は，「分割モード」で行う．分割モードを有効 (図 15) にすると，下段グループ列の境界を置きたい，または削除したい付近をクリックした時，境界の追加・削除が可能になる．

図 15: 分割モード ON 状態

下段グループの各入力欄も，上段と同様にテキストボックスをクリックして編集，左右キーによる隣のグループ欄へのフォーカス移動が可能となっている．

5.4　追加モード

「追加モード」を ON にすると，線の下端付近に追加ガイドが表示される．(図 16) 利用者はガイドをクリックすることで，クリック位置に新しい線とそれに連動したテキストボックスを挿入できる．イベント間に新たなイベントを追加したい時に使用する．

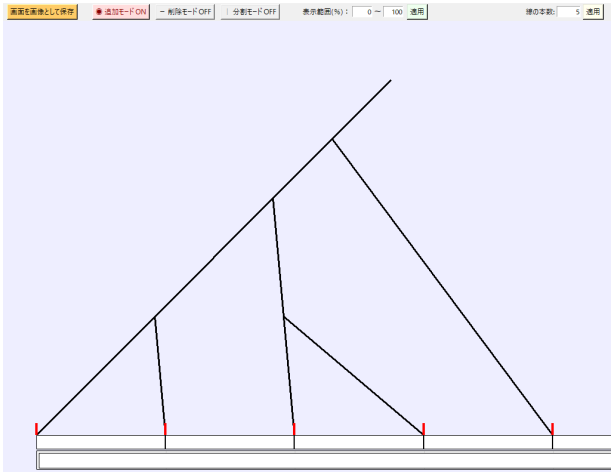


図 16: 追加モード ON 状態

5.5 削除モード

「削除モード」を ON にすると，各線の上端に削除ガイドが表示される．(図 17) 削除したい線のガイドをクリックすることで，その線とそれに連動したテキストボックスを削除できる．削除候補が分かりやすいよう，ガイドにカーソルを合わせた時に連動するテキストボックスの枠も強調表示される．

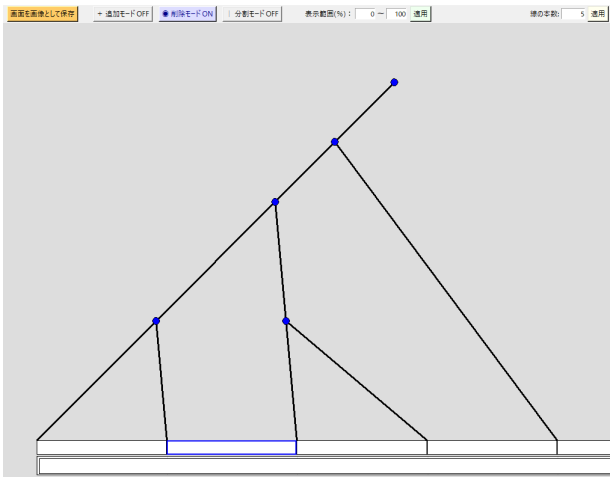


図 17: 削除モード ON 状態

5.6 線数の指定と増減

線の本数は，操作パネル右側の「線の本数」入力欄に数値を入力して適用することでも変更できる．また，「＋線を追加」「－線を削除」ボタンにより，1 本単位の増減も可能である．

5.7 表示範囲操作

マウスホイールによりズーム操作を行う事が出来る．ズームはマウスカーソル位置を基準として行われるため，注目したい領域を中心に拡大できる．右クリックのドラッグ操作により，表示範囲を上下左右へ移動できる．キーボードの左右キーでも左右操作が可能である．操作パネルの「表示範囲（%）」により，現在の表示範囲が確認できる．表示メニューから初期範囲へ戻すリセット機能を使用できる．

5.8 Redo / Undo と初期化

線の移動，接続，追加・削除などを操作履歴として復元できる機能である．Ctrl+Y により「Redo」，Ctrl+Z により Undo を実行する．編集メニューでも実行が可能である．(図 18) また，編集メニューには初期状態へ戻すリセット機能も実装している．このリセットは編集操作として扱われるため，Undo によりリセット前へ戻すことも可能になっている．

図 18: 編集メニュー選択時

5.9 状態の保存・復元

4.4 で述べたように作成した構造と入力内容をファイルとして保存，再読み込みにより復元できる機能である．ファイルメニューから実行が可能 (図 19) であり，選択後にファイルの保存場所，ファイル名などを決める．

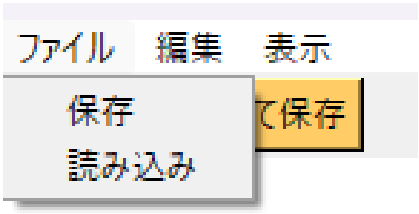


図 19: ファイルメニュー選択時

5.10 スクリーンショット

キャンバス領域のみを画像として保存するスクリーンショット機能である．操作パネルの「画面を画像として保存」を押し，保存先とファイル名を指定すれば画面の内容 (木構造) が画像として出力される仕組みになっている．

6. 性能調査

今回ツールを作成した目的は，木構造作成の簡略化及び時間の短縮である．そこで，PowerPoint を用いての作成とツールを用いての作成でどれくらい作業効率が上がったのかを調べることにした．

6.1 調査方法

PowerPoint とツールで，ランダム生成した木構造を 5 つずつ作成し，時間を計測するという手段を取った．線の親子関係は完全ランダム，上段テキストボックスには 1 から通し番号を振り，下段グループ列は分割位置をランダム，分割後のテキストボックスには gr1，gr2…といった通し番号が振られるようにした．この時，線の数を 8〜12 本で 1 本ずつ増やすことで，似た木構造がランダム生成される確率を減らした．また，PowerPoint とツールの作業を交互に行う事で，木構造作成作業に出来るだけ慣れが生じないように工夫した．(図 20)

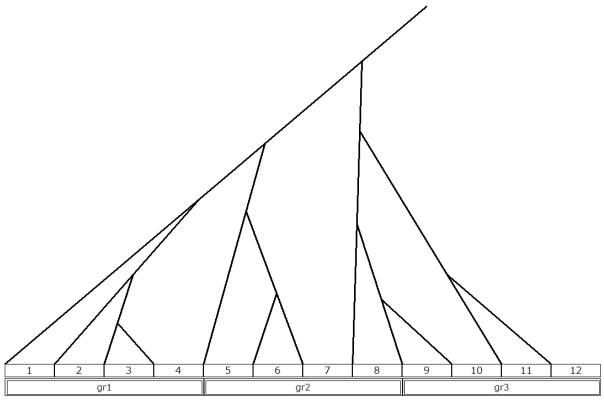


図 20: 計測時にツールで作成した木構造 (n＝ 12)

6.2 調査結果と考察

5 回ずつ測定した結果が表 2 である．

	PowerPoint	ツール
1 回目(n=8)	5:41	1:35
2 回目(n=9)	4:07	1:58
3 回目(n=10)	4:27	1:50
4 回目(n=11)	4:40	1:45
5 回目(n=12)	4:42	1:44

表 2: 計測結果 (単位は分)

1 回目の PowerPoint の時間が極端に長いのは表の出し方や線の配置に苦戦したためであるが，ツールを使用すればそのような事態も起きないと考え，記録に含んでいる．時間を比較した時に，PowerPoint に比べてツールを使用した時の時間は 2〜3 倍速くなっている事が分かる．また，PowerPoint を使用して木構造を作成する場合，線を 1 本動かすだけでも親子関係にある線や，全体の見え方を考慮した時の周りの線も 1 本ずつ手作業で移動させなければならない．今回はそのパターンが無かったためこのような時間差になっているが，あった場合はさらに時間の差が広がると考えられる．この結果より，今回作成したツールは，EMDA 分析における木構造の作成を大きく簡略化するものになったと言えるだろう．

7. おわりに

本研究では，EMDA 分析において大きな負担となりやすい「木構造の作成」に着目し，その簡略化と作業時間短縮を目的とした支援ツールを開発した．

私自身が PowerPoint で木構造を作成した際に感じた煩わしさは解消されたものの，使い方次第では不足している機能や不具合なども見えてくると考えられるため，ユーザーフィードバックを継続的に取り入れながら適宜改良していきたい．ツールがどのくらい作業を簡単にしたか，という点も作品や対象人物によって変化するため，テストデータを増やし，有用性を明確にしていきたいと考えている．

今回，EMDA 分析を行う際に不可避だが時間がかかる作業である木構造を対象に，専用のツールを作成してきた．本ツールが，EMDA 分析の作業効率を高めるだけでなく，これまで EMDA 分析をしていなかったユーザーにとって，分析へのハードルを下げる一助となれば幸いである．

参考文献

[1] 寛也三浦，理美森，確長尾，圭二平田，Hiroya, M.，Satomi, M.，Katashi, N.，Keiji, H.: 音楽理論 GTTM に基づく議論タイムスパン木の生成方式とその評価，情報処理学会論文誌， Vol. 56, No. 3, pp. 942–950 (2015).

[2] 晴弘片寄，光代橋田，なみ飯野: Emotion Movement Design Annotator による感動デザインの分析 - M. Rigolo のバランス芸を例として-，エンタテインメントコンピューティングシンポジウム 2019 論文集， Vol. 2019, pp. 267–274 (2019).