ヒューマンインタフェース 84−12
情 報 メ デ ィ ア 35−12
（1999. 8. 20）

ヒストリーグラフを用いたアンドゥ機構の提案と評価

西田 知博† 林 真志†† 辻野 嘉宏††† 都倉 信樹†
†大阪大学, ††松下電器産業, †††京都工芸繊維大学

過去の作業状態の遷移をグラフとして表現したヒストリーグラフで示し，各状態を直接選択することによってその状態に移行できるインタフェースを提供するアンドゥ機構を提案する．ここでは，アンドゥは間違った操作の取消しではなく，過去の状態を回復するための操作としてとらえられる．また，この機構を実際にグラフィックエディタに実装し評価を行った．このグラフィックエディタでは，ユーザに操作の流れがわかりやすい形でヒストリーグラフを提示し，各状態を表すノードをマウスでクリックすることによって自由に過去の状態に移行することができる．それぞれの状態はマウスカーソルを各ノード上に置くことによってプレビューが可能で，戻りたい状態の探索はグラフをマウスでなぞるという簡単な操作で行うことができる．

An Undo Mechanism with History Graph and its Evaluation

Tomohiro Nishida† Chikashi Hayashi†† Yoshihiro Tsujino††† Nobuki Tokura†
t-nisida@rd.ecip.osaka-u.ac.jp
†Osaka University, ††Matsushita Electric Industrial, †††Kyoto Institute of Technology

We propose a new undo mechanism which users can go back to all past working states with an easy operation. In order to give users an easy way to go back any working states, the new mechanism provides graphical presentation of the states for direct selecting. We introduce a graphical presentation, "history graph" which vertexes show working states and edges show transitions between states. We made a graphic editor with history graph undo and evaluate it. It provide good presentation of history graph to show users' work flows clearly. Users can preview each state by moving their cursors on each vertex, so they can search and browse past states by tracing the graph.

1 はじめに

直接操作を用いるグラフィカルユーザインタフェース (GUI) では，マウスクリックなどのちょっとした操作が，大きな状態の変化を引き起こすことがある．このような場合，ユーザは何が起ったかさえわからなくなり，混乱することも少なくない．このような例をはじめ，誤操作によって生じた状態を回復するためには，行った操作を取り消すというアンドゥ(undo)機構は非常に重要である．アンドゥを用いることによって，ユーザは間違いを正すために自力で元の状態を再現する必要はなくなり，操作効率を上げたり，負荷を軽減することができる．またユーザはアンドゥが行えることで間違いを恐れずに気楽に操作を行えるようになるという利点もある．

また誤りを修正するだけでなく，いろいろな操作を試してみるための機能としてもアンドゥを用いることもある．すなわち，ユーザはある操作を取り消せることを前提に行い，思った結果が得られない場合はそれをアンドゥして別の操作を行うことが可能になる．この場合，アンドゥは「間違いの修正」ではなく,「過去の状態の回復」という意味で利用される．

ここではアンドゥ機構を間違いの修正を行うための「操作の取消し」としてだけ提供するのではなく,「過去の状態の回復」を行うための操作としてとらえたモデルを考える．このために，過去の作業状態とその遷移をグラフとして表したヒストリーグラフを導入する．このヒストリーグラフを用いることにより，ユーザが状態を表すノードをマウスクリックなどの１回のアクションで過去の状態に移行できるインタフェースを提供できる．また，過去の状態遷移をグラフィカルに表示することにより，ユーザに過去の状態を整理した形で提示することができる．

以下ではまず既存のアンドゥ機構について紹介し，その後，本研究で提案するヒストリーグラフを用いたアンドゥ機構について述べる．

2 既存のアンドゥ機構

2.1 線形アンドゥと非線型アンドゥ

線形アンドゥ(linear undo) は，操作を線形の履歴リストとして保持し，直前に行った操作から順にリスト上の操作がアンドゥできるというものである．アンドゥされた操作は，取り消した操作を再度実行するリドゥ(redo) 操作が可能なように，リドゥリストの中に保持される．線形アンドゥにおいて希望した状態に戻るためには，アンドゥを繰り返し，生成された状態を確認するという単純な操作を繰り返すだけですみ，容易に利用できるという利点がある．しかし一度の操作でアンドゥできるのはその直前の操作のみであるので，希望の状態まで戻るためにはアンドゥの操作を繰り返していかなければいけないという手間がかかる．また，リドゥリストにはアンドゥした操作がすべて保持されるので，再実行が出来ないような操作がリドゥリストに残ってしまうこともありうる．そこで一般には，新しい操作が行われた時点でリドゥリストを空にし，再実行不可能なリドゥ操作が出来ないようにした，制限線形アンドゥ(restricted linear undo) が用いられる．

線形アンドゥに対し，過去の任意の操作をアンドゥできるようににしたものが，非線形アンドゥである．非線型アンドゥの初期の研究としては，スクリプトモデル [1] や US&R モデル [5] などがある．スクリプトモデルは過去の操作列をスクリプトとして保持し，そのスクリプトを編集することによって，過去の作業のアンドゥを行う．このモデルは非常に強力な方法ではあるが，スクリプトを直接編集するという操作はユーザにとっては困難な作業であるため，一般のインタフェースとしては不向きである．US&R モデルは非線型アンドゥを実現するためにアンドゥしたくない作業状態をスキップして目的の状態のアンドゥを行うというものである．このモデルはスクリプトモデルに比べ操作が容易であるという利点があるが，スキップした状態はそのまま保持されリドゥが可能になっているため，保存している作業状態の関係が複雑になり，ユーザがそれを把握できなくなるという危険性がある．

2.2 選択的アンドゥ

選択的アンドゥ(selective undo)[2] とは，過去の任意の操作を選択してその操作を取り消すものである．このとき選択した操作の対象がすでに存在しない場合など，アンドゥが意味をなさない場合がありうるが，このときにはその操作はアンドゥの対象として選択できないようになっている．

カーネギメロン大学で開発されたユーザインタフェース開発環境である Amulet[4] はこの選択的アンドゥを標準の機能として提供している．図１はそのアンドゥ操作パネルである．各操作はコマンドテキストの形でリストボックスに表示され，それぞれに対してアンドゥやリドゥが指定できるようになっている．しかし，テキス

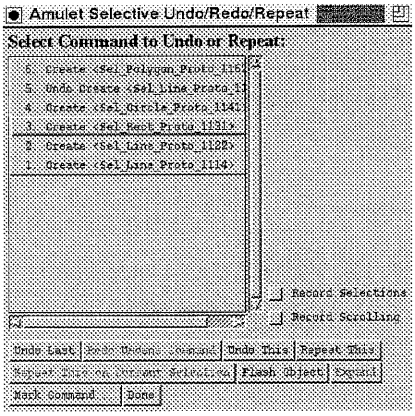

図 1: Amulet で提供している選択的アンドゥ

トで表現されたコマンドはそれぞれがどの操作に相当するのかがユーザには特定しにくい欠点がある．[3] では，これを解決するために，コマンドテキストのかわりに各作業状態のスナップショットを用い，グラフィカルな形でユーザに提示するという手法をとっている．しかし，この場合にはスナップショットをどのような形でユーザに提示するかや，一度に提示できるスナップショットの数がテキストよりも制限されることなどが問題となる．

3 ヒストリーグラフを用いたアンドゥ

3.1 設計

本研究で提案するアンドゥ機構は，アンドゥを「操作の取消し」としてではなく，「過去の状態を回復するための操作」としてとらえて考える．回復したい状態に戻るためには，線形アンドゥのように一つ一つ状態をさかのぼる方法もあるが，ここでは過去のどの状態もマウスのクリック等の 1 回のアクションで復元できるようにすることを考える．また，任意の過去の状態に戻れるようにするためには，過去の状態をすべて保持しておく必要があるが，作業を進めるにしたがい，その関係は複雑になる．したがって，ユーザにとって過去の作業状態がどのような形で遷移しているかをわかりやすく提示し，戻りたい作業状態を特定しやすいようにしなければいけない．

このようなアンドゥ機構を考えたとき，もっとも重要な問題は，選択すべき過去の作業状態をどのようにして表現するかである．ここでは過去の操作状態をノードとして表現し，それに連続する作業状態との間を辺として結んだグラフ (これをヒストリーグラフと呼ぶ) によって表現することを考える．このヒストリーグラフを提示することにより，ユーザが各ノードを直接クリックすることによって過去の状態に移行できるようなインタフェースを提供することができる．

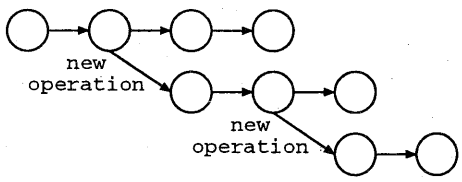

図 2: ヒストリー木

作業状態とその遷移を表したグラフは，アンドゥである状態に戻って新しい操作を行った場合，新しい状態を表すノードを生成しグラフを枝分かれさせる形になるので一般に木構造 (ヒストリー木) となる (図 2)．このようなヒストリー木は 1 操作ごとにノードが生成され，枝分かれも多くなるので，その構造が複雑になって，ユーザが把握できないものになる可能性がある．しかし，新しい操作によって出来る状態が必ずしも過去の状態と異なっているとは限らない．したがって「状態」を基本とした考え方の上では，新しい操作を行った場合でも，過去の状態と同一であるならば，新しい状態が生成されたのではなく，その操作によって過去の状態に戻ったとみなすことが妥当である．一方，これをユーザの立場から考えた場合，新しい操作を行ったにもかかわらず過去の状態に戻るということには違和感を感じるかも知れない．しかし，この

ように過去の状態と同一になったことを意識させることによって，ユーザは新しいメンタルモデルを形成することができ，過去に行ってきた作業をよりよい形で整理できるようになると考えられる．

そこで，ここで用いるヒストリーグラフでは，その状況を作り出すための操作手順が異なっても，対象となるオブジェクト等の構成要素がまったく同じである状態は同一であると考え，一つのノードとして表現する．その結果，アンドゥ以外の操作した場合でも過去の作業状態と同一の状態を作り出すものであるならば，ヒストリーグラフでは新しい状態のノードを追加せず，過去の状態への辺のみを生成する．

また，選択的アンドゥなどではアンドゥ操作そのものが履歴として残り，それが過去の状態履歴を複雑にする一因となっている．一方，ここでのアンドゥは過去の状態を選択し，その状態に移行するだけ操作であり，アンドゥ操作はヒストリーグラフ上には追加しないようにすることで，状態の履歴が複雑化することを避けている．

3.2 グラフィックエディタへの実装

ヒストリーグラフを用いたアンドゥのアプリケーションとして，ここではグラフィックエディタ（ドローツール）への実装を考える．

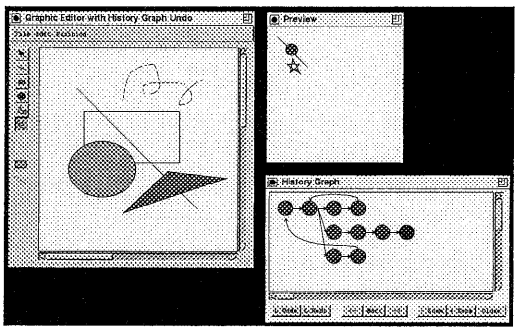

図 3: ヒストリーグラフアンドゥを実装したグラフィックエディタ

今回実装したグラフィックエディタは図3のようなもので，グラフィックエディタ本体と，作業状態の遷移を示すヒストリーグラフ，および状態を確認するためのプレビューの 3 つのウィンドウからなる．

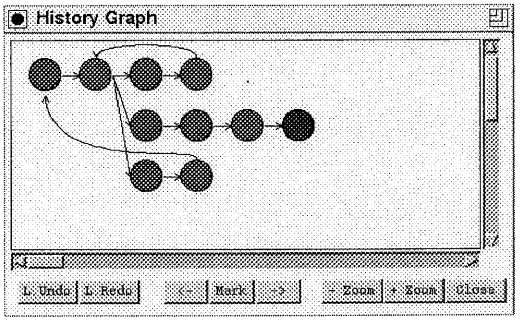

図 4: ヒストリーグラフ

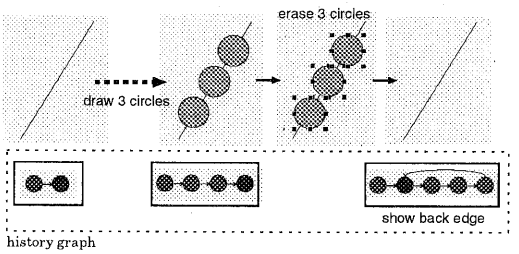

図 5: 過去の状態への遷移の生成

ヒストリーグラフの提示法はいろいろな形が考えられるが，ここではユーザが作業の流れを容易に把握できることを主たる目的とし，図 4 のような形でユーザに提示する．グラフの配置は，基本的に左から右へ時系列の流れを表現し，アンドゥ後の新規操作により新たな作業系列が始まった場合は，下方向にグラフを枝分かれさせた形で新たな系列を表示する．図5に示した例はオブジェクトを消去することで過去の状態と同一となった場合である．このようにアンドゥ以外の操作によって過去の状態と同じものとなった場合は，新しいノードは作成せず，同一である過去の状態へ戻るような遷移辺のみを作る．この

ような遷移辺を追加した結果，既存の辺やノードと交差することもある．そのような場合はグラフのノードと辺の配置を再計算し，交差等が少なくなるよう再描画する方法も考えられるが，ユーザがグラフの形で作業の流れを認識している場合は，その変化によって流れを把握できなくなってしまう危険性もある．したがって，ここでは新しい状態や遷移辺の追加があった場合も，既存のノードや辺の位置は一切変更しないこととした．

各ノードが表す状態が何であるかをユーザに提示する方法については，グラフのノードや辺に状態に関する情報を記述する方法が考えられるが，ディスプレイ装置の解像度等の制約を考えると，画面上に提示できるヒストリーグラフの状態数が少なくなり，グラフの構造がわかりにくくなるため好ましくない．そこでここでは，ヒストリーグラフは状態の遷移のみを表すものとし，それぞれのノード上にマウスカーソルを置くことによって，別のプレビューウィンドウに，その状態においてのグラフィックオブジェクトの縮小画面が表示されるという形にした．これにより，ユーザはヒストリーグラフ上をマウスカーソルでなぞることによって状態の変化を見ることができる．そして，希望する状態を見つけたときにノードをクリックすることによって，プレビューされている状態に戻ることができる．

また，状態が増加してグラフのサイズが大きくなった場合に対応するため，ヒストリグラフの表示には拡大，縮小の機能を持たせている．さらに，ユーザが作業の基点と考えている状態にマーク付けをすることができ，それらのマークされた状態間をボタンで遷移することも可能である．なお，線形アンドゥ／リドゥボタンを付加することにより，ボタンを順にクリックすることによって操作の順に１つずつ状態を遷移させることも可能にしている．

4 評価

4.1 他のアンドゥ機構との比較

ここでは，提案したアンドゥを線形制限アンドゥおよび選択的アンドゥと比較する．

過去の作業状態への移行手順

ヒストリーグラフを用いたアンドゥでは，任意の過去の状態への移行をマウスを１回クリックするだけで行うことができる．これに対し，制限線形アンドゥでは１つ前の状態へのアンドゥを繰り返さなければいけない．また選択的アンドゥの場合は，任意の過去の１操作を１回のアクションで取り消すことはできるが，過去の状態に戻るためには線形アンドゥと同様に，移行したい状態から現在までに行った操作を逆順に取り消していく操作が必要となる．

任意の作業状態に移行可能か？

ヒストリーグラフを用いたアンドゥでは，すべての作業状態を保持しているので，任意の状態に戻ることが可能である．制限線形アンドゥにおいてはアンドゥ後，新しい操作を行った時点で，そのときのリドゥリストは空にされる．このため再実行不能な操作が発生して，戻れない状態が発生する．また，選択的アンドゥでは逆の手順で操作のアンドゥを繰り返すことによって過去の作業状態に戻ることができる．しかしこの場合は，その状態から現在までのすべての手順を記憶する必要があり，ユーザへの負荷は高い．

アンドゥ操作の利用しやすさ

ヒストリーグラフを用いたアンドゥではグラフ上でマウスカーソルでなぞるという操作で各状態を連続的にプレビューすることができ，操作の流れに沿った形で手早く，希望の状態を見つけることが可能である．また，実際にマウスクリックをしてアンドゥを行っても，もう一度，もとの状態をクリックするだけで状態を戻すことができるので，気楽な形で操作が行える．制

限線形アンドゥでは1つ前の操作に関してしかアンドゥまたはリドゥできない．しかし，そのため操作は単純でかつ，ユーザは変化する状態を確認していくだけでよいので，利用しやすさの面では優れている．選択的アンドゥでは，行った任意の操作に対してアンドゥできるが，作業が進むごとに，アンドゥ可能な操作が増え，どの操作をアンドゥすべきかがユーザにはわからなくなってしまう危険性がある．

任意の操作の再利用

選択的アンドゥでは過去の任意の操作を再利用することができる．一方，制限また，線形アンドゥではアンドゥした操作を再び実行することのみが可能である．ヒストリーグラフを用いたアンドゥは，状態のみを対象としたものであるため，操作の再利用はできない．ただし，ヒストリーグラフにおいても遷移辺が操作を表すため，これを用いて操作の再利用を行うことは可能である．ただしこの場合は，インタフェースが複雑になる可能性もあるので，慎重な検討が必要である．

4.2 適用範囲

ヒストリーグラフを用いたアンドゥはすべての状態を保持する必要があるため，それを記憶する領域の負担が大きくなる．今回のようなドロー系のグラフィックエディタの場合，各状態は線や円等のグラフィックオブジェクトの列として保持されるが，それに要する記憶領域はそう大きくない．しかし，ペイントツールでの実装を考えると，その各状態をビットマップのイメージとして保持しておく必要があるため，かなりの記憶領域を要することになる．しかし最近では，パソコン等でも記憶デバイスの容量はかなり大きくなっているため，多大のリソースを必要とはするが，実装することは不可能ではない．WWWブラウザに関しては，状態はURLという形で扱うことが可能であるし，プレビューも一般にブラウザに実装されているキャッシュを利用すればいいので，アンドゥ機構としては記憶量の負荷をかけることなく実装が可能である．

5 まとめ

過去の作業状態の遷移を表すヒストリーグラフを導入して，これを利用したアンドゥ機構を提案した．従来の方式とは異なり，ここではアンドゥを操作の取消しではなく，過去の状態を回復する操作であるととらえる．このため，任意の過去の作業状態に1回のアクションで移行できるよう，すべての作業状態とその遷移をグラフの形で表現し，ユーザインタフェースとして提供する．

ここで示したグラフィックエディタへの実装では，グラフ上をマウスでなぞるという簡単な操作で，各ノードが表す状態を連続的にプレビューしながら戻りたい状態を探すことが可能で，目的の状態を見つけた場合はそのノードをマウスクリックするだけで移行することができる．

本手法に関する今回の評価は定性的なものであったが，今後，実験などを行うことにより，さらなる評価を行いたい．

参考文献

[1] James E. Archer et al.: "User recovery and reversal in interactive system", ACM Transactions on Programming Languages and Systems, Vol. 6, No. 1, pp.1-19(1984).

[2] Thomas Berlage: "A Selective Undo Mechanism for Graphical User Interfaces Based on Command Objects", ACM Transactions on Computer Human Interaction, Vol. 1, No. 3, pp.269-294(1994).

[3] Meng Chii et al.: "Visualization of Selective Undo/Redo History", 情報処理学会ヒューマンインタフェース研究会報告, 71-3, pp. 15-22(1997).

[4] Brad A. Myers et al.: "The Amulet Environment: New Models for Effective User Interface Software Development", IEEE Transactions on Software Engineering, Vol. 23, No. 6. pp.347-365(1997).

[5] Jeffrey S. Vitter: "A new framework for redoing", IEEE Software, Vol. 1, No. 4, pp.39-52(1984).