HyperSnapping – スナッピングを活用した図形編集手法

増井俊之

ソニーコンピュータサイエンス研究所

masui@acm.org

http://www.csl.sony.co.jp/person/masui/

図形をエディタで美しく揃えて編集しようとするとき、格子状のグリッドや他の図形を基準として配置を行なう「スナッピング」機能がよく用いられる。スナッピング機能を拡張し、特別な操作を用いることなく適応的にグリッドを調整したり図形配置の制約条件を指定したり図形をグループとして操作したりできるようにした「HyperSnapping」手法について報告する。

HyperSnapping

Toshiyuki MASUI

Sony Computer Science Laboratories, Inc.

masui@acm.org

http://www.csl.sony.co.jp/person/masui.html

We introduce a new drawing technique called HyperSnapping. In many drawing editors, various operations are provided for object alignment, and snapping is one of the most frequently used techniques, where users can snap the mouse cursor or objects to existing objects or square grids. HyperSnapping is an extension of snapping operations, where users can control the behavior of snapping grids and the constraints between objects only by dragging snapping objects, without explicitly changing drawing modes. With HyperSnapping, simple constraints and drawing macros are easily constructed for later editing operations.

1　はじめに

図形エディタで美しい図表や Web ページを作成しようとする場合、図形をきれいに揃えて配置する作業がよく必要になる。望ましい配置を保ったまま編集作業を行なえるように、ユーザが定義した「図形の間隔を常に一定にする」などの制約条件を常に満たすような配置を行なう編集システム (制約型図形編集システム) が多数提案されているが、制約の定義が面倒なため、現実に使われているものは少ない。格子状のグリッド上にのみ図形を配置できるような制約を用いる方法・別の図形に揃うように図形を描いたり移動したりする方法・選択した複数の図形の端や中心を揃えるコマンドを用意する方法・移動を水平 / 垂直に制限する方法などがよく利用されている。

描画 / 編集中の図形がグリッドや他の図形に揃うような制約をうける手法では、描く図形の制御点がグリッドや他の図形の重力に引きつけられるように感じられたり、移動中の図形がグリッドや他の図形に引きよせられてパチッとくっつく (スナップする) ように感じられたりするので、このような操作はスナッピングと呼ばれる。

スナッピング操作は単に図形を揃えるために使用されるのが普通であるが、ユーザの行なうスナッピング操作からユーザの意図をシステムが適切に解釈することができれば、それをもとにした高度な編集作業を行なうことが可能になる。たとえば図形 B を移動して図形 A にスナップさせて揃えた場合、たまたま A の位置が都合が良かったからというよりも、 A と B は関係が深い (A の位置と B の位置に制約条件が存在する) からスナッピング操作を行なったと解釈することも可能である。これがユーザの意図と一致していた場合、 A の位置を変えた場合は B も同時に変えると都合が良いであろう。この場合、 B を移動して A にスナップするというひとつの作業で B と A の制約関係が定義できたことになる。また、似たようなスナッピング操作を繰り返し実行した場合、それらの操作に共通する処理を抽出することにより自動的にマクロとして定義することもできる。このように、意図をユーザがはっきり認識しながらスナッピング操作を行なうことにより、通常の編集操作を行ないつつ、ユーザの意図をシステムに指令してその後の編集作業に活用することができるようになる。本稿では、**ユーザの意図をシステムに伝えるためにスナッピング操作を活用する**、拡張されたスナッピング手法「HyperSnapping」について述べる。

2　HyperSnapping の操作

HyperSnapping では、モード切替操作をユーザが行なうことなく、図形を動かすスナッピング操作のみで制約などを指定できるような工夫を行なっている。以下に、例を用いて HyperSnapping の機能を説明する。

通常のドラッグ操作

図形上でマウスをクリックして動かすことにより図形を直接ドラッグすることができる。図 1は矩形をドラッグして任意の位置に移動しているところである。通常はマウスの動きに従って矩形が移動する。

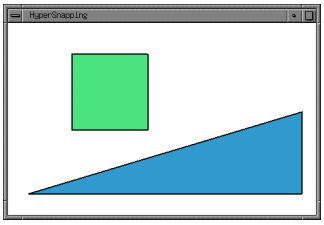

図 1: 矩形をドラッグ

グリッドの制御

一定の距離を越えてドラッグを行なうと背景にグリッドが出現し、図形はグリッドにスナッピングしながら移動する。グリッドへのスナップ点は小さな矩形で表現されている。以下ではスナップ点をアンカーと呼び、アンカーを持つ図形をアンカー図形と呼ぶ。アンカーは後で解説する様々な操作において基準点として働く。スナッピングの候補点が複数存在するときはドラッグの向きによりアンカーが選択される。

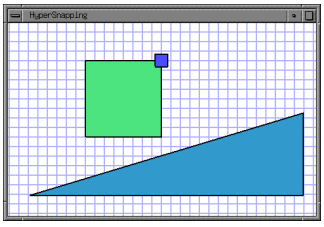

図 2: 小さなグリッドにスナッピング

さらにドラッグを続けるとグリッドのサイズが段階的に大きくなる。遠くまでドラッグするとスナッピングが行なわれる単位が大きくなり、ドラッグ距離が小さいときは小さな単位でスナッピングが行なわれることになる。

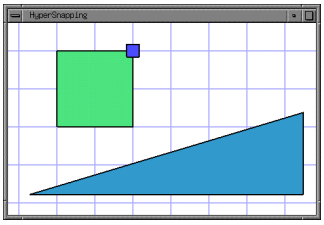

図 3: 大きなグリッドにスナッピング

このような機能により、図形を遠くに移動する場合はおおかまな位置に揃え、近くに移動するときは細かく制御することが可能になる。

グリッドの基準点の制御

グリッドは以前の操作においてアンカーを持っていた図形を基準に生成される。前の操作において三角形がアンカー図形となっていた場合は、図 4のように三角形の大きさにもとづいてグリッドが生成されるので、三角形の縦横にあわせた位置に簡単に別の図形を配置することができる。

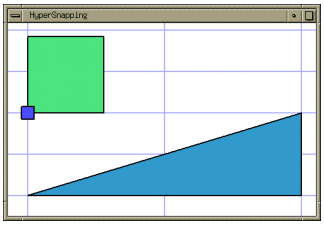

図 4: 別の図形を基準としたグリッド使用

アンカーによる回転 / 拡大中心の指定

ドラッグ中の図形は、グリッドだけでなく他の図形の頂点や辺にもスナップする。

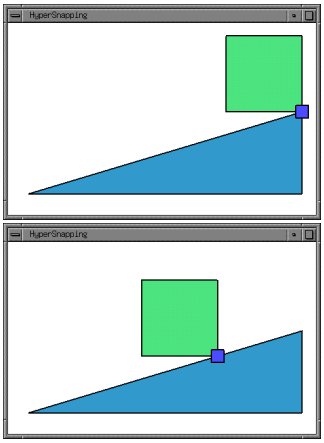

図 5: 頂点や辺へのスナッピング

グリッドへのスナッピングの場合と同様に、図形のドラッグ距離が小さい場合はスナッピングは起こらず、ある程度図形を移動したときのみスナッピングが起こる。図形 A を図形 B からほんのわずか離れた位置に移動させたい場合は、まず A を移動して B にスナップさせてから、ドラッグをやりなおしてもう一度 A をわずかだけ移動させればよい。

図形の頂点付近をクリックして移動することにより、アンカーを中心として図形の回転 / 拡大を行なうことができる。クリックを行なわずにに図形上でマウスを移動させると、クリックした場合の操作が影として表示される。図 6下側はアンカーの反対側の頂点を回転させようとしているところである。

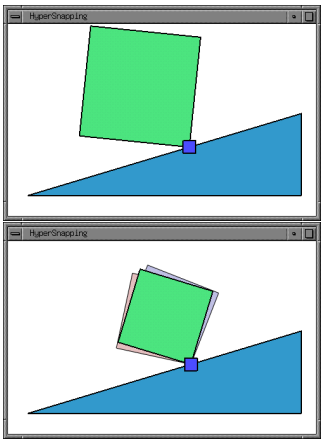

図 6: アンカーを中心とした拡大 / 回転

図形の回転時や拡大時も、移動時と同様に他のオブジェクトやグリッドへのスナッピングが行なわれるので、図形を他の図形の辺に揃えることができる。

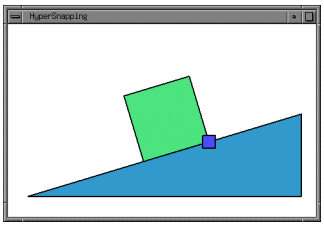

図 7: 回転して斜辺に揃える

グループ操作と制約の自動生成

アンカー図形に別の図形をスナップさせると、その図形とアンカー図形の間に制約関係が生成され、図形グループとしての操作が可能となる。

図 8の状態で、中央の矩形を移動して左の矩形にスナップさせると図 9のように白い小さな矩形でスナップ点 (サブアンカー) が表示される。

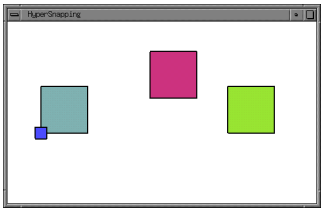

図 8: 3 個の矩形の初期状態

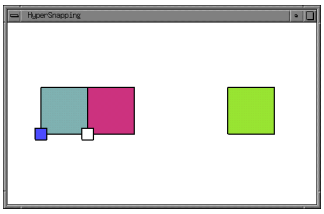

図 9: 中央の矩形を左の矩形にスナップ

右の矩形も移動して真中の矩形にスナップさせると図 10のようになる。これらの 3 個の矩形間には制約関係が自動的に定義される。たとえば右または中央の矩形を移動すると、図 11のように他の矩形もそれに従って移動する。

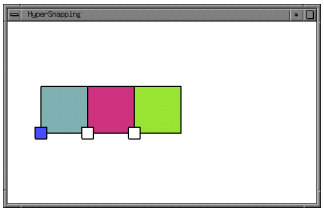

図 10: 右の矩形を中央の矩形にスナップ

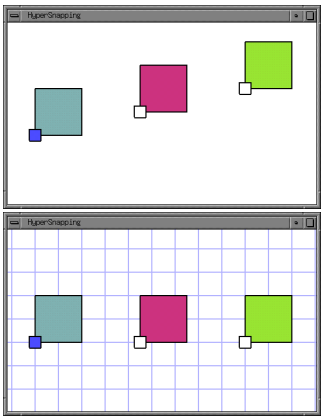

図 11: 右の矩形をドラッグ

アンカー図形を移動した場合は図 12のようにサブアンカー図形も同時に移動する。

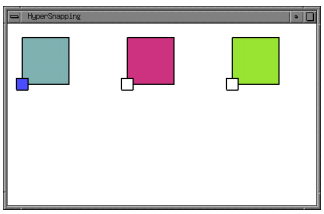

図 12: アンカー図形をドラッグ

また、サブアンカーを回転させると図 13のようにアンカーを中心として全体が回転する。

アンカー図形を拡大 / 回転した場合はすべての図形が同じように拡大 / 回転される。

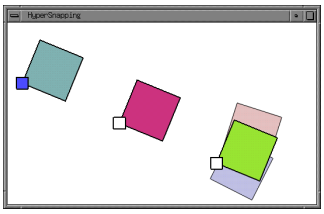

図 13: サブアンカーを回転

図形以外の場所をクリックするとアンカー / サブアンカーは全て消去され、制約関係も解消される。

繰り返し操作

図 14は 1 個の矩形をコピーしてからペースト操作により新たな矩形を生成してもとの矩形にスナップさせたところである。また同じ操作をもう一度実行すると図 15の状態になる。

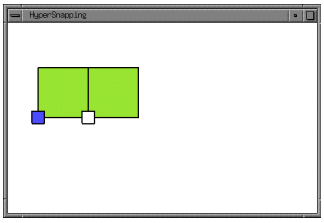

図 14: 矩形をコピー / ペーストしてもとの矩形にスナップ

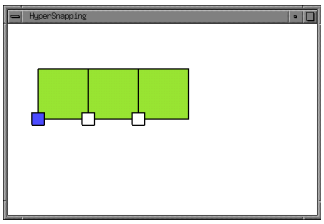

図 15: 同じ操作を繰り返した状態

このように同じ操作を 2 度以上繰り返した場合、 Dynamic Macro[4] と同様の手法を用いて次の操作を予測 / 実行することができる。この状態で「繰り返し実行」を指令すると、操作履歴中の繰り返し操作が自動的に抽出され、図 16のように再実行される。

予測により生成された図形の間には制約関係が成立しているため、サブアンカー矩形を移動すると図 17のように他の矩形も移動する。

他の描画例

スナッピングを用いると、図 18の「タングラム」のような傾いた図も簡単に描くことができる。

図 19は三角形の斜辺に 4 個の矩形を等間隔で並べようとしているところである。制約やスナッピングの機能の

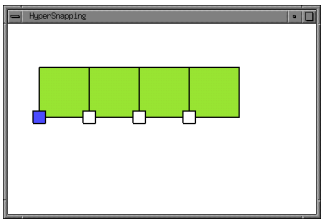

図 16: 繰り返し操作指令による再実行

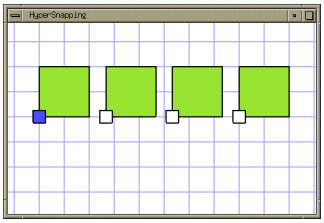

図 17: サブアンカーを移動

無いエディタではこういった図形を描くことは簡単ではないが、 HyperSnapping ではメニューなどを全く用いることなくこのような図形を描くことが可能である。

3 評価

HyperSnapping システムは現在開発中の段階でありユーザ評価などはまだ行なわれていない。

上で述べた HyperSnapping の利点をまとめると以下のようになる。

- 図形操作のみでグリッドやスナッピングを制御可能
- メニューなどを使用することなくグループ化や制約を定義可能

これらの結果として、斜線に沿った等幅の矩形 (図 19) のような図を簡単に描くことが可能になっている。

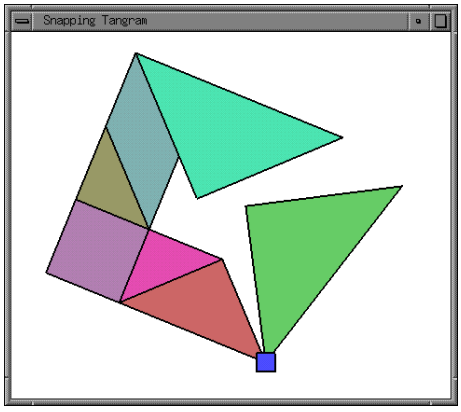

図 18: タングラム

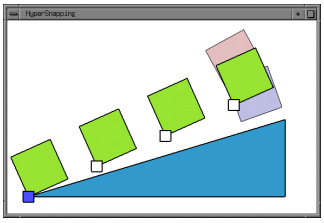

図 19: 三角形の斜辺に矩形を並べる

また、 HyperSnapping には

- スナッピングでシステムに意図を知らせることにより例示インタフェース (Programming by Example) の手法を適用可能

という特徴がある。例示インタフェースの手法を GUI 操作に適用する場合、ユーザが何を考えて操作しているのかがわからないためにユーザの意図を汎化することが難しいという問題があった。たとえばユーザが図形を画面の左の方に移動したような場合、左に移動することに意味があるのか / 移動距離に意味があるのか / 枠に近づけることに意味があるのか / などがわからないため、このような操作を汎化してプログラムを作成することが非常に困難であるが、 HyperSnapping の場合は、スナッピングを行なう対象の違いによってこのような意図の違いを自然に表現することができるので汎化によるプログラム作成が容易になると考えられる。

一方、 HyperSnapping は以下のような問題点がある。

- スナッピングに対応した制約をユーザが変更できない
- スナッピングで定義されたグループや制約は一時的なものであり別の作業を行なった後で再利用することができない

4 関連研究

既存の図形を基準としたスナッピングを行なう描画システムとして Snap-Dragging[1] がよく知られており、それをもとにした各種の拡張システムが提案されている。たとえば Adobe Illustrator 8.0 では Snap-Dragging が実際に取り入れられている。

本田らは、移動 / 拡大 / 回転などのモードを変更することなく図形編集を簡単に行なうことができるような Integrated Manipulation という操作手法を提案している [3]。 Integrated Manipulation では、図形の回転中心 (ピボット) を別の図形にスナップさせることができ、周囲の図形の環境によっては図形を移動しながら回転してスナップさせることもできるようになっている。 HyperSnapping では移動と回転の同時実行は支援していないが、アンカーを用いた移動と回転の 2 段階のスナッピングで同様の編集を行なうことができる。

制約を用いた描画システムは、 SketchPad[7]、 Juno[6] をはじめ数多くのものが提案されており、制約解決アルゴリズムも色々なものが研究されているが、商用のものも含め、制約関係を指定するための操作が面倒であるため広く使われているものは存在しないようである。

Briar[2] は Snap-Dragging 操作により描画した図形の間で制約を指定することのできる制約型図形編集システムで、スナッピング操作によりユーザの意図を知るという考え方が HyperSnapping と共通している。スナッピング操作後の指定により制約条件を指定することができるが、汎用の制約型図形編集システムとしているため、制約の種類などをユーザが指定する必要がある。 HyperSnapping は、使用できる制約は非常に限られているが、制約の作成に関してユーザは全く指定を行なう必要がない。

MetaMouse[5] は、編集中の図形の接触関係をもとにして推論を行なうことによりプログラム (プロダクションルール) を生成する例示プログラミングシステムである。図形編集作業中に線の交点などが出現した場合、そのどれが重要であるかについてシステムがこまめにユーザに質問を行なうことにより、ユーザの意図を正しく推論してプログラムを生成する。例示プログラミングモードにおいて正しい推論を行なうためにこのような工夫が必要になっているが、 HyperSnapping では制約が簡単なものに限られているため、プログラミングモードやユーザへの質問は不要である。

これらのシステムに限らず、スナッピング編集 / 制約指定 / 例示プログラミングに関して各種の複雑なシステムが存在する。 HyperSnapping は、複雑な制約を定義したり複雑な例示プログラミングを行なったりすることはできないが、通常の編集作業の枠組をわずかに拡張しただけで、図形編集作業においてしばしば必要になるような、単純な制約を用いた編集や繰り返し編集操作などを、特別な手間を必要とせずに実行できるという特徴がある。

5 結論

図形編集時のスナッピング操作を拡張し、様々な編集作業に有効な HyperSnapping システムを提案した。図形エディタとしての基本的な機能を実装し、通常の図形エディタとの比較評価を行なう予定である。

参考文献

[1] Eric Allan Bier. Snap-dragging. *Computer Graphics*, Vol. 20, No. 4, pp. 233–240, August 1986.

[2] Michael Gleicher and Andrew Witkin. Drawing with constraints. *The Visual Computer*, Vol. 11, No. 1, pp. 39–51, 1994.

[3] Masaaki Honda, Takeo Igarashi, Hidehiko Tanaka, and Shuichi Sakai. Integrated manipulation: Context-aware manipulation of 2d diagrams. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'99)*. ACM Press, November 1999.

[4] Toshiyuki Masui and Ken Nakayama. Repeat and predict – two keys to efficient text editing. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'94)*, pp. 118–123. Addison-Wesley, April 1994.

[5] David L. Maulsby, Ian H. Witten, and Kenneth A. Kittlitz. Metamouse: Specifying graphical procedures by example. In *Proceedings of SIGGRAPH'89*, Vol. 23, pp. 127–136, Boston, MA, July 1989.

[6] G. Nelson. Juno, a constraint-based graphics system. *Computer Graphics*, Vol. 19, No. 3, pp. 235–243, July 1985.

[7] I. Sutherland. Sketchpad: A man-machine graphical communication system. *IFIPS Proceedings of the Spring Joint Computer Conference*, January 1963.