ベクターグラフィックスとバージョン管理 (1)

プログラミング演習 1#9
平成 26 年 6 月 13 日

1　ベクターグラフィックス

ベクターグラフィックスとは，図や絵を点や曲線のような幾何学的図形の組み合わせで表現したものです．これに対して，画素ごとに色情報を保持する表現方法を用いたものをラスターグラフィックスまたはビットマップと言います．

ベクターグラフィックスを描くためのツール (ドロー系ツール) に，Tgif, OpenOffice.org/LibreOffice Draw, Inkscape などがあります．

1.1　Tgif

`tgif`(図 1) はベクターグラフィックスを描くためのツールの一つです．片山の学生時代から使われて続けている，非常に古いプログラムです．ユーザーインターフェイスに多少癖があって初心者にはとっつきにくい点がありますが，未だに使っている先生も多いようです．

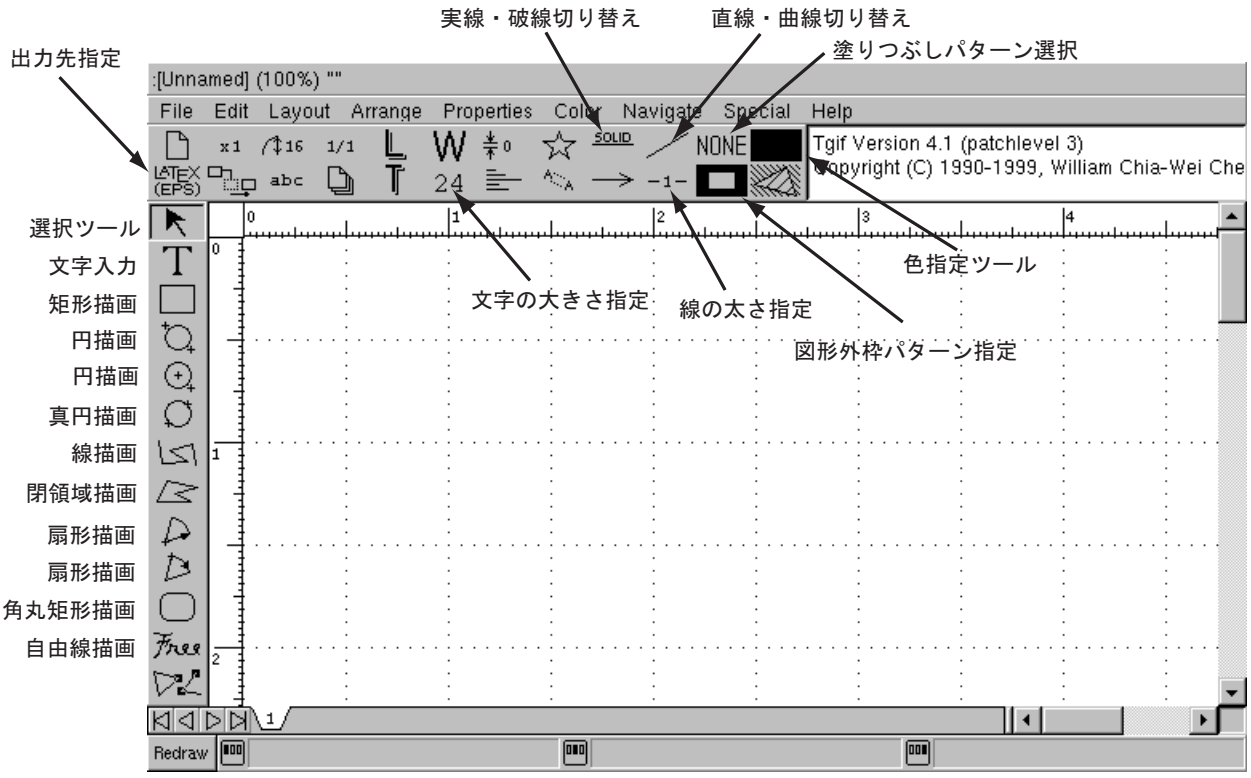

図 1: tgif の画面

1.2 OpenOffice.org/LibreOffice Draw

OpenOffice.org や LibreOffice にも，ドロー系ツールが付属しています．特徴としては，プレゼンテーションツール (Impress) と同様の操作性が提供されているため，ありがちなプレゼンテーションツールに慣れたユーザーにとってとっつきやすい点が挙げられるでしょう．

1.3 Inkscape

Inkscape は新しく高機能なドロー系ツールです．

多彩なプラグインが特徴です．特に，Sozi プラグインを入れることでズームによるプレゼンテーションを作成するツールとして使うことができます．

本日のやってみよう#1
図 2 を Inkscape で描いてみなさい（問題の図は tgif で作成しています）．一番外の枠線は描く必要はありません．ID 横には自分の学籍番号を入れること．

2 バージョン管理とは？

長めだったり複数ファイルにまたがったりするような文章やプログラムを時間をかけて編集していると，時々

- 「直したつもりが元のが正しかった〜！」とか，
- 「文字化けしてたけど気にせず編集していたらもともと何を書いていたのかわかんなくなった〜！」とか，
- 「投稿・提出した時点のバージョンがどんなんだったか分かんなくなった〜！」とか，
- 「冷静に見なおしてみるとどう考えても以前のバージョンのほうがいい感じだった〜！」とか，

などといろいろ後悔して，昔に戻りたくなることがあります．特に，飲み会の後とかに気分が高揚した状態で「俺って勤勉〜！」とか言いながら文書をいじくったり，あるいは締め切り前の徹夜続きの状態でヒイヒイ言いながら書いていると，こういうことが非常に頻繁に起きます．また，このような失敗を経験すると，今度は失敗を恐れるあまり推敲を避けるようになってしまう，ということも起きます．

一人の人間が 1 つの文書を編集している場合にもこういうことが起きますが，多くの人間のチームで文書やプログラムを編集する場合は更に深刻です．たとえば，index された木を意味する trie という専門用語がありますが，この用語を知らない B さんという人がいたとすると，

1. A さんが trie に関してなんか書く．
2. B さんが勝手に「こりゃtree のミスタイプだろ」と解釈して間違って直してしまう．
3. A さんが気づいて[1],「ありゃ，俺間違って tree と書いちゃったっけ？」と思って直す．
4. B さんが「あれ，俺直さなかったっけ」と思って更に直す．

[1] ま，気づかないのが一番恐ろしいわけですが．

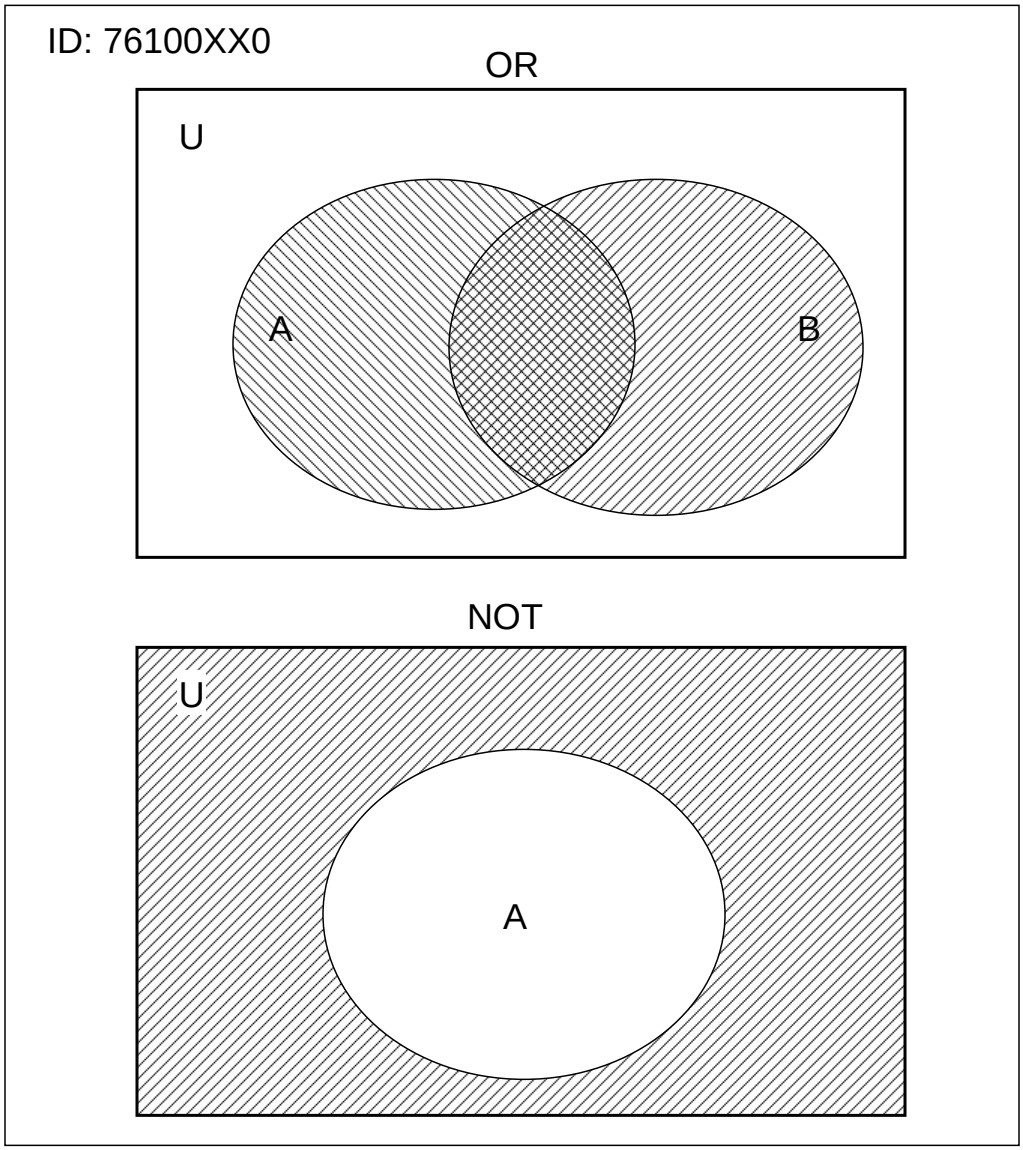

図 2: やってみよう（1）

5. ...

なんて感じでムダな仕事が増えるばかりで，場合によってはいつの間にか殺し合いに発展する可能性も否めません．実際のところは多くの日本人はむしろ遠慮して,「違うんじゃないかなあと思うんだけど勘違いかもしれないし，まあ細かいことだしいいか」となるかもしれませんが，それはそれで先ほどの「失敗を恐れるあまり推敲を避けるようになってしまう」ってのと同様よくないことです．

メールで確認してから直すって手はありますが，細々した修正ごとにやっていると手間がかかる上に遅いです．ではこういうのはどうでしょう．

- 修正の履歴が残る形でさっさと修正してしまう．履歴を見れば誰が何の目的でその修正をしたかわかるし，その修正を元に戻すのも (ツールを使えば) 簡単．
- それでも安易にコロコロ修正されても困る文書であれば，編集する人間が全員手元に (修正履歴を含む) コピーを持っておいて，そっちを修正し，履歴ごとメール送信．時々取りまとめをしている人がそれらを確認して，問題なければ適用していく．(これもツールを使えば簡単．)

このような，文書の変更履歴を管理支援するためのソフトウェアをバージョン管理システムとかリビジョン制御システムとか (version/revision management/control system) 言ったりします．

3 分散型リビジョン制御システム Darcs

OpenOffice.org などの最近のオフィススイートには特化されたバージョン管理の仕組みが入っているようですが，ここではより一般に使える (特に，LaTeX ソースやプログラムファイルなどのテキストファイルの集まりに対する処理が得意な) バージョン管理システムを扱います．現在，このようなバージョン管理システムには free, non-free を含め数多くのものがあります[2]が，この演習では最もアカデミックな香りのする Darcs を扱います．会社に入ると Git あたりを使うことになるかもしれませんが，Darcs が使えれば応用が効くはずです．

Darcs とは，Darcs advanced revision control system (“Darcs という先進的リビジョン制御システム”) の略です．もともと David Roundy という宇宙物理学者が趣味のプログラミングのために開発したもので，以前は David’s advanced revision control system の略だったのですが，“やっぱソフトウェア名に自分の名前を入れるのは恥ずいわ〜”ってことで，現在の再帰的な名前になったみたいです．

Darcs には以下のような特徴があります．

- 分散型である，つまり，CVS や Subversion などの旧来のシステムと違い，中央集権的な保管庫を持つ必要がない．履歴情報はすべて手元のコピーに入っている．(ただし，同一ユーザーが複数のコピー (ブランチ) を同じファイルシステム上に持っている場合は，可能な限りデータを共有してディスク使用量を減らそうとする)
- 個々の修正 (パッチ) の抜き差しが他のどのシステムよりも自由に行える．

4 Darcs の使い方

Darcs はディレクトリツリーごとにバージョンの管理を行います．ディレクトリツリーというのは，あるディレクトリとそのサブディレクトリ（とそのサブサブディレクトリとサブサブサブディレクトリと....）をあわせ

[2] http://ja.wikipedia.org/wiki/バージョン管理システム, http://en.wikipedia.org/wiki/Comparison_of_revision_control_software

たひとかたまりのことです．プログラムや文書のたぐいは，大抵は１つのディレクトリツリー内の複数のソースファイル（コンパイルする前のテキストファイル）から構成され，あるソースファイルを修正した時に別のソースファイルも同時に修正する必要があることが多いため，ディレクトリ (ツリー) ごとにバージョンがあったほうがよいわけです．

ディレクトリツリーを Darcs で管理するには，そのディレクトリ内で

```
darcs init
```

を実行することによって，そのディレクトリの履歴情報などを含む管理情報を保持する_darcs/ というサブディレクトリを Darcs に作らせます．さらに，

```
darcs add ファイル名
```

とすることで，Darcs の管理下に置くファイルを加えておきます．ファイル名は複数指定可能です．また，darcs add -r *とやると，(バックアップファイルなどの自動生成ファイルを除く) すべてのファイルを一気に加えることができます．

そうしておいた上で，ファイルに一連の編集 (パッチ, patch) を加えるたびに，

```
darcs record -a
```

を実行してそれらを 1 つのパッチとして登録します．上記のコマンドを実行すると，パッチの名前を聞かれるので，適当に英語で書いておいて下さい.（てゆーか，最初に実行したときはまずメールアドレスを聞かれると思うので，その場合は適切に入力して下さい.）さらに，より長いコメントを書くか聞かれるので，書きたかったら y を押せばエディタが起動するので書いてセーブして終了すればいいし，書きたくなかったら n を押せばよろし．

-a をつけずに darcs record を実行すると，対話モードに入り，どの変更を登録してどの変更を (今回は) 登録しないかを選択できるようになります．既に入力した変更を複数のパッチに分けて登録したい時は，こちらを使うことになります．登録するなら y キー，登録しないなら n キー，１個前の選択をやり直すなら k キー，それ以外にどういう選択肢があるかとりあえずヘルプを見るなら?キーを押します．

その他，よく使うコマンドを以下に示します．

darcs add　ファイル (やサブディレクトリ) を Darcs の管理下に置く．

darcs remove　ファイルやディレクトリを Darcs の管理下から削除．ファイル自体は削除されない．

darcs whatsnew　未登録の変更としてどういうものがあるかを調べる．

darcs record　一連の変更をパッチとして登録.

darcs unrecord　パッチを登録解除．どのパッチを登録解除するか聞いてくる．ファイルそのものは変更されない.

darcs revert　登録されていない変更を捨てて，登録されていない状態に戻す．-a を付けなければ，変更毎に選択可能．revert しちゃった変更は捨てられてしまい戻せないので，最後に「本当に revert していいんやな」と確認してくる．

darcs unpull　パッチを捨てる．つまり，unrecord と revert の両方を行う．パッチが他所（pull コマンドや push コマンドや put コマンドや get コマンドで作ったコピー）にない限り，捨てられてしまって戻せないので，最後に「本当に unpull していいんやな」と確認してくる．ただし，時々確認してこない時もある気がする．

darcs put コピー先　現在のディレクトリツリーのコピーを他所に作る．バージョンを枝分かれ (branch) させて，実験的なバージョンを作ったり，安定版を作ったり，など，とにかく特別版を作るのに使われる．
コピー先としては，ディレクトリを指定することもできるし，ユーザー名@ホスト名:ディレクトリで指定すると ssh 接続した先に作ることもできる．

darcs get コピー元　他所のディレクトリツリーのコピーを手元に作る．コピー元で実行するか，コピー先で実行するかの違い以外には，`darcs put` との効果上の違いはない（はず）．ただし，`darcs get` の場合コピー元として URL を指定することができ，HTTP 公開されている Darcs レポジトリから取ってくることも可能．

darcs push コピー先　コピー先として既にあるブランチを指定することで，手元にあるパッチのうちコピー先にないものを送り込む．`-a` を指定しなければ，送り込むパッチを対話的に選択できる．
同じ箇所を複数のパッチが同時に編集している場合，衝突 (conflict) といって，デフォルトではそのパッチは push できない．ただし，ムリヤリ push するオプションもある．

darcs pull コピー元　コピー元にあるパッチのうち，手元にないものを取ってくる．コピー元で実行するか，コピー先で実行するかの違い以外には，`darcs push` との効果上の違いはないはずだが，どうも darcs pull の場合デフォルトでムリヤリ pull するオプションが選ばれているみたい．
実は，`darcs get` は `darcs init` して `darcs pull` するのと全く同じ．

darcs annotate ファイル名　ファイルの各行に関して，どのパッチが最後に変更を行ったかを表示する．おかしな変更を見つけた時に，誰がやらかしたのかを特定するのに便利．

darcs changes　パッチの一覧を表示する．

darcs tag 名前　現在あるパッチの集合に名前を付ける．この場合の名前は，大抵はバージョン名だったりする．

なお，パッチの間では依存関係があることがあります．たとえば，文書の最初の方を編集しているパッチ A と最後の方を編集しているパッチ B の間では，重複部分がないため，両方のパッチを含む文書からパッチ A のみ引きぬくこともパッチ B のみ引きぬくことも可能です．しかし，ある箇所を編集しているパッチ X と，同じ箇所を後からさらに編集しているパッチ Y がある場合，パッチ Y はパッチ X に依存しているため，パッチ Y を残してパッチ X を引きぬくことはできません．このような場合は，パッチ X と逆のことをする変更を加えて新しく登録したり，あるいは場合によってはパッチ X とパッチ Y を両方 unrecord して直して record し直したりすることで対処する必要があります．

本日のやってみよう#2

先程作成した（あるいは作成途中の）図にいろいろ落書きして，元に戻せるか試してみよう．

1. `~/Penshu1/figure/` というディレクトリを作成して，このディレクトリにカレントディレクトリを変更．
2. `darcs init` を実行．
3. 作成中の図をこのディレクトリに `ben.svg` という名前で保存．
4. `darcs add ben.svg` を実行．
5. `darcs record -a` を実行．電子メールアドレスは適切に入力して，パッチ名は適当に，たとえば “add ben.svg” とでもしときます．適切に，ってのは，たとえば私の場合だと電子メールアドレスは `Susumu Katayama <skata@cs.miyazaki-u.ac.jp>`と入力しているわけですが，とにかく最低限自分の正しいウェブメールアドレスを書いておいて下さい．
6. 思いに任せて `ben.svg` に色々落書きしてみてください．Undo 機能が使えないように，一旦 Inkscape を再起動するとよいかも．
7. `ben.svg` に上書き保存．Inkscape を終了．
8. やっぱり落書きした内容がいまいちだったと思ったら，`darcs revert` を実行して，先ほどのパッチを revert するか聞いてきた時に y(es) で答えて，本当にいいか聞かれても yes で答えましょう．再度 Inkscape で `ben.svb` を開くと，変更点が元に戻っているはずです．6 に戻る．
9. 逆に，消すのがもったいないほどうまくかければ，`darcs record -a` を実行．パッチ名はさらに適当に，“add rakugaki” とでもしときます．
10. 落書きする前のバージョンが欲しければ，`darcs unpull` を実行して “add rakugaki” のパッチを選択削除すればよいです．が，落書きしたバージョンも残したければ，`darcs put ../raku` (ここで `raku` は適当に決めた名前．) とすることで，`../raku` にブランチ (コピー) を作った上で `darcs unpull` を実行するべし．
11. 余裕があれば，別のブランチで別の落書きをして `darcs record` して，マージしてみてくれたまえ．