SuperIME: IME によるテキスト編集機能の統合

匿名で査読を行うため著者名なし *

概要. 計算機上で様々なテキストエディタが利用されているが、システムごとに編集方法や編集機能が異なっているのが不便である。本論文では、各国語入力のために OS に用意されている IME(Input Method Editor) 機能を利用することにより、あらゆるテキストエディタにおいて同じ操作によるテキスト編集を可能にする方法を提案する。我々の手法を利用すると、異なるテキストエディタ上での編集操作が共通化されるだけでなく、テキスト編集時に便利な様々な機能をあらゆるエディタで利用することが可能になる。

1 はじめに

計算機上でテキストを編集するために様々なテキストエディタが利用されている。文書を作成するときはワープロを利用し、メールを書くにはメールクライアントを利用し、文字端末でのプログラム開発には vim や Emacs を利用し、IDE を利用した開発では付属のエディタを利用し、ネット上でテキストを扱うにはブラウザのテキストフォームを利用するといったように、場合によって異なるエディタが利用されている。

エディタの機能や操作体系はシステムごとに異なっているのが普通である。ブラウザやワープロでテキストを 1 行消したい場合はマウスで行全体を選択してから削除キーを押せばよいが、vim では「d」キーを 2 回タイプして消すことが多く、Emacs では Ctrl-K キーがよく使われている。Emacs に慣れたユーザがワープロ上でも Ctrl-K で行を消去したいと思っても、そういう機能は用意されていないのが普通であるし、機能拡張が可能なシステムを利用している場合でも、操作体系を完全に同じにすることは難しい。あらゆるエディタの操作を統一することは難しいが、様々なエディタで共通に利用できるソフトウェア層をユーザとアプリケーションの間に置くことができれば異なるエディタの編集操作をある程度共通化できる可能性がある。

現在のパソコンには IME(Input Method Editor) と呼ばれる文字入力機構が用意されており、様々な言語のテキスト入力に利用されている。IME はエディタなどとは独立したソフトウェアであり、ユーザのすべてのキー入力を受け取って各国語に変換した結果をアプリケーションに送出する。IME はあらゆるアプリケーションで共通に利用されるので、たとえば日本語入力用の IME を利用すれば、Emacs でもブラウザでも IDE でも同じ操作で日本語を入力できる。IME は一般には各国語入力のみのために利用されているが、テキストの挿入/移動/削除といった編集操作も IME が受け持つようにすれば、様々なエディタ上で同じ操作で編集を行なうことが可能になる。Mac 上の様々なテキストエディタにおいて同じキー操作でテキスト編集を可能にする SuperIME システムについて述べる。

2 実装例

MacRuby で記述された IME である「Gyaim」に様々な編集機能を追加したものを利用した編集作業の例を示す。一般に IME はアルファベット以外の文字を入力するときだけ有効にするのが普通であるが、ここでは Gyaim を 常に有効にしておくことによりあらゆるキー入力を Gyaim が取得している。

2.1 ブロック移動

テキストの一部を別の場所に移動したいことは多いが、エディタによって操作方法が全く異なっている。たとえば Emacs でテキストを移動させたい場合は、移動する領域をキー操作によって指定してから削除/コピーし、カーソルを移動してペーストするという手順を利用するが、ブラウザの編集領域でテキストを移動させたい場合は、マウスで領域を指定した後で領域をドラッグして別の位置に移動するのが基本である。テキスト移動のような基本操作でもエディタごとに操作が異なっているのが不便であるが、Gyaim を利用するとあらゆるエディタで同じ操作でテキストを移動することができる。

図 1 は Mac OS に標準でインストールされている、「テキストエディット」で編集中のテキストである。

* 匿名で査読を行うため所属名なし

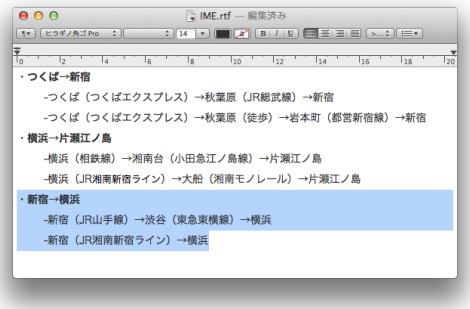

図 1. ブロック移動前の状態.

ここで Shift+↑キーを押すとテキストは図 2 のように変化する。

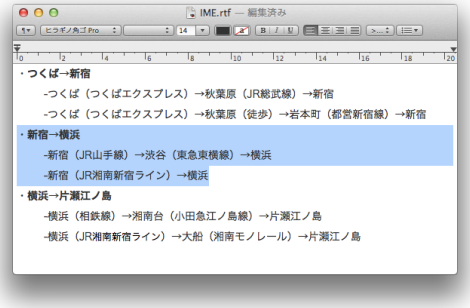

図 2. Shift+↑キーを押した後の状態.

図 3 はブラウザ上で Google Docs のテキストを編集しているところである。ここで Shift+↑キーを二度押すと、テキストは図 4 のように変化する。

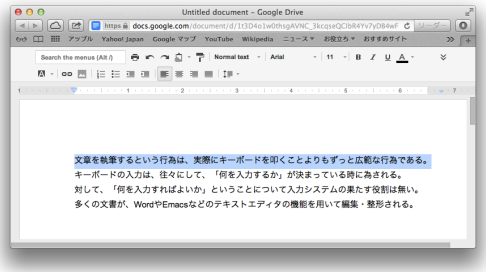

図 3. ブロック移動前の状態.

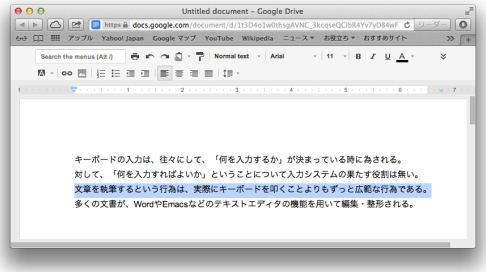

図 4. キーを押した後の状態.

このように、あらゆるエディタにおいて同じ操作でブロック移動を行なうことができることがわかる。

2.2 連続インデント

前述したブロック移動は、インデントを調整する機能も備えている。プログラミングを行なっている時だけでなく、普通のテキストを編集している場合でも行頭の空白やタブの量を調整するインデント処理は頻繁に行われるが、自動的にインデントを行うエディタもあれば、コマンドを打ち込まなければならないものもあり、そのような調整機能を持っていないエディタも多い。Gyaim を利用すると、あらゆるエディタや入力フィールドでブロック移動と同時にインデントが自動で行われる。

図 5 は、Xcode 上で Python プログラムを書いている例である。Xcode は、Objective-C や Ruby などの言語で自動インデントの機能を備えているが、Python には対応していない。たとえば、図 5 のようなテキストに対して図 6 のように移動させたいブロックを選択し、Shift+↓キーを入力することで、テキストは図 7 のように変化する。

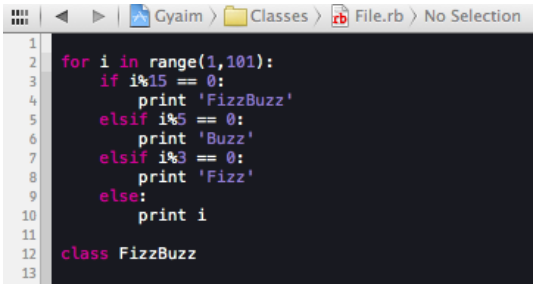

図 5. ブロック移動前の状態

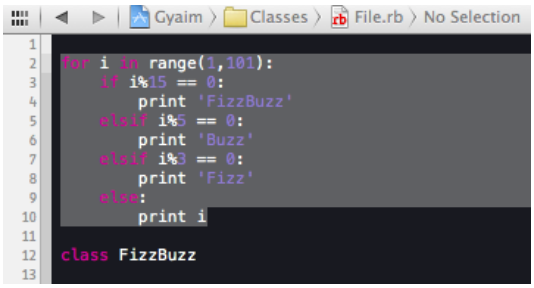

図 6. テキストを選択した状態

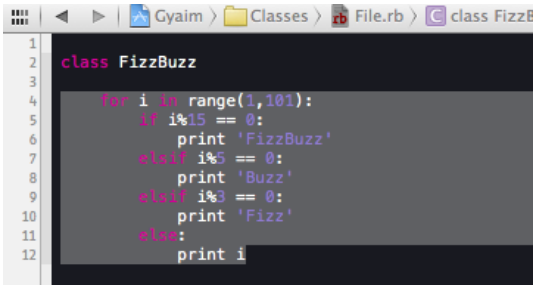

図 7. ブロック移動後の状態

このように、あらゆるエディタにおいて、ブロック移動を行う際に適切な空白とタブを自動で補完することができる。

2.3 単語置換

テキスト編集における単語の検索と置換は、多くのエディタが備えている機能であるが、その操作方法は異なっている。また、Web ブラウザや、DTP ソフトなどでテキストを扱うとき、検索や置換といった機能は用意されていない。ブラウザに入力したテキストを、エディタにコピーして編集操作を行うといった作業、あるいはその逆といった作業を Gyaim により共通化することが出来る。

以下に Gyaim による単語置換の例を示す。図 8 のように入力語と置換語をテキストエリアに打ち込み、任意に設定したファンクションキーを押すと、テキストは図 9 のように置換される。

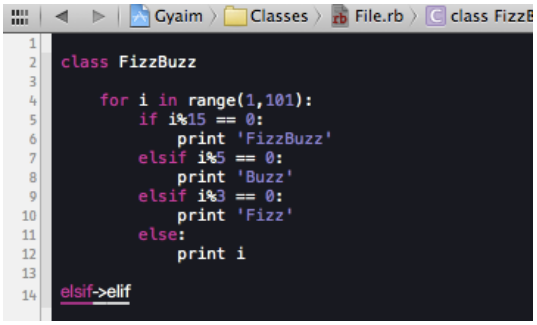

図 8. 単語置換操作の例

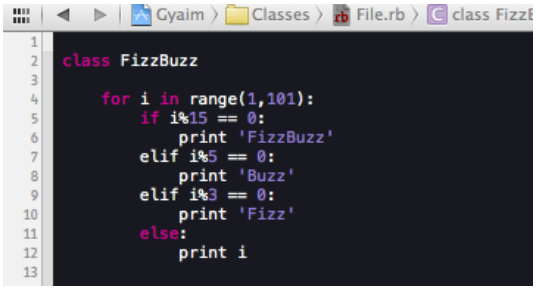

図 9. 単語置換後の状態

このように、あらゆるエディタ上で検索と置換が可能である。

2.4 Emacs Lisp

Emacs においては lisp 拡張を導入することでエディタの機能を拡張することが出来るが、これらのスクリプトを IME 上に実装することで、他のエディタ上でもこれらの有用な拡張機能を動作させることができる。

以下に Gyaim による [?] の例を挙げる。Dynamic Macro は、入力の繰り返しを自動化する Emacs 拡張である。図 10 のように、ユーザの入力操作が繰り返しになっているとき、任意に設定したファンクションキーを押すと、テキストは図 11 のように編集される。

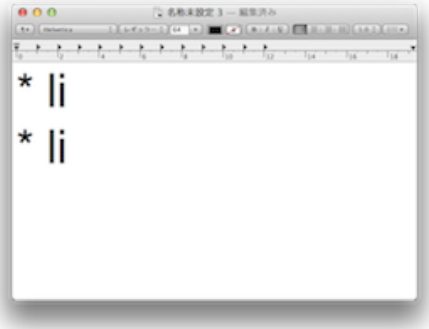

図 10. Dynamic Macro 機能を使用する前のテキスト

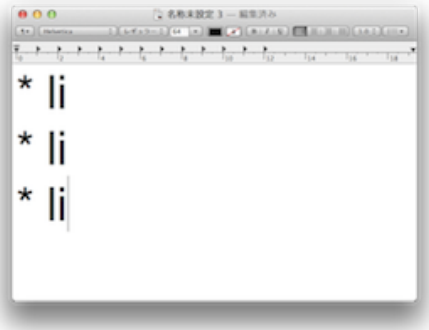

図 11. Dynamic Macro 機能を使用した後のテキスト

以下は Gyaim の Dynamic Macro 機能をブラウザのアドレスバー上で実行した例である。図 12 のように、ユーザの入力に”abc”が繰り返されているとき、任意に設定したファンクションキーを押すと、テキストは図 13 のように編集される。

図 12. Dynamic Macro 機能を使用する前のテキスト

図 13. Dynamic Macro 機能を使用した後のテキスト

3 実装

Gyaim は MacRuby で記述された Mac 用の IME であり、ソースが 500 行程度とコンパクトであるにもかかわらず、他の IME に見られない機能を実装しており、本論文のような実験も容易である。

3.1 MacRuby

MacRuby は、Mac 用のアプリケーションを開発するために拡張された Ruby 実行環境であり、Mac の Objective-C ライブラリを Ruby で扱うことが出来る。Gyaim では、日本語などの入力を補助するフレームワークである InputMethodKit Framework を MacRuby から呼び出すことによって基本的な IME の機能を実装している。

しかし、ブロック移動やインデントの処理をあらゆる テキストエリアで行うためには、テキストフィールドに 入力されている全文を IME が取得する必要があり、InputMethodKit はこの機能を備えていない。Gyaim では、AppleScript などを併用することによりこの問題を解決したが、実装には課題が残っている。

4 本手法の限界

IME はアプリケーションと独立に実装されているため、現在のような実装ではアプリケーションの内部状態によって動作を変えたり、アプリケーションの振る舞いを制御したりすることはできず、表に出ているテキストの編集操作しかできない。 既存のシステム自体は変更せずに、皮をかぶせる形で機能を拡張する手法はある程度有用ではあるが、問題の根本的な解決が必要な場合には限界がある。テキスト編集の場合は根本的に解決しなければならない問題は多くないので、本論文の手法はとりあえず有効だといえるが、根本的な解決のためには、テキスト入力の枠組みである IME に加えて、各コンピュータがテキスト編集のための枠組みを用意する必要があるだろう。

謝辞

謝辞は、ブラインドレビューのため、投稿時には削除すること．カメラレディ時に、必要があれば追加すること．

Dynamic Macro system[?] [?] [?] [?] [?] [?] [?] [?][?][?] [?][?][?][?] [?]. [?][?].

参考文献

[1] WISS ホームページ. http://www.wiss.org/.

[2] H. Aoki, B. Schiele, and A. Pentland. Realtime Personal Positioning System for Wearable Computers. In *Proceedings of the 3rd IEEE International Symposium on Wearable Computers*, pp. 37–43, 1999.

[3] 暦本 純一. まえがき：WISS2000 について. インタラクティブシステムとソフトウェア VIII, pp. i–ii. 近代科学社, 2000.

未来ビジョン

パソコンやスマートフォンで利用されている様々なテキスト入力システムは変換方式も使い勝手も全く異なっているのが普通になっているが、単純で柔軟な入力方式を利用すると、パソコンでもスマートフォンでもほぼ共通の入力を行なうことが可能である。Gyaim はこのような思想にもとづいて作成された IME であるが、本論文のような手法を取り入れることにより、あらゆる機器において入力も編集もユニバーサルにすることが可能であろう。

また、本論文ではテキストエディタに絞った説明を行なったが、IME はユーザのすべてのキー入力を直接受け取る窓口になっているため、編集と関係無いキー操作も IME に担当させることによってより幅広い計算機操作を実行することができる。たとえばシステム音量や画面の明るさをコントロールするにはシステムに用意された特別のキーを使ったり、システムに用意されたショートカットを利用したりすることが多いが、このようなものも IME から制御するようにしておけばシステム全体のショートカット設定などは不要になる。