インタラクティブ タートルグラフィクス

01ca0125 鈴木 藍
2002 年 8 月 27 日

目 次

概 要

作成したインタラクティブタートルグラフィクスの仕様と実行例をまとめた。独自の追加点についてもいくつか挙げてある。

レポートの目的

アーキテクチャに沿っての設計やそれを意識した実装を心がけて作成してみたプログラムを振り返る。

1 全体の仕様

- マルチタートルをサポート
- gnuplot の使用できる色を全色使用可能
- スクリプトファイルを ロード、セーブが可能
- タートルの状態を検査できる inspector を実装

2 命令一覧

Gturtle では、コマンドを大きく二つの種類に分類している。

2.1 システムコマンド

turtle を生成、削除したり タートル達を検査するといった 彼らへの間接的な通信やユーザからタートル達への命令を補助するようなコマンド郡。

2.1.1 コマンドの記述形式

システムへのコマンドは以下のフォーマットで行う。

$$< \text{コマンド名} > \quad < \text{第一引数} > \quad < \text{第二引数} > \ldots.$$

2.2 タートルへの命令

生成したタートル達へ通信するための手段。

2.2.1 命令の記述形式

タートルへの命令は以下のフォーマットで行う。

$$< \textit{turtle} \text{の名前} > \quad < \text{命令} > \quad < \text{第一引数} > \ldots.$$

2.3 システムコマンド一覧

コマンド名と引数	説明	制約
init <name>	<name> という名の turtle を 1 匹生成します	同じ名前の turtle は 作れません
delete <name>	<name> という名の turtle を 削除します	全ての turtle が いなくなった時、画面は クリアされます
inspect <name>	<name> という名の turtle を 検査します	
inspectall	すべての turtle 検査します	
history <name>	<name> という名の turtle の 命令実行履歴を表示します	
load <filename>	<filename> という名の スクリプト ファイルをロードします	拡張子 *.gt 以外は読み込みません ファイル形式については '4. ファイル形式' を参照して ください
save <name> <filename>	<filename> という名の スクリプト ファイルに <name> という名の turtle を保存します	同じ名のファイル名が存在する ならば それを消して上書きします
saveall <filename>	<filename> という名の スクリプト ファイルに すべての turtle を 保存します	同じ名のファイル名が 存在する場合は保存が 出来ません 別名で保存してください
ls	カレントディレクトリのファイル 一覧を表示します	
bye (or 'q')	プログラムを終了します	全ての記憶は消去されます
list	存在する turtle の一覧を表示します	
colorlist	使用できるペンの色を表示します	全 8 種類です
help	ヘルプを表示します	

2.4 タートルへの命令一覧

命令と引数	説明	制約
<name> pd	<name> という名の turtle のペンを おろして線を描けるようにします	
<name> pu	<name> という名の turtle のペンを あげて線が描けないようにします	
<name> home	<name> という名の turtle を フィールドの中心に戻します	
<name> fd <length>	<name> という名の turtle を <length> 分前に歩かせます	
<name> bk <length>	<name> という名の turtle を <length> 分後退させます	
<name> rt <theta>	<name> という名の turtle を <theta> 分右に向かせます	
<name> lt <theta>	<name> という名の turtle を <theta> 分左に向かせます	
<name> color <colorname>	<name> という名の turtle のペンの 色を <colorname> に変更させます	色は colorlist にあるもののみです
<name> undo	<name> という名の turtle の一つ 前の命令を取り消させます	

3　プログラムの仕様

3.1　アーキテクチャ

今回は、MVC を意識して実装した。モデル、ビュー、コントローラにそれぞれ役割を分担するような形にした。

3.2　主な変数

3.2.1　Turtle

モデルであるタートルの状態を表す構造体。ソースの冒頭で Turtle で typedef 宣言されている。現在の位置、向いている方向、ペンの上げ下げ、ペンの色、現在までの命令の数、コマンドとその引数の履歴、前と後ろのタートルのポインタを情報として持つ。117 行目

```
/*--------- Instance Variable. ----------------*/
struct _turtle {
    double          x, y, angle;
    int             pen;
    int             color;
    int             commandNumber;
    char            name[NAME_SIZE];
    char            command[BUF_SIZE][BUF_SIZE];
    char            parameter[BUF_SIZE][BUF_SIZE];
    struct _turtle *prev, *next;
};
typedef struct _turtle Turtle;
```

3.2.2　リストを管理する変数

タートルは リストで管理されている。リストの先頭、末尾、現在命令を与えられているタートルを表す変数を 3 つ使用している。

513 行目

```
/*----------- Class Variable. -------------------*/
Turtle     *frontPointer  = NULL;
Turtle     *rearPointer   = NULL;
Turtle     *activeTurtle  = NULL;
/* ^ClassVariable */
/*-----------------------------------------------*/
```

変数一覧

型	変数名	説明
double	x, y	現在の座標
double	angle	タートルが向いている方向
int	pen	ペンの上げ下げの状態。 0 なら上げていて、1 なら下げている
int	color	ペンの色
int	commandNumber	現在までの命令の数
char	name[]	名前
char	command[][]	命令
char	parameter[][]	命令の引数
Turtle *	prev	前のタートルへのポインタ
Turtle *	next	後ろのタートルへのポインタ
Turtle *	frontPointer	リストの先頭のタートルのポインタ
Turtle *	rearPointer	リストの末尾のタートルへのポインタ
Turtle *	activeTurtle	最後に命令をあたえられたタートルへのポインタ

3.3 関数の階層構造

別紙に添付

3.4 関数一覧

3.4.1 モデル

モデルであるタートルを操作する関数の一覧。

型	関数名	引数	説明
double	locationX	Turtle *tp	引数のタートルの x 座標を返す
double	locationY	Turtle *tp	引数のタートルの y 座標を返す
void	setX	Turtle *tp , double x	引数のタートルの x 座標 に 引数の x を設定する
void	setY	Turtle *tp , double y	引数のタートルの y 座標 に 引数の y を設定する
void	setAngle	Turtle *tp , double th	引数のタートルの angle に 引数の th を設定する
double	yourAngle	Turtle *tp	引数のタートルの angle を返す
void	setColor	Turtle *tp	引数のタートルの angle を返す
char *	penColor	Turtle *tp	引数のタートルの color を返す
void	setName	Turtle *tp, char *name	引数のタートルの name に引数の name を設定する
char	self	Turtle *tp	引数のタートルの name を返す
void	history	Turtle *tp	引数のタートルの命令の履歴を全て表示する
int	isDown	Turtle *tp	引数のタートルの pen が 1 なら 1 を返す
int	isUp	Turtle *tp	引数のタートルの pen が 0 なら 1 を返す
void	turnRight	Turtle *tp, double th	引数のタートルの向きを th 分右に向ける
void	turnLeft	Turtle *tp, double th	引数のタートルの向きを th 分左に向ける
void	Forward	Turtle *tp, double len	引数のタートルを len 分 前方に進ませる
void	Back	Turtle *tp, double len	引数のタートルを len 分 後方に進ませる
void	home	Turtle *tp	引数のタートルを座標 x=0, y=0 の位置に移動する
void	penDown	Turtle *tp	引数のタートルの pen に 1 を設定する
void	penUp	Turtle *tp	引数のタートルの pen に 0 を設定する
void	undo	Turtle *tp	引数のタートルの一つ前の命令を取り消す
void	Do	Turtle *tp, int index	引数のタートルの index 番目の命令を実行する
void	line	int color, double x1, y1, x2, y2	color の色で線を描く
int	isExistColor	char *colorname	引数の colorname という色が存在すれば 1 を返す
void	Replot	void	全てのタートルは命令をはじめからやり直す

3.4.2 ビュー

命令の結果を gnuplot に反映させるための関数の一覧。

型	関数名	引数	説明
void	Draw	void	最後に命令を与えられたタートルの命令を実行する
void	WindowClear	void	gnuplot 上に描かれた線を全て消す

3.4.3 コントローラ

今回は、主にシステムにあたえられるコマンドを処理するという役割にした。

型	関数名	引数	説明
int	isEmpty	void	タートルが一つも存在しなければ 1 を返す
int	isSame	Turtle *tp, char *name	tp の name と 引数の name が同じなら 1 を返す
Turtle *	addTurtle	Turtle *tp	tp を リストの末尾に追加する
void	deleteTurtle	Turtle *tp	tp を リスト から削除する
void	ls	void	カレントディレクトリを表示する
void	help	void	ヘルプを表示する
void	bye	void	Gturtle を終了する
void	init	char *name	name というタートルを生成する
void	delete	char *name	name というタートルを削除する
void	showColor	void	カラーリストを表示する
void	inspectIt	char *name	name というタートルを検査する
void	inspectAll	void	全てのタートルを検査する
int	getCommand	void	システムコマンドを受け取れば実行して 0 を返す
Turtle *	searchTurtle	char *name	name というタートルのポインタを返す
void	clearData	Turtle	tp の現在の状態を初期化する
void	SystemInit	void	gnuplot の画面の大きさなどを設定する
void	TurtleList	Turtle	存在するタートルの名前を全て表示する
void	Save	char *name,*filename, *mode	タートルを filename に保存する
void	SaveAll	char *filename	全てのタートルを filename に保存する
void	Load	char *filename	filename からデータをロードする
int	fileCheck	char *filename	filename 読み込み可能かチェックする

4 追加した機能

いくつか、機能を追加した。

- マルチタートル
- タートルの命令の履歴の表示
- ペンの色の変更が可能
- タートルの検査が可能
- ディレクトリの表示

gnuplot が使用可能な色を全て使えるようにした。又、タートルの検査は、タートルの状態を知るために用いる。

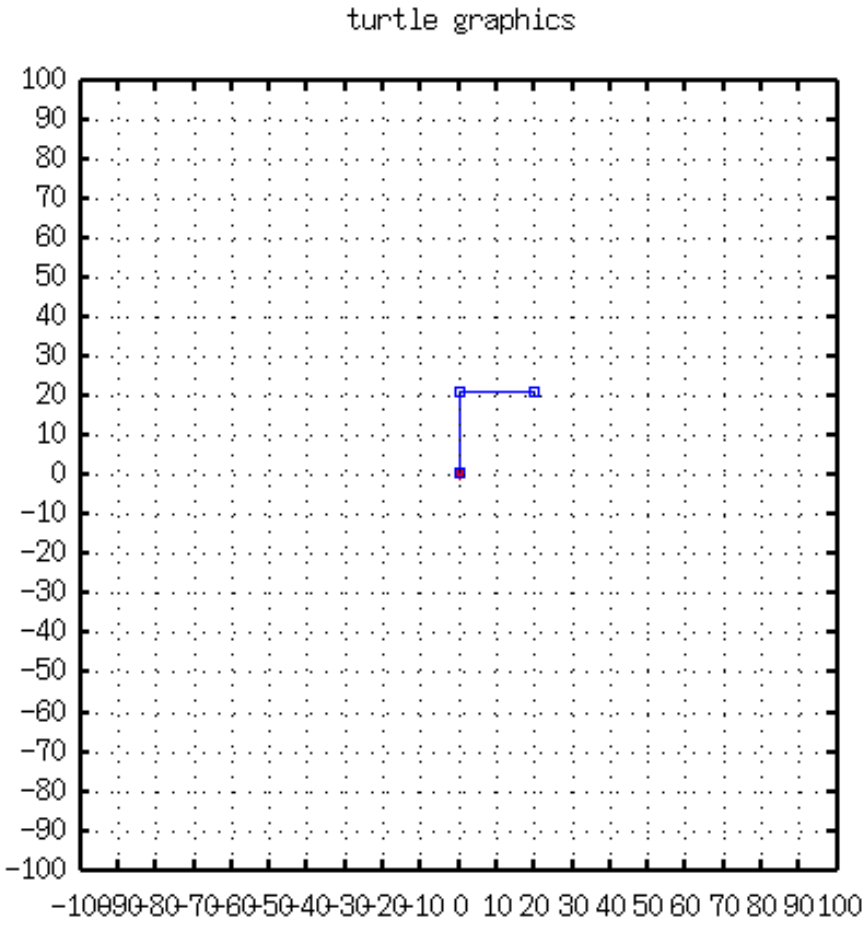
5 実行例

5.1 対話的に実行する

以下は、tt というタートルを生成して、ペンを下げ、ペンの色を青に変えた後に 前に 20 進み、右に 90 曲がり、前に 20 進むという命令を実行した例である。そのあとに tt を検査した結果を表示している。

```
[:spiral|^bin] ./gturtle
*** WELCOME TO GTURTLE! ***
    Author: ai suzuki
    Aug. 13 2002
    See <help> to usage
gturtle> init tt
gturtle> tt pd
gturtle> tt color blue
gturtle> tt fd 20
gturtle> tt rt 90
gturtle> tt fd 20
gturtle> inspect tt
--<Turtle Inspector>--
name     : tt
now      : x = 20.000000, y = 21.000000
distance : 90.000000
pen      : down
pen color: blue
command  : 8
gturtle> q
BYE.
```

5.2 実行結果 1



5.3 スクリプトを実行する

以下は、サンプルのスクリプトをロードした結果である。スクリプトファイル star.gt の内容は

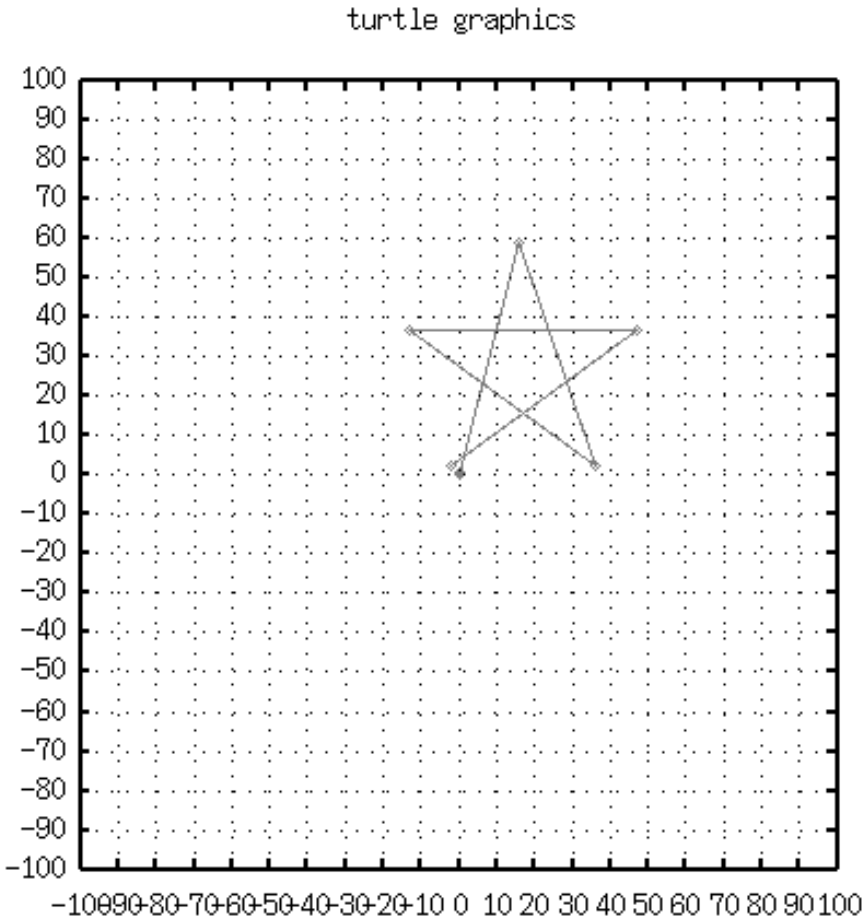
```
(hoge
    color orange
    pd
    rt 15
    fd 60
    rt 145
    fd 60
    rt 145
    fd 60
    rt 145
    fd 60
```

```
    rt 145
    fd 60
    rt 145
)
```

である。

```
gturtle> load ../examples/star.gt
```

5.4 実行結果 2



成果・反省

MVC を意識して実装した事はとても勉強になったと思う。その他、実装する際には 小さなプログラムのテストを重ねて作り上げて行くという方法で行ったが、バグが少なく順調に作れたと思

う。しかし、設計についてはそれ程慎重に行わなかったので、今度は設計に力をいれたいと思った。

感想

楽しかったです。

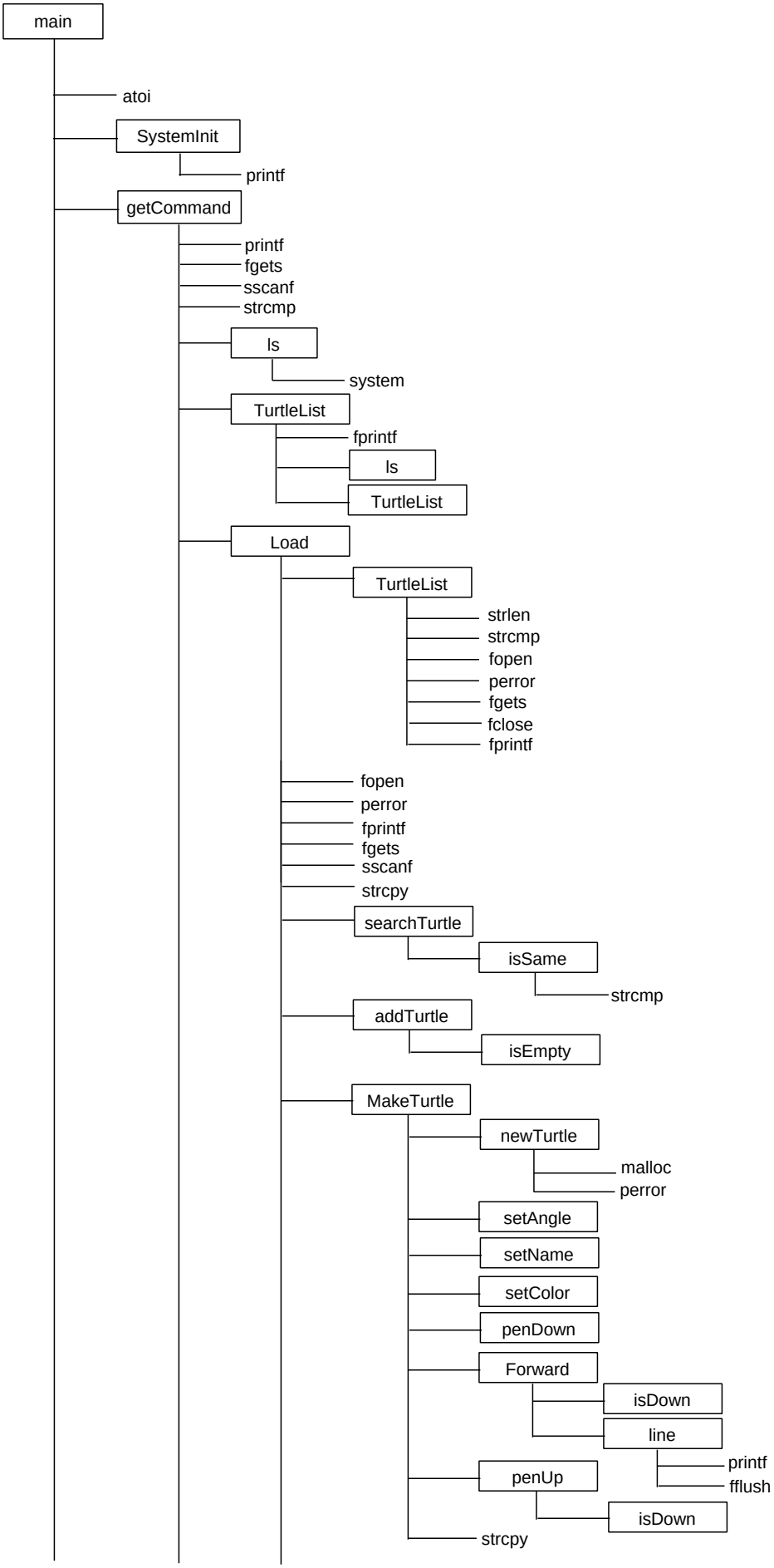
参考文献

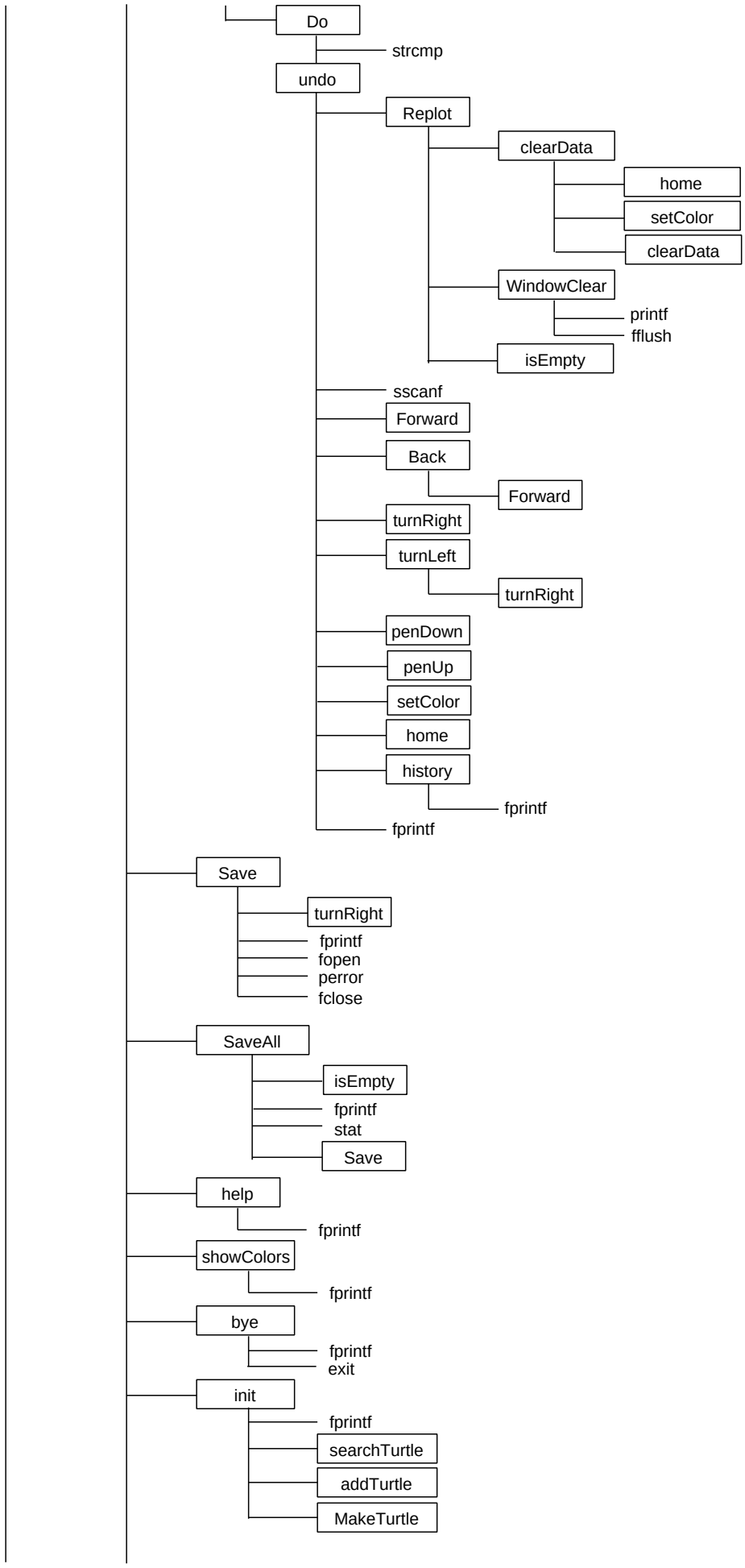
[1] プログラミング言語 C
カーニハン , リッチー 著
2000 年 4 月 15 日 (第 2 版)

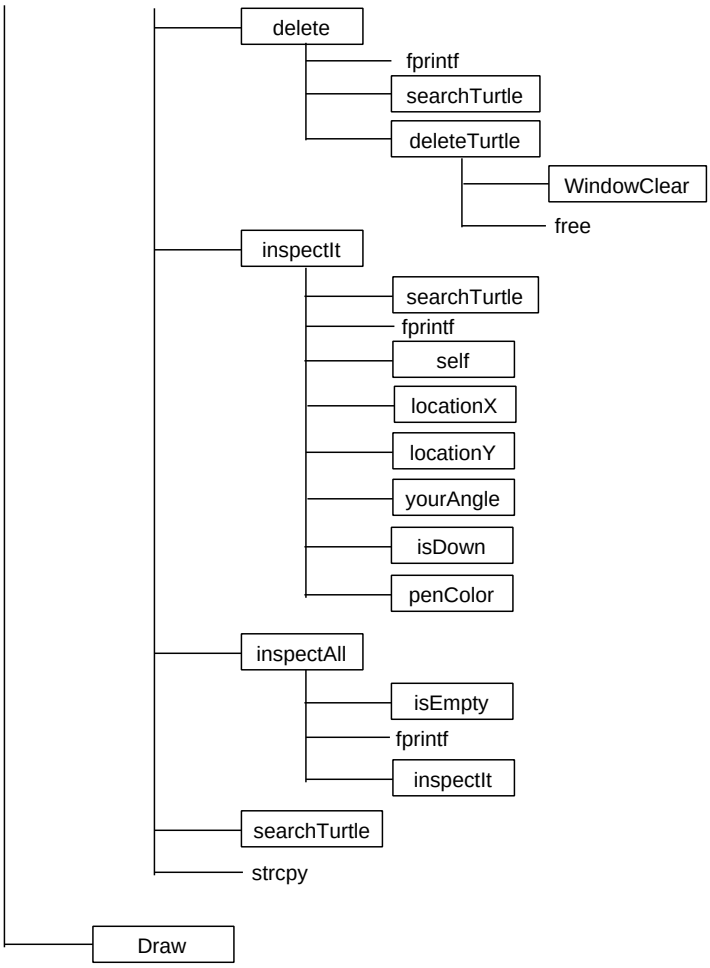
[2] Smalltalk イディオム
青木 淳 著
1997 年 2 月 25 日 (第 1 版)

[3] Happy Squeaking!
http://www.ogis-ri.co.jp/otc/hiroba/technical/Squeak/index.html
2002 年 8 月 5 日 (第 1 版)

関数の階層構造図


main
atoi
SystemInit
printf
getCommand
printf
fgets
sscanf
strcmp
ls
system
TurtleList
fprintf
ls
TurtleList
Load
TurtleList
strlen
strcmp
fopen
perror
fgets
fclose
fprintf
fopen
perror
fprintf
fgets
sscanf
strcpy
searchTurtle
isSame
strcmp
addTurtle
isEmpty
MakeTurtle
newTurtle
malloc
perror
setAngle
setName
setColor
penDown
Forward
isDown
line
printf
fflush
penUp
isDown
strcpy


Do
strcmp
undo
Replot
clearData
home
setColor
clearData
WindowClear
printf
fflush
isEmpty
sscanf
Forward
Back
Forward
turnRight
turnLeft
turnRight
penDown
penUp
setColor
home
history
fprintf
fprintf
Save
turnRight
fprintf
fopen
perror
fclose
SaveAll
isEmpty
fprintf
stat
Save
help
fprintf
showColors
fprintf
bye
fprintf
exit
init
fprintf
searchTurtle
addTurtle
MakeTurtle


delete
fprintf
searchTurtle
deleteTurtle
WindowClear
free
inspectIt
searchTurtle
fprintf
self
locationX
locationY
yourAngle
isDown
penColor
inspectAll
isEmpty
fprintf
inspectIt
searchTurtle
strcpy
Draw