# １　はじめに

コンピュータはＧＵＩの導入によって使いやすくなり、広く浸透している。しかし、現在のプログラミング環境は、初心者には取っ掛かり難いものである。科学計算や事務処理用のプログラミング言語を学んでも使う機会がない。また、エディタを使って意味不明なコードを間違えずに打たなければならない。これでは興味がわかず、プログラミングを理解する迄に飽きてしまう。

そこで本研究では、ジグソーパズルのようにプログラミングし音や絵を動かすことのできるプログラミング環境を構築する。

まずここで本環境の目的について述べた。次に２章では環境の概要、３章では環境の実現、４章では使用法について述べる。

# ２　プログラミング環境の概要

本プログラミング環境はアセンブラを模した、図形を用いてプログラミングする環境である。

## ２．１　プログラミング環境の仕様

プログラミング環境の仕様を以下に示す。

| | |
|---|---|
| 配置できるピースの数 | 縦５０個、横５個、全２５０個 |
| レジスタ：Ｒ | Ｒ０～Ｒ１２７の１２８個、数値の格納に用いる。<br>小数は使えない |
| スタックポインタ：ＳＰ | スタックの最上段のアドレスを保持する |
| スタック | ６４個までアドレスを保持する |
| プログラムレジスタ：ＰＲ | 次に実行する命令のアドレスを保持する |
| ピース | 命令用４５種類、編集用３種類 |
| 保存形式 | テキスト形式（拡張子.gao） |

## ２．２　プログラミング環境の特徴

本プログラミング環境の特徴を以下に示す。

＊ジグソーパズルのように図形を配置することでプログラミングできる。
　コードを記述するかわりにピースを並べてプログラミングすることにより、スペルミスを防げ、またプログラムの流れを目で確認しながらプログラミングできる。

＊ピースを配置するときに、命令に応じて必要な入力が要求される。
　数を入力するのか文字を入力するのか説明が表示される。
　数は１６進表記でも入力できる。

＊主な操作はマウスとテンキーだけでできる。
　アドレスなど文字列を入力するとき以外はキーボードを使わずに操作できる

＊実行は視聴モードとトレースモードの二つから選択できる。
　以下に各モードの説明を記す
　・視聴モード
　　実行画面が表示され、音が鳴ったり画像が表示される。
　　横にレジスタの値が表示される。
　　プログラムが思うとおりに動いているか見るのに向いる。
　・トレースモード
　　音や画像を扱う命令は無視される。
　　マウスをクリックするごとに命令が一つずつ実行され、実行中のピースの色が反転する。

エラーが発生した場合にプログラムの流れを確認して、エラーの発生位置と原因を特定するのに向いている。

## ２．３　言語

ピース一つ一つに割り振られている命令はアセンブラ言語に似ているが、実行環境が違い、また音や画像を扱う命令があるため、互換性は無い。アルゴリズムの学習専用の言語である。

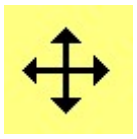
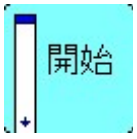
言語の構成

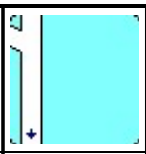
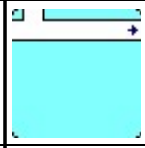
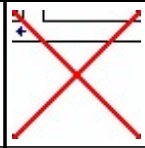
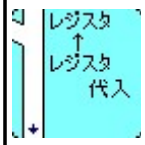
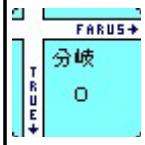
・流れ制御：開始・終了位置の指定、条件分岐、FOR 文・WHILE 文的な繰り返し
・演算：加算、減算、乗算、除算、余り
・音：MIDI ファイルの再生、単音出力、音色変更
・画像：ビットマップファイルの表示、四角形・円の描画
・入力：マウスのボタン・座標の状態を認識

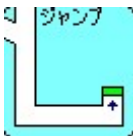
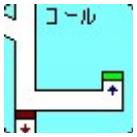
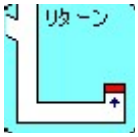
命令は全部で４５種類あり、この構成に従って、命令の代わりとなるピースは色分けられている。

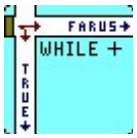
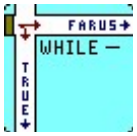
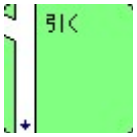
## ２．４　編集ピースと命令ピースの仕様

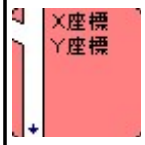
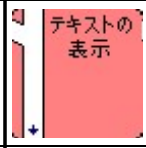
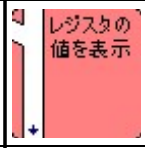
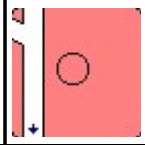
ピースの説明を下の表に示す。番号に＊マークのあるものは編集ピースを示す。

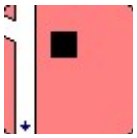
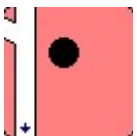
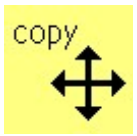
| 番号 | 絵 | 命令（表記） | 入力 | 説明 |
|---|---|---|---|---|
| ０＊ |  | 編集<br>Edit | | クリックしたピースの入力内容の表示と変更。 |
| １＊ |  | 移動<br>Move | | ピースをドラッグ＆ドロップすることでピースを移動する。 |
| 2 | | 空<br>Null（NUL） | 無し | |
| 3 |  | 開始<br>Start（STR） | n | 開始位置の指定と命令を実行する間隔（ｎミリ秒）の指定 |
| 4 |  | 終了<br>End（END） | 無し | 終了位置の指定 |

| | | | | |
|---|---|---|---|---|
| 5 |  | 下へ<br>Down（DON） | 無し | 下に移る<br>ＰＲ　←　ＰＲ＋１ |
| 6 |  | 右へ<br>Right（RIT） | 無し | 右の列に移る<br>ＰＲ　←　ＰＲ＋５０ |
| 7 |  | ~~左へ~~<br>~~Left（LET）~~ | ~~無し~~ | ~~左の列に移る~~ |
| 8 |  | ロード　ナンバー<br>Load number（LDn） | ｒ，ｎ | レジスタｒに数値ｎをはめる<br>Ｒ（ｒ１）　←　ｎ |
| 9 |  | ロード　レジスタ<br>Load register（LDr） | ｒ１，ｒ２ | レジスタｒ１にレジスタｒ２の値をはめる<br>Ｒ（ｒ１）　←　Ｒ（ｒ２） |
| １０ |  | 零分岐<br>If zero（IFz） | ｒ | レジスタｒの値が０なら下に、でなければ右の列に移る<br>Ｒ（ｒ）＝０：ＰＲ←ＰＲ＋１<br>それ以外　　：ＰＲ←ＰＲ＋５０ |
| １１ |  | 非零分岐<br>If nonzero（IFn） | ｒ | レジスタｒの値が０以外なら下に、でなければ右の列に移る<br>Ｒ（ｒ）！＝０：ＰＲ←ＰＲ＋１<br>それ以外　　：ＰＲ←ＰＲ＋５０ |
| １２ |  | 正分岐<br>If plus（IFp） | ｒ | レジスタｒの値が正の数なら下に、でなければ右の列に移る<br>Ｒ（ｒ）＞０：ＰＲ←ＰＲ＋１<br>それ以外　　：ＰＲ←ＰＲ＋５０ |
| １３ |  | 負分岐<br>If minus（IFm） | ｒ | レジスタｒの値が負の数なら下に、でなければ右の列に移る<br>Ｒ（ｒ）＜０：ＰＲ←ＰＲ＋１<br>それ以外　　：ＰＲ←ＰＲ＋５０ |

| | | | | |
|---|---|---|---|---|
| １４ |  | 繰り返し開始点<br>For（FOR） | r | レジスタｒの値の数だけ繰り返す。<br>For end とセットで使う。<br>ｓｔ（　）<br>ＰＲ←ＰＲ＋１ |
| １５ |  | 繰り返し終了点<br>For end（FRe） | 無し | レジスタの値が０なら下へ移る、そうでなければレジスタの値を１引いて For に戻る。For とセットで使う。<br>Ｒ（ｒ）＝０：ＰＲ←ＰＲ＋１<br>それ以外　　：Ｒ　←Ｒ－１<br>ＰＲ←ｐｏｐ（　） |
| １６ |  | ポイント<br>Point（PNT） | ｎ | JMP、CALL 時に使うポイント番号の指定 |
| １７ |  | ジャンプ<br>Jump（JMP） | ｎ | ポイントｎに飛ぶ。<br>ＰＲ　←　ａｄｄ（Ｐｏｉｎｔ　ｎ） |
| １８ |  | コール<br>Call（CAL） | ｎ | アドレスをスタックしポイントｎに飛ぶ（RETURN で戻れる）。<br>ｓｔ（　）<br>ＰＲ　←　ａｄｄ（Ｐｏｉｎｔ　ｎ） |
| １９ |  | リターン<br>Return（RET） | 無し | CALL 位置の次に戻る<br>ＰＲ　←　ｐｏｐ（　）＋１ |
| ２０ |  | 零間繰り返し<br>While zero（WHz） | ｒ | レジスタｒが０の間繰り返す、０以外になったら右の列移る。While　end とセットで使う。<br>Ｒ（ｒ）＝０：ｓｔ（　）<br>ＰＲ←ＰＲ＋１<br>それ以外　　：ＰＲ←ＰＲ＋５０ |

| | | | | |
|---|---|---|---|---|
| ２１ |  | 非零間繰り返し<br>While nonzero（WHn） | r | レジスタｒが０以外の間繰り返す、０になったら右の列に移る。While　end とセットで使う。<br>Ｒ（ｒ）！＝０：ｓｔ（　）<br>ＰＲ←ＰＲ＋１<br>それ以外　　：ＰＲ←ＰＲ＋５０ |
| ２２ |  | 正間繰り返し<br>While plus（WHp） | r | レジスタｒが正の数の間繰り返す、正の数以外になったら右の列に移る。While end とセットで使う。<br>Ｒ（ｒ）＞０：ｓｔ（　）<br>ＰＲ←ＰＲ＋１<br>それ以外　　：ＰＲ←ＰＲ＋５０ |
| ２３ |  | 負間繰り返し<br>While minus（WHm） | r | レジスタｒが負の数の間繰り返す、負の数以外なったら右の列に移る。While end とセットで使う。<br>Ｒ（ｒ）＜０：ｓｔ（　）<br>ＰＲ←ＰＲ＋１<br>それ以外　　：ＰＲ←ＰＲ＋５０ |
| ２４ |  | 間繰り返し終了点<br>While end（WHe） | 無し | 直前の WHILE の位置に移る<br>ＰＲ　←　ｐｏｐ（　） |
| ２５ |  | 加算<br>Plus（PLS） | ｒ１，ｒ２ | Ｒ（ｒ１）←Ｒ（ｒ１）＋Ｒ（ｒ２） |
| ２６ |  | 減算<br>Minus（MIS） | ｒ１，ｒ２ | Ｒ（ｒ１）←Ｒ（ｒ１）－Ｒ（ｒ２） |
| ２７ |  | 乗算<br>Multiply（MUT） | ｒ１，ｒ２ | Ｒ（ｒ１）←Ｒ（ｒ１）×Ｒ（ｒ２） |
| ２８ |  | 除算<br>Divide（DIV） | ｒ１，ｒ２ | Ｒ（ｒ１）←Ｒ（ｒ１）÷Ｒ（ｒ２） |
| ２９ |  | 余り<br>Remainder（REM） | ｒ１，ｒ２ | Ｒ（ｒ１）←Ｒ（ｒ１）％Ｒ（ｒ２） |

| | | | | |
|---|---|---|---|---|
| ３０ |  | 表示位置<br>Position（POS） | ｒ１，ｒ２ | 画を表示する座標Ｐを指定<br>Ｘ座標＝Ｒ（ｒ１）<br>Ｙ座標＝Ｒ（ｒ２） |
| ３１ |  | 表示色<br>Color（COL） | ｒ | レジスタｒの値を描画色に指定<br>色＝赤色の強さ（０～２５５）<br>＋緑色の強さ×２５６<br>＋青色の強さ×２５６ |
| ３２ |  | 表示線サイズ<br>Size（SIZ） | ｒ | レジスタｒの値を線の太さに指定 |
| ３３ |  | 点描画<br>dot（DOT） | | 座標Ｐに点を打つ |
| ３４ |  | 文字表示<br>Text（TXT） | Ｓ | テキストＳの表示 |
| ３５ |  | 値表示<br>Text register（TXR） | ｒ | レジスタｒの値を表示 |
| ３６ |  | 線描画<br>Line（LIN） | ｒ１，ｒ２ | 座標Ｐから座標（Ｒ（ｒ１）、Ｒ（ｒ２））へ線を引く |
| ３７ |  | 円描画<br>Circle（CIR） | ｒ１，ｒ２ | 座標Ｐを中心に楕円を書く<br>高さ＝Ｒ（ｒ１）　横幅＝Ｒ（ｒ２） |
| ３８ |  | 単音出力<br>Midi out（MDo） | ｒ | MIDI の単音出力<br>（注）MIDI ファイル再生中は出力できない |
| ３９ |  | 音色変更<br>Midi change（MDc） | ｒ | MIDI の音色変更<br>０～１２７まで |
| ４０ |  | 画面クリアー<br>Clear（CLR） | ｒ | レジスタｒの色で画面を塗りつぶす |

| | | | | |
|---|---|---|---|---|
| ４１ |  | テキストサイズ<br>Size text（SIt） | ｒ | レジスタｒの値を文字サイズに指定 |
| ４２ |  | ビットマップ表示<br>Bitmap（BMP） | Ｓ | ビットマップファイルＳを表示する（注１） |
| ４３ |  | 再生<br>Midi play（MDp） | Ｓ | MIDI ファイルＳを再生する（注１） |
| ４４ |  | 停止<br>Midi stop（MDs） | 無し | MIDI ファイルの再生を止める |
| ４５ |  | マウス感知<br>Mouse（MUS） | ｒ | マウスの状態をレジスタｒ、Ｘ座標をｒの次のレジスタ、Ｙ座標をｒの次の次のレジスタに取り込む。<br>左クリック　Ｒ（ｒ）　←　－１<br>右クリック　Ｒ（ｒ）　←　１<br>それ以外　　Ｒ（ｒ）　←　０<br>Ｒ（ｒ＋１）　←　Ｘ座標<br>Ｒ（ｒ＋２）　←　Ｙ座標 |
| ４６ |  | 塗りつぶし四角描画<br>Solid Square（Ssq） | ｒ１，ｒ２ | 座標Ｐを中心に塗りつぶした四角を描画する。<br>高さ＝Ｒ（ｒ１）　横幅＝Ｒ（ｒ２） |
| ４７ |  | 塗りつぶし円描画<br>Solid circle（Sci） | ｒ１，ｒ２ | 座標Ｐを中心に塗りつぶした楕円を描画する。<br>高さ＝Ｒ（ｒ１）　横幅＝Ｒ（ｒ２） |
| ４８<br>＊ |  | 複写<br>Copy | | ピースをドラッグ＆ドロップすることでピースをコピーする。 |
| ４９ |  | 未定義 | | |

（注）

ｒ，ｒ１，ｒ２　　レジスタの番号を示す。指定できるのは０～１２７。

Ｒ（ｒ），Ｒ（ｒ＋１）　ｒが７ならＲ（ｒ）はＲ７，Ｒ（ｒ＋１）はＲ８を示す。

ｎ　　正数。入力できるのは-2,147,483,648 ～ 2,147,483,647。

| | |
|---|---|
| Ｓ | 文字列。 |
| ← | 演算結果または値を、左辺のレジスタに格納することを示す。 |
| ｓｔ（　） | アドレスをスタックすることを示す。 |
| ｐｏｐ（　） | アドレスをポップすることを示す。 |
| ａｄｄ（Ｐｏｉｎｔ　ｎ） | Ｐｏｉｎｔ　ｎのアドレスを示す。 |

（注１）ファイルのアドレスは絶対アドレスで記入するか、gaon ファイルと同じフォルダに保存してファイル名だけを書く

# ３　プログラミング環境の実現

## ３．１　本環境の構成

本環境はプログラム実行部とピース配置部からできています。プログラム実行部はプログラムファイルを読み込みんでおいて、トレースモード、視聴モードから呼ばれるたびに命令を一つ実行し、返値とエラーフラグを渡します。

ピース配置部は、メイン画面とピース選択画面のピースを管理し、ピースの配置が変わった場合ピース番号を命令に換えてプログラムファイルを書き換えます。また、プログラムファイルを読み込んで命令をピース番号に置き換え画面上にピースを表示させます。

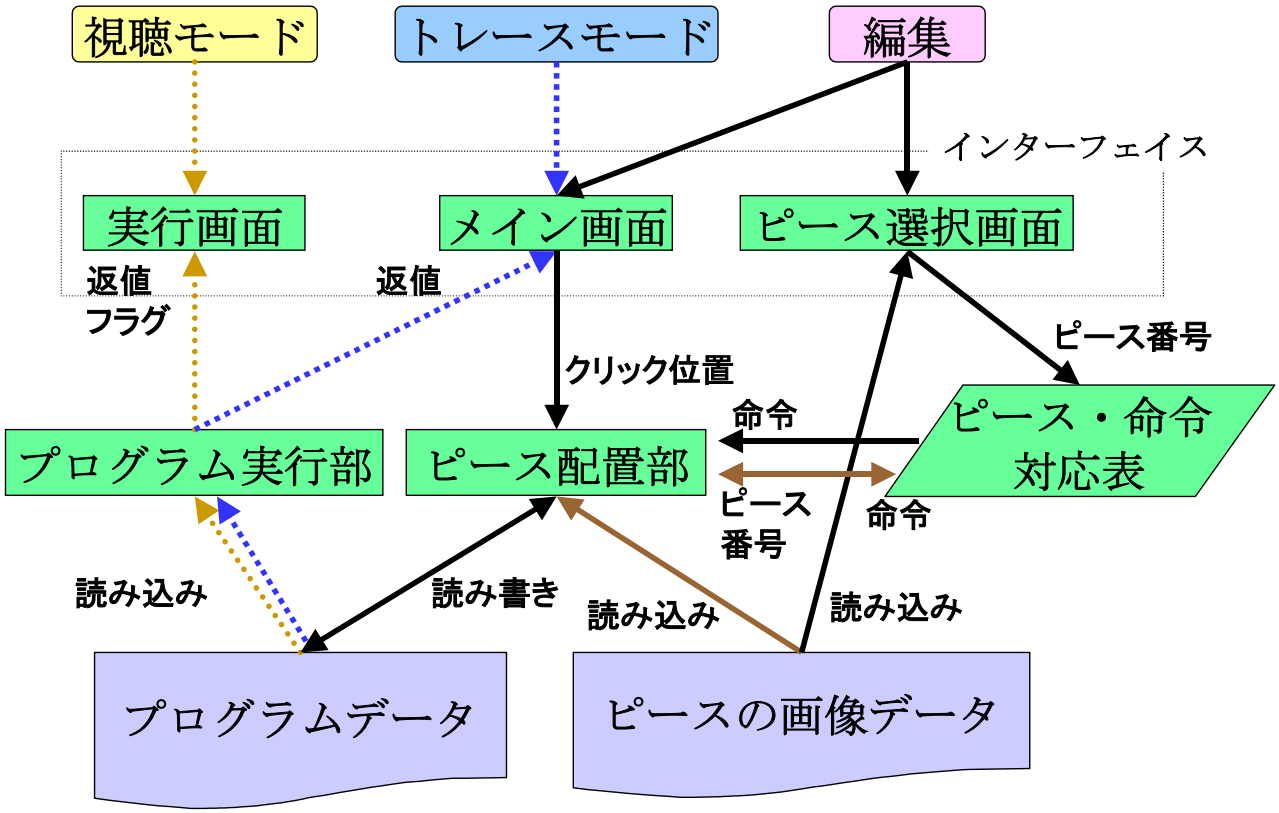



図３．１　構成図

### ３．１．１　視聴モード

メニューの「実行：視聴モード」が選ばれると、プログラムファイルを実行部に写し、実行画面を表示される。

実行画面のスタートボタンが押されるとエラーフラグ、レジスタの初期化、開始位置の検索、実行間隔の指定しＳｅｔＴｉｍｅｒ関数で指定した間隔でタイマー起動（図３．２）。ＯｎＴｉｍｅｒ関数に実行の制御が移る。

ＯｎＴｉｍｅｒ関数

返値をＳＷＩＣＨ文で分岐させ図形の表示、ＭＩＤＩの再生、または、実行を終了する。そのあと、レジスタの内容を表示をタイマーが破棄されるまで繰り返す（図３．３）

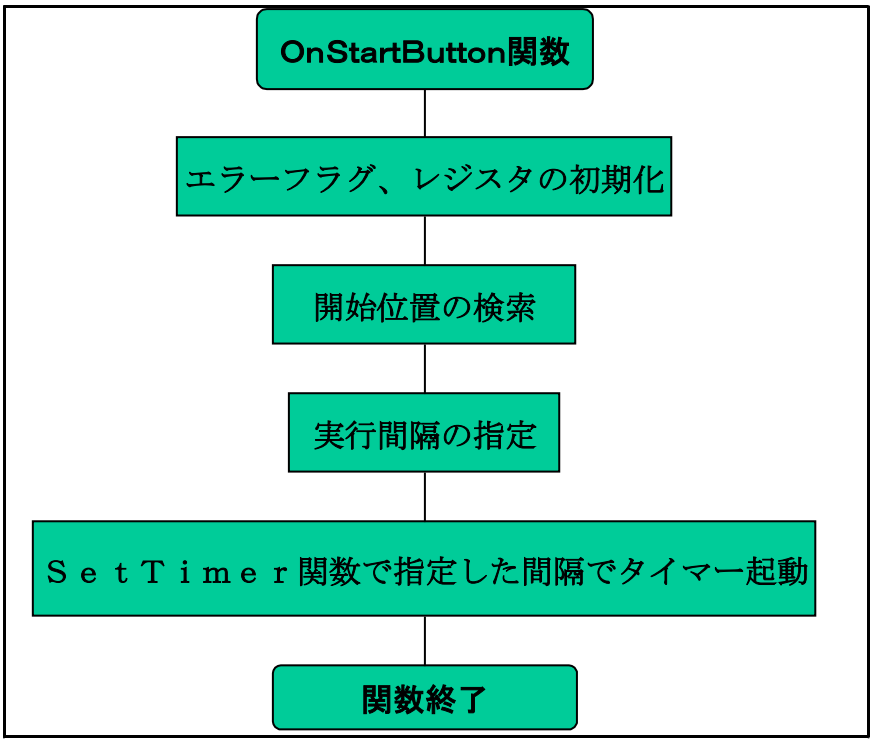



図３．２
ＯｎＳｔａｒｔ
Ｂｕｔｔｏｎ関数

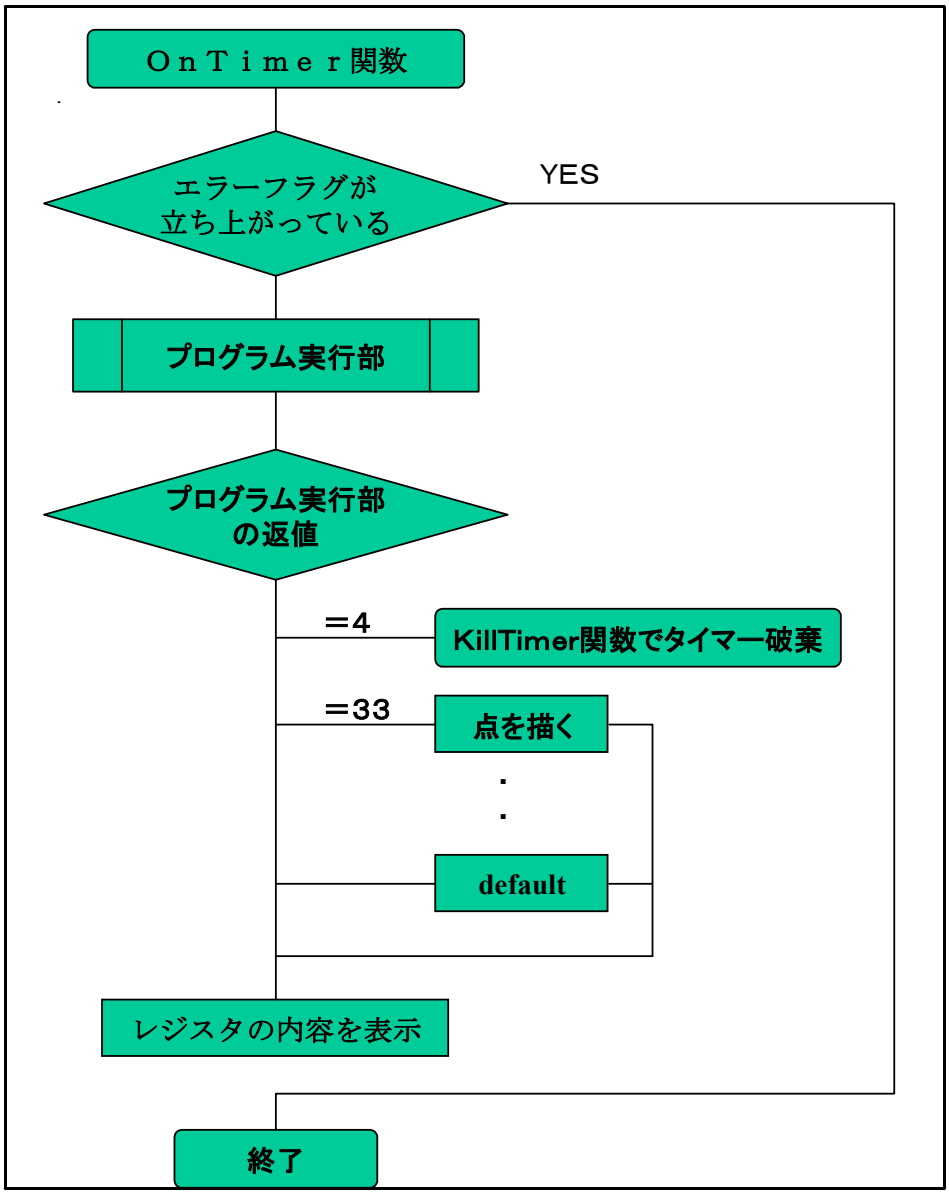



図３．３
ＯｎＴｉｍｅｒ関数

### ３．１．２　トレースモード

メニューの「実行：トレースモード」が選ばれると、トレースフラグがあがっていなければ、レジスタの初期化、開始位置の検索、実行間隔を指定し開始位置の色を反転させトレースフラグを上げる。上がっていればトレースフラグを下げ、トレースフラグを下げる。トレースフラグが上がるとＯｎＬＢｕｔｔｏｎＵｐ関数に実行の制御が移される。

ＯｎＬＢｕｔｔｏｎＵｐ関数
トレースモード実行中ならプログラム実行部を呼び出し、返値が４ならトレースフラグを下げ、トレースモード実行を終了。それ以外なら実行中のピースの色を反転させる。

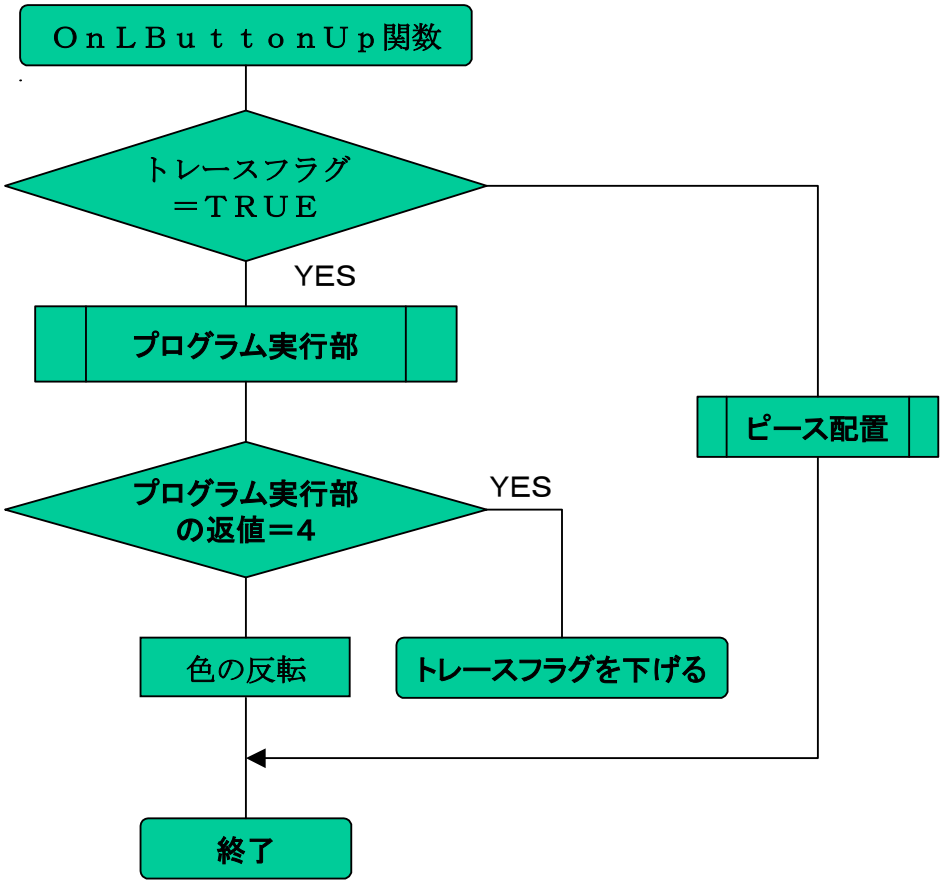



図３．４　ＯｎＬＢｕｔｔｏｎＵｐ関数

### ３．１．３　プログラム実行部

配置画面の端に来ていなければ命令を１つ取り出し、判別し処理を行う。端に来ていれば４を返す。

トレースモードと視聴モードのどちらでも同じ処理のものは、ここで処理し、エラーの出る可能性のある場合はエラーになっていないか判別する。エラーが出なければ０を返す。エラーならば４を返す。

図形や音を扱う命令など、違う処理をする場合はそれぞれ違う値を返す。

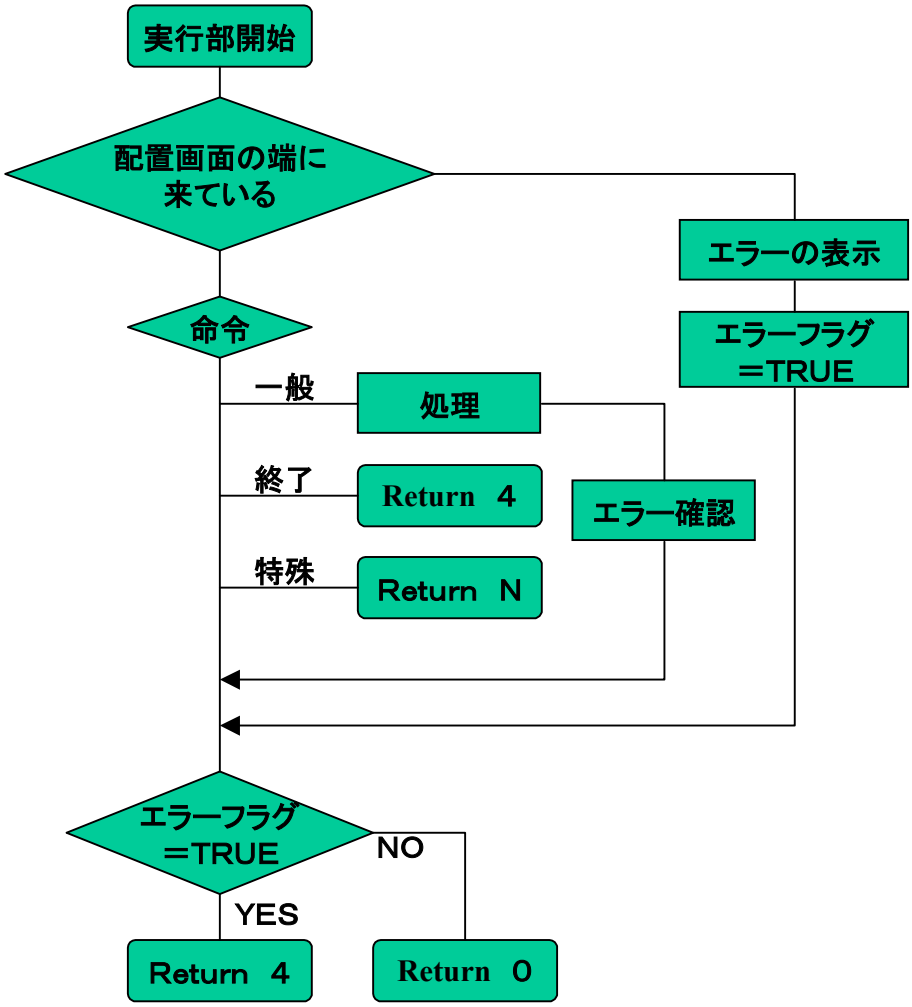



図３．５　プログラム実行部

### ３．１．４　編集

ピース選択画面でぴーすをクリックすると、マウスの位置からピース番号を割り出し、さらに、そのピース番号を配列にはめて命令に換えて、変数に保持する。

メイン画面でクリックすると、トレースモード実行中でなければピース配置へ。

### ３．１．５　ピース配置

変更フラグを下げる。

ピース選択画面で選ばれている命令に必要な引数ごとに分岐する。

ダイアログに表示する説明と入力に必要なエディットの数を渡して、引数入力画面を表示させる。

ダイアログはエディットに記入があり、ＯＫボタンが押されたときＩＤ＿ＯＫを返す。

ＩＤ＿ＯＫを受け取ったとき変更フラグ上げ、変更された命令を格納する。

変更フラグが上がっていたらプログラムを書き換える。

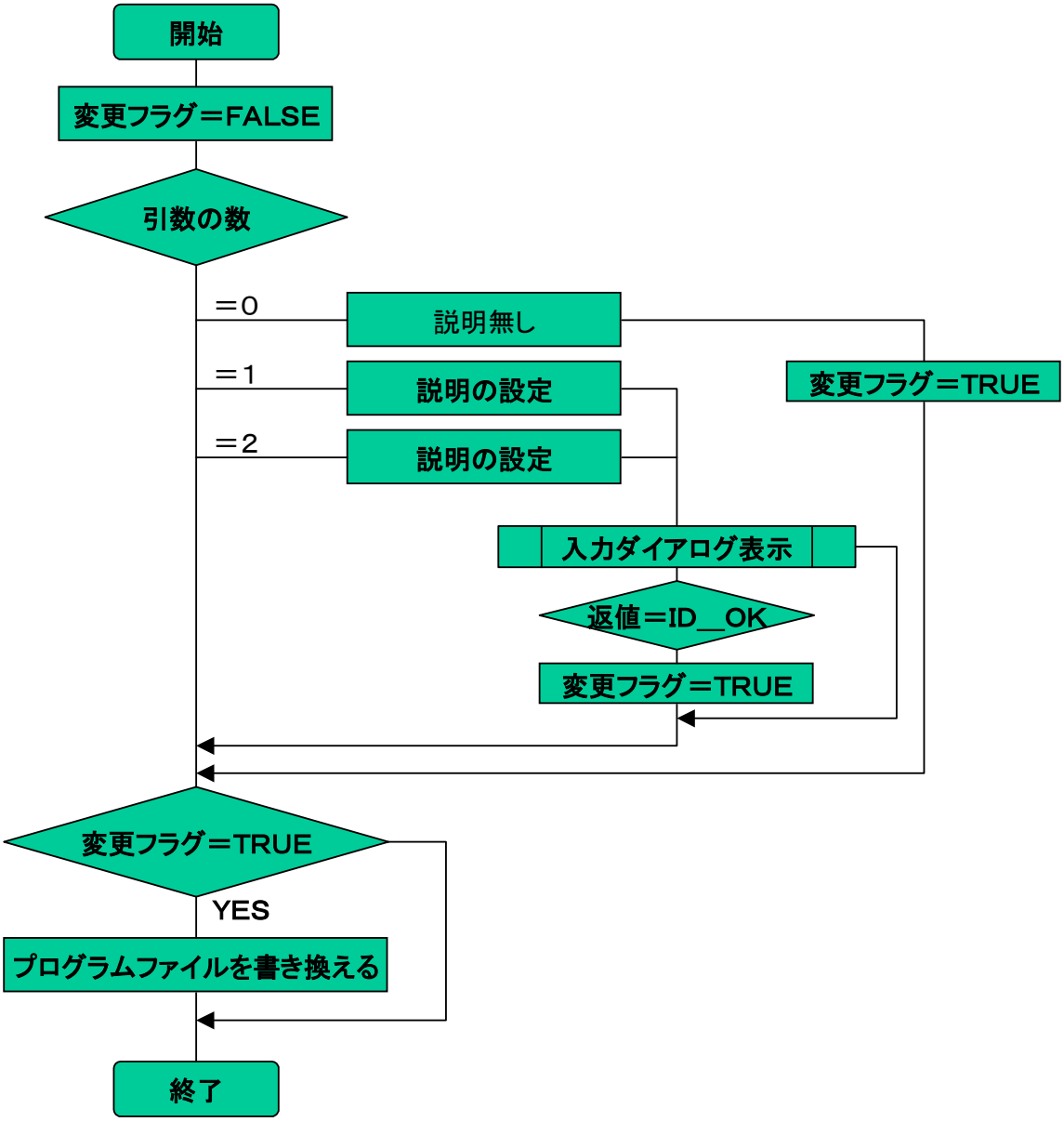



図３．６　ピース配置部

## ３．２　本環境でのプログラミング手順

ピース選択、配置を繰り返しプログラミングしてから実行します。これを繰り返してプログラムを完成させます。従来のプログラミングと変わらず、デバッグ＆ランが基本である。プログラミング手順を下に示す

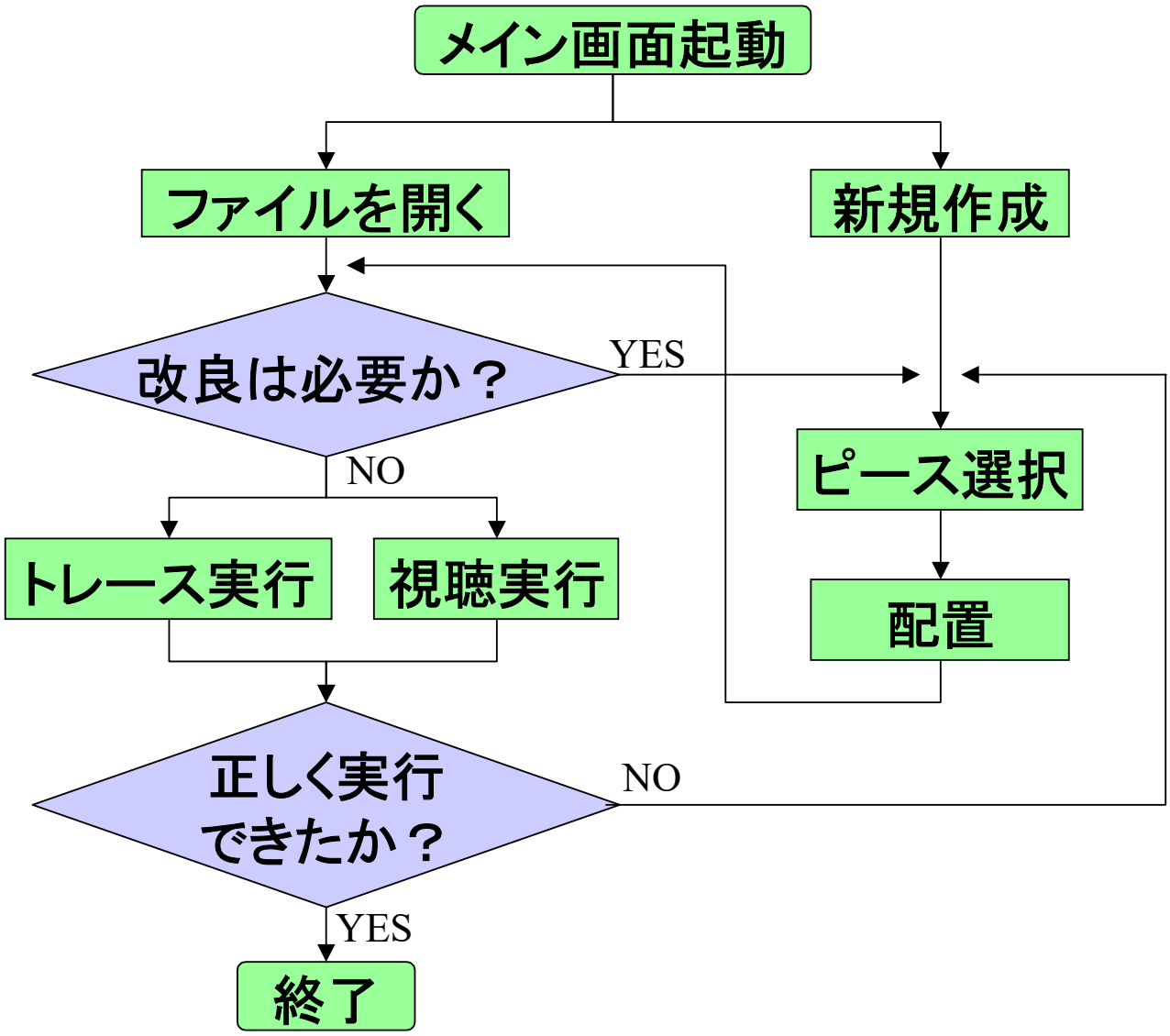



図３．７　プログラミング手順

## ３．３　ピースとコードの対応

「Ｈｅｌｌｏ　ｗｏｒｌｄ」と表示するプログラムを例にピースとコードの対応について説明します。

ピース１個がコード１行に置き換わり、先頭３文字が命令になっています。その後ろに第１引数、コンマで区切られて第２引数が続きます。

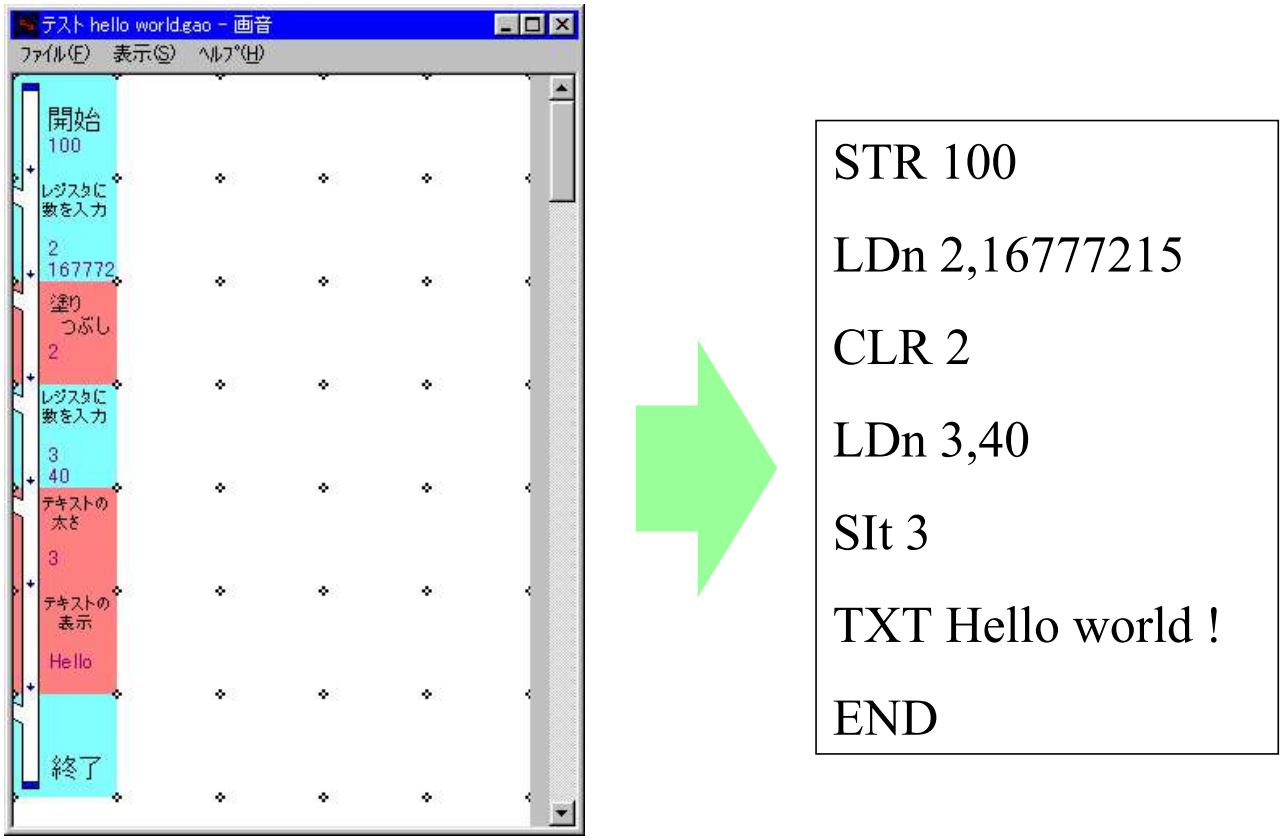



図３．８　命令とピースの対応

# ４．使用法

## ４．１　画面の説明

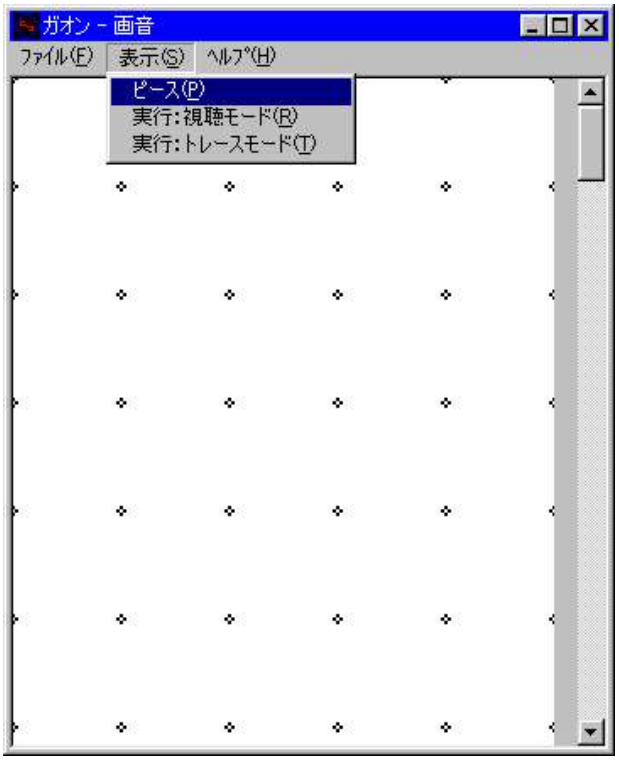



図４．１メイン画画

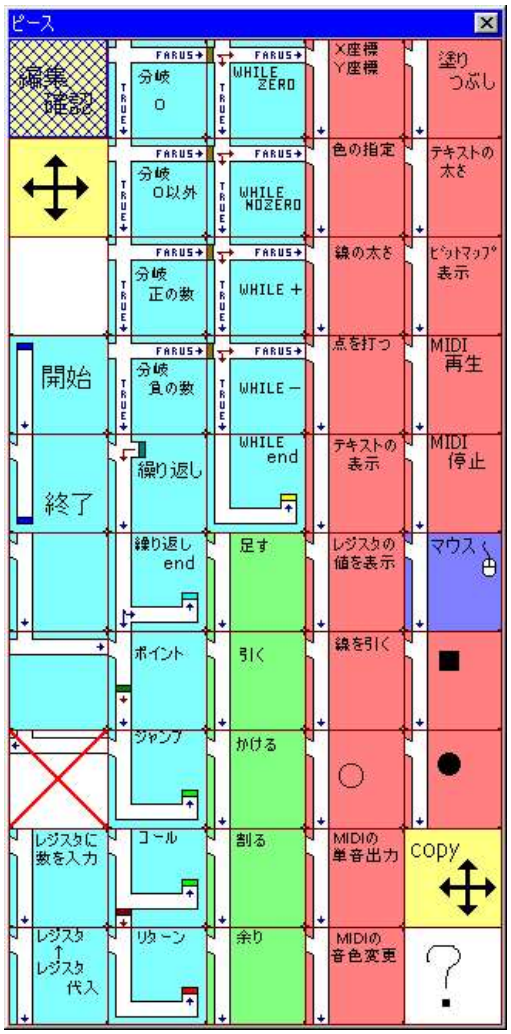



図４．２ピース選択画面

**メイン画面**

ピースを配置する画面。

メニューの説明

「ファイル」からファイルの保存やファイルを開くことができる。

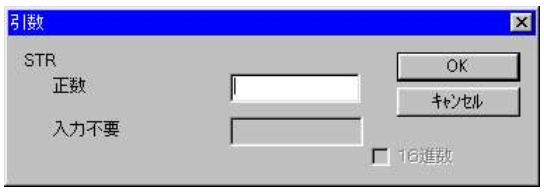
「表示」→「ピース」を選ぶとピース選択画面が表示される。

「表示」→「視聴モード」を選ぶと実行画面が表示される。

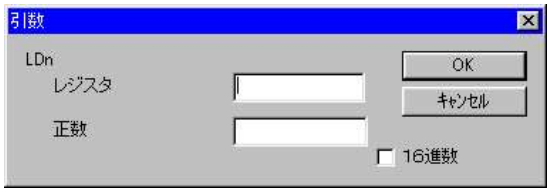
「表示」→「トレースモード」を選ぶとトレース実行が開始される。トレース中にもう一度選ぶとトレース実行を終了する。

**ピース選択画面**

利用したいピースの上でクリックすると青い網がかかり、選択される。




開始ピースの場合




レジスタに数を入力ピースの場合

図４．３　引数入力画面

**引数入力画面**

メイン画面にピースを配置するときに表示される。

入力の必要なエディットだけ入力可能状態になる。

「１６進数」にチェックをはめると１６進数で入力も可能になる。

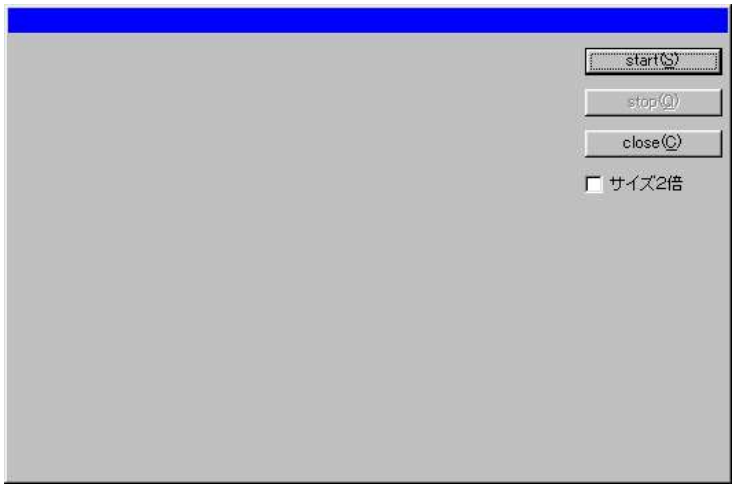



図４．４　実行画面

**実行画面**

START ボタンでプログラムが開始される。

STOP ボタンでプログラムが停止される。

CLOSE ボタンで実行画面を閉じる。

押すことの可能なボタンだけが文字が黒く表示される。

「サイズ２倍」にチェックをはめると実行時の画面が２倍になる。

## ４．２　プログラミング例

ここで、「Ｈｅｌｌｏ　ｗｏｒｌｄ！」と表示するプログラムを作る手順を説明する。

起動するとメイン画面が現れる（図４．１）。

メインメニューの「表示」から「ピース」を選ぶとピース選択画面が表示される（図４．２）。

「開始」ピースをクリックし選択する。メイン画面の好きな位置でクリックすると、入力画面が表示されるので、今回は「１００」と入力し「ＯＫ」をおす。すると「開始」ピースが配置される。（図４．５）

次はテキストの表示ピースを選択し、「開始」ピースの下に配置する。入力画面には「Ｈｅｌｌｏ　ｗｏｒｌｄ！」を入力する。

最後に「終了」ピースを選択し「テキスト表示」ピースの下に配置する。入力は必要ないのでそのまま配置される。

メインメニューの「表示」から「視聴モード」を選んで実行し、「Ｈｅｌｌｏ　ｗｏｒｌｄ！」を確認したら完成。（図４．６）

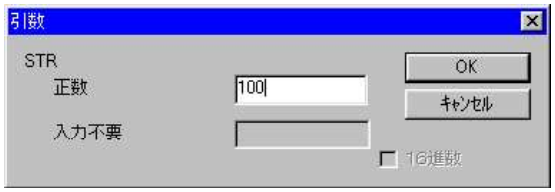



図４．５　入力画面

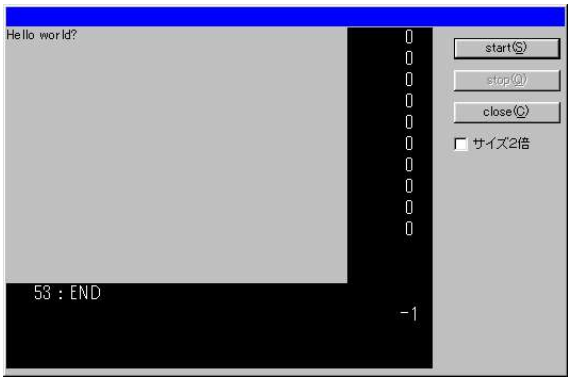



図４．６　実行後の実行画面

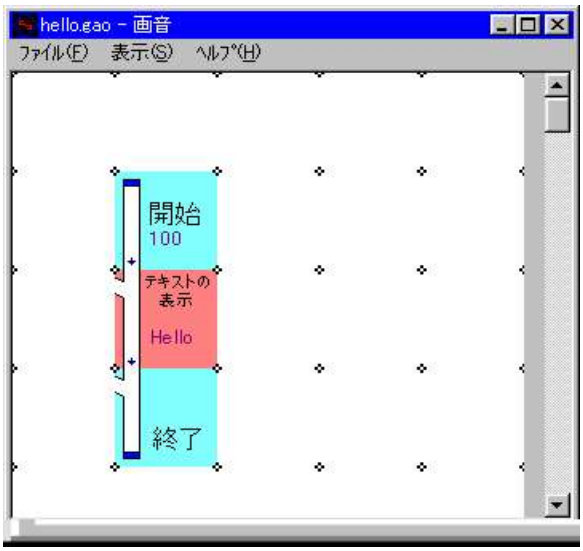



図４．７　完成した画面

# ５　終わりに

デバックの補助はエラーの表示が出来ています。これに、エラーの修正例の表示を加えることにより、より良くなると思います。また、一度作成されたプログラムを他のプログラムで利用できるようにできれば、時間短縮になり、その分勉強に時間を使え、さらによい環境になると思います。

# 謝辞

本環境の作成にあたり、２年間多々なる助言、ご指導いただいた宮武教官に心からお礼申し上げます。また、共に研究をがんばった中下さん、三岡君、村上君に感謝いたします。

# 参考文献

情報処理技術者試験　出題範囲　http://www.jitec.jipdec.or.jp/1_13download/hani01.pdf
情報処理技術者試験センター　http://www.jitec.jipdec.or.jp/
VC++ML（Programming with VC++）　http://mfc.acty-net.ne.jp/ml/mfc/

わかるＣ言語　佐藤孝幸・増井和也　共著　（学研）