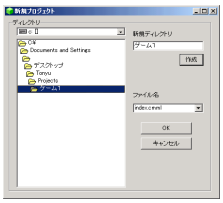
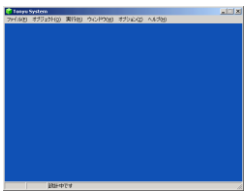
Tonyu System

ゲームの作り方

・大まかな流れ

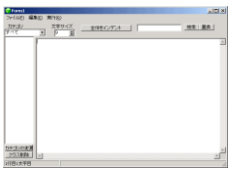


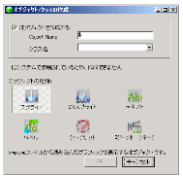
新規プロジェクト

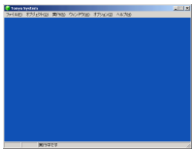


設計画面

- プログラミング
- オブジェクト/クラス作成
- グラフィック登録
- 音登録
- ページ作成
- マップ作成・編集
- オブジェクト配置



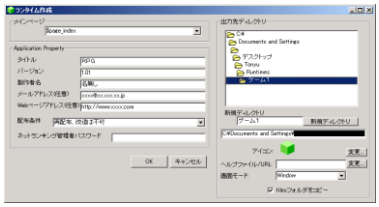




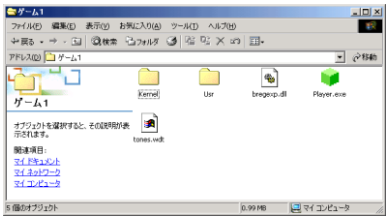
実行画面

まだ作る時・バグ探しなど

完成時



ランタイム作成



実行ファイルの完成

・意味

新規プロジェクト	：	プログラムを、どこに保存するか決める。
設計画面	：	設計中の画面。マップやオブジェクトを、表示している。
プログラミング	：	プログラムを書く。
オブジェクト/クラス作成	：	プログラムを１個（１ファイル）作る。
グラフィック登録	：	Tonyuでは、グラフィックは、登録制になってる。ファイルを追加すると、その絵が使える。
音登録	：	音も、登録制になってる。ファイルを追加すると使える。
ページ作成・編集	：	１ページ、１ステージとして使う。ステージを増やす時に、作成する。
マップ作成・編集	：	１ページにつき、１マップ作れる。設計画面で、編集できる。
オブジェクト配置	：	キャラクターなどを配置する。ページが変わると、配置も変わる。
実行画面	：	ゲームを動かす。ちゃんと動くか確かめる。
ランタイム作成	：	プレイヤーが遊べるようにする。タイトル・製作者など入力。
実行ファイルの完成	：	遊べるようになる。(Player.exeを起動する)

Tonyu System

Tonyuの仕組

プログラム 2020/10/22追記：ここは説明がちょっと不適切なので参考程度にしてください。

普通のプログラム　：１つのプログラム　＝１つのシステムができる。（例：ＣＯＢＯＬなど）
Tonyu System　　：何個かのプログラム＝１つのシステム（ゲーム）ができる。
Tonyuでは、１つのプログラムのことを、クラスという。
　　例（味方クラス・敵クラス・システムクラス）
例のように、クラスとは、それぞれの役割を持ったプログラムです。

クラスは、オブジェクトともいいます。だいたい同じ意味ですが、
クラス　　：１つのプログラム自体。
オブジェクト：あるプログラムに、基づいて動くもの。
　　例（味方オブジェクト・敵オブジェクト・システムオブジェクト）

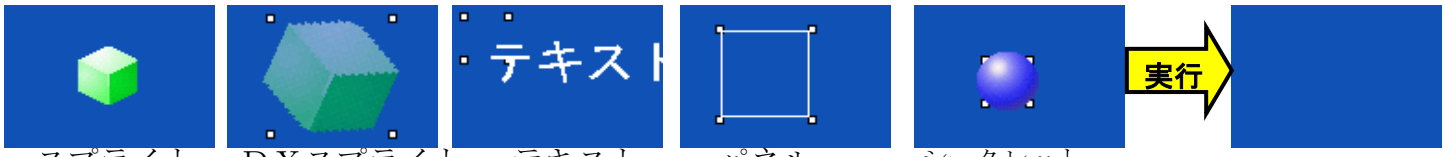
それぞれのプログラムファイルの例

ＣＯＢＯＬのプログラム	Tonyuのプログラム		
REI1.CBL （プログラムファイル）	mikata.tonyu （味方クラス）	teki.tonyu （敵クラス）	system.tonyu （システムクラス）

クラスの種類

クラスには、種類があって、普通はこの６種類を使う。（他にもいっぱいある）

スプライト	：	絵が表示できる。横反転ができる。
ＤＸスプライト	：	絵が表示できる。横反転・回転・半透明・拡大縮小ができる。
テキスト	：	文字を表示できる。サイズ・色が設定できる。
パネル	：	絵を作って表示できる。回転・半透明・拡大縮小ができる。
シークレット	：	実行中は、何も表示しない。プログラムの処理だけする。
フレームマネージャー	：	細かい処理ができる。実行中は、表示しない。



スプライト　ＤＸスプライト　テキスト　パネル　シークレット・フレームマネージャー

描画の回数

表示のことを、描画ともいう。
Tonyuは、１秒間に６０コマ分の処理をする。
つまり、描画も、１秒間に６０回している。
でも、パソコンの処理が重いときは、描画回数が減る。
プログラムでは、「ここまでが、１コマです」という命令がある。

パフォーマンスのFPSを見れば、描画回数がわかる。
上の場合は、62回描画している

グラフィックと音

Tonyuでは、グラフィックと音は登録制になっており、登録して初めて使うことができる。
それぞれ、ファイルのリストがあって、追加・削除ができる。
ただし、グラフィックは、ページごとにリストがある。

使えるファイル

グラフィック	：	bmpファイル,pngファイル
音	：	midファイル(音楽),wavファイル(効果音),mzoファイル(Tonyu用の音楽)

Tonyu System

Tonyuの仕組２

ページ

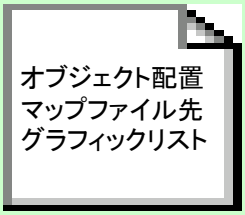
ページとは、例えば１面・２面・３面・スタート画面など、場面を変える時に便利です。
１ページには、オブジェクト配置・マップファイル・グラフィックリストの、情報が入ってて
ページが変わると、この３つの情報も変わります。（画面も変わる）

index.cmml
ステージ１

index2.cmml
ステージ２

index3.cmml
ステージ３

start.cmml
スタート画面

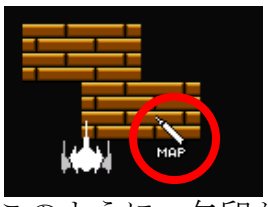


～.cmml

ページファイルには、
これらの情報が入ってる。

マップ

マップは、同じ大きさの画像を、画面に並べています。
マップは１ページに、１つ作成できます。画面から、ペンで描くように編集できます。



index.map
マップファイル

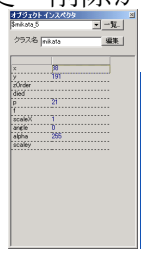
マップを作成すると、マップファイルができます。
名前は、ページ名と同じで、最後に「～.map」と付きます。

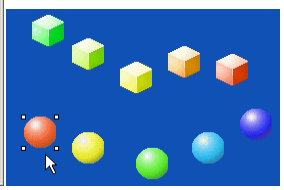
このように、矢印がペンに変わって、ドット絵を描くように、編集できます。

オブジェクトの配置

オブジェクトは、画面に配置できます。
実行時には、その配置した場所から、オブジェクトが動きます。
オブジェクトは、新規作成・切り取り・コピー・貼り付け・名前の変更・削除ができます。

オブジェクトは、クラス名とオブジェクト名の２つの名前をもっています。
オブジェクト名は、最初に「$」が付きます。
同じクラス名の、オブジェクトも作成できます。
　　例（tekiクラスのオブジェクトを、１０個作ることもできる）
ただし、同じページ内で、同じオブジェクト名は使えません。
　　例（オブジェクト名は、$teki, $teki_1, $teki_2, $teki_3, …）
また、クラス名は原則として、変更できません。
ｵﾌﾞｼﾞｪｸﾄｲﾝｽﾍﾟｸﾀで、ｵﾌﾞｼﾞｪｸﾄの情報を見れます。





この画面では、mikataｸﾗｽを１０個出しています。
画像が違っても、ｸﾗｽは同じなので、これらは、
同じﾌﾟﾛｸﾞﾗﾑで動きます。
ｵﾌﾞｼﾞｪｸﾄ名は、$mikata, $mikata_1, …, $mikata_9。

ファイルの種類

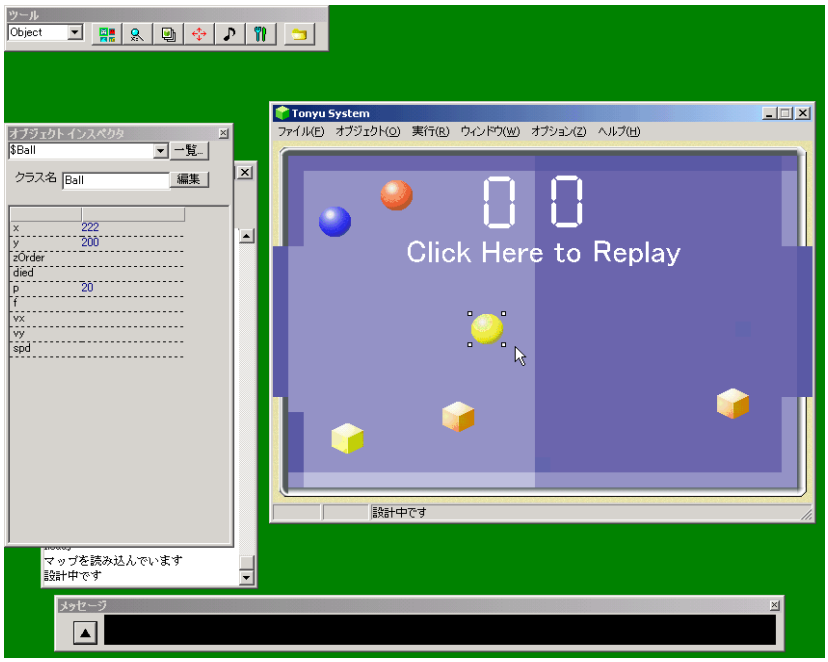
Tonyuでは、こんなファイルを使っています。　（～の部分は、てきとうな名前が入る）

デフォルトファイル	:	default.tonyuprj　　（必ず作られるファイル）
プログラム(クラス)ファイル	:	～.tonyu
ページファイル	:	～.cmml
マップファイル	:	～.map
グラフィックファイル	:	～.bmp　～.png
音ファイル	:	～.mid　～.wav　～.mzo

default.tonyuprj　：このファイルには、クラスのリスト、音のリスト、ページのリストなどが保存されています。

Tonyu System

基本操作



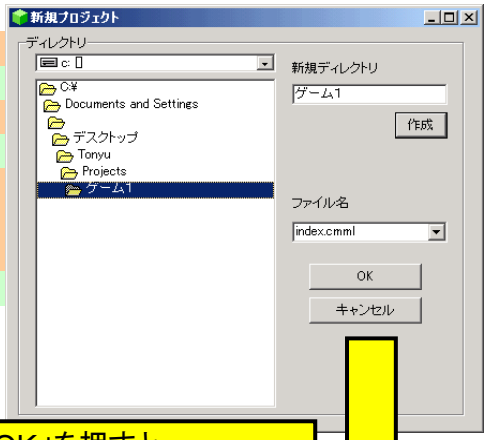
← Tonyuの画面

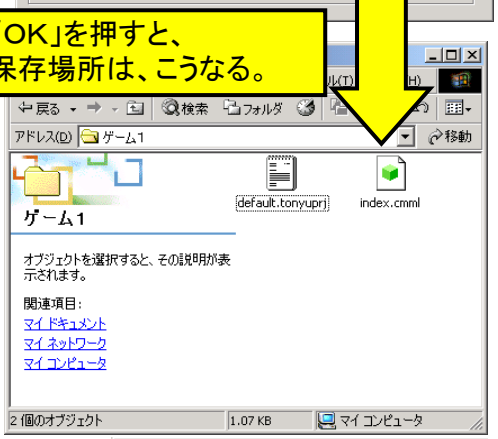
・新規プロジェクトの作成

意味　：新しいゲームを作る時に、プログラムの保存場所を、決めること。

手順

1．「ファイル」→「新規プロジェクト」
2．「projectフォルダ」をダブルクリック。
3．「新規ディレクトリ」の欄に、好きなファイル名（ゲーム名）を入力。
4．「作成」をクリック。（１回だけ）
5．「ファイル名」の欄に、半角英数字で、好きなページ名を入力。
　入れ方は、「～.cmml」。最初の１文字が、半角数字ではいけない。
　例）　×「1stage.cmml」　○「stage1.cmml」
6．「OK」をクリック。





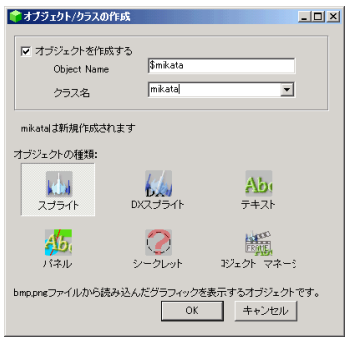
・オブジェクト/クラスの作成

意味：　新しいオブジェクトと、クラスのない場合は新しいクラスを作る

手順

1．「オブジェクト」→「新規」
2．「クラス名」に好きな名前を入力。
　日本語入力可能。
　最初の１文字が半角数字では、ダメ！
3．オブジェクトの種類を選ぶ。
4．「OK」ボタンをクリック。

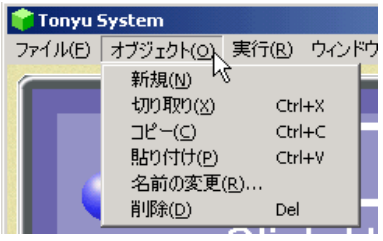
※クラス名は、一度決めたら変更できない。



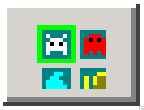
・オブジェクトの配置

オブジェクトは、自由に配置できる。
切り取り・コピー・貼り付け・削除・オブジェクト名の変更ができる。

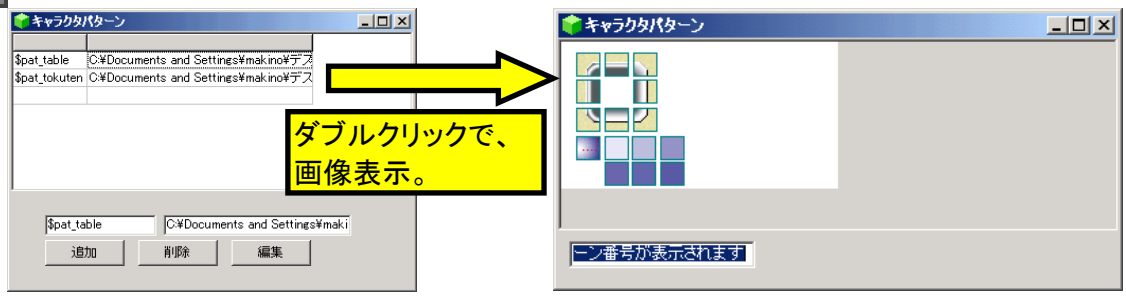
オブジェクト名は変更できますが、クラス名は変更できません。



・グラフィックや音を使う
　これらは、登録制になっている。ツールのボタンから、リストを出すことができる。

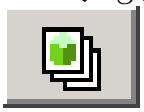


これを押すと、グラフィックのリストが出てくる。
グラフィックのリストは、ページごとに変わります。



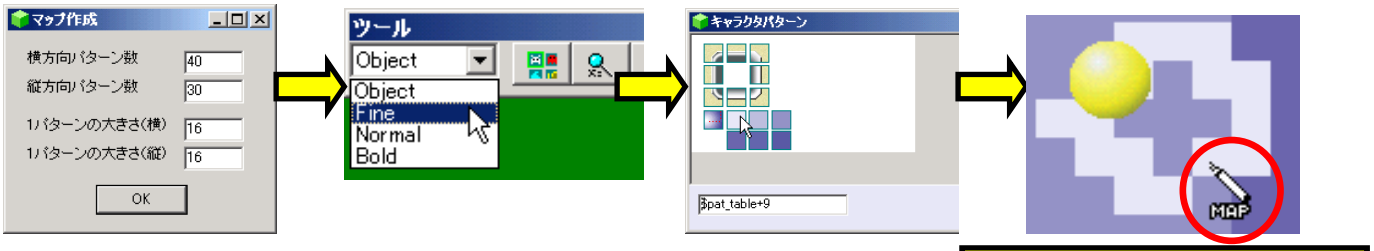
これを押すと、音のリストが出てくる。
音のリストは、特に変わりません。

・ページ
　ページも、リストになっている。



このボタンを押すと、ページのリストが出てくる。

・マップ
　マップは、１ページに１つ作成できます。作成して、初めてマップが使えます。
「ウィンドウ」→「マップ作成」と進み、マップの大きさ・長さを設定すると、作成されます。
ツールの「object」を「Fine」にして、グラフィックのリストから、マップの絵を選んで、かけます。



マウスが、ペンになり
マップをかけるようになる。

Tonyu System

確かめ問題

＜問題＞　（赤枠のところは重要）	＜回答＞
１．新規プロジェクトとは？	保存先を決める
２．設計画面には、何が写ってる？	マップと、オブジェクト
３．ページは、何のためにある？	場面を変えるため
４．ページのファイルの中身は、どんな内容が書かれている？	ｵﾌﾞｼﾞｪｸﾄ配置・ﾏｯﾌﾟﾌｧｲﾙ先・ｸﾞﾗﾌｨｯｸﾘｽﾄの情報
５．ページのファイルの拡張子は？	cmml
６．マップは、最初から作られてる？	作られてない
７．１つのページに、マップは何個作れる？	１個
８．マップの情報は、ページファイルに書かれている？	書かれてない
９．マップのファイルの拡張子は？	map
１０．グラフィックでは、どういう拡張子のファイルが、使える？	bmpとpng
１１．グラフィックのリストは、ページごとに変わるか？	変わる
１２．音では、どういう拡張子のファイルが、使える？	midとwavとmzo
１３．音のリストは、ページことに変わるか？	変わらない
１４．クラスとは何？	１つのプログラム自体
１５．オブジェクトとは何？	あるプログラムに、基づいて動くもの
１６．１つのオブジェクトには、名前が２つある。何と何か？	クラス名・オブジェクト名
１７．オブジェクト名は、最初に何がつくか？	$
１８．同じページ内に、同じクラス名のオブジェクトは、作れるか？	作れる
１９．同じページ内に、同じオブジェクト名のオブジェクトは、作れるか？	作れない
２０．クラス名は、変更できるか？	できない
２１．オブジェクト名は、変更できるか？	できる
２２．クラスは、普通、何種類使うか？	６種類
２３．クラスの種類を、答えよ。	ｽﾌﾟﾗｲﾄ・DXｽﾌﾟﾗｲﾄ・ﾃｷｽﾄ・ﾊﾟﾈﾙ・ｼｰｸﾚｯﾄ・ﾌﾚｰﾑﾏﾈｰｼﾞｬｰ
２４．クラスのファイルの拡張子は？	tonyu
２５．デフォルトファイルの名前は？	default.tonyuprj
２６．デフォルトファイルの中身は、どんな内容が書かれてる？	クラスのリスト・音のリスト・ページのリスト
２７．実行画面では、どうなる？	ゲームが動く
２８．ランタイム作成とは？	プレイヤーが、遊べるようにする
２９．実行ファイルが、完成すると？	プレイヤーが、遊べるようになる

＜解説＞

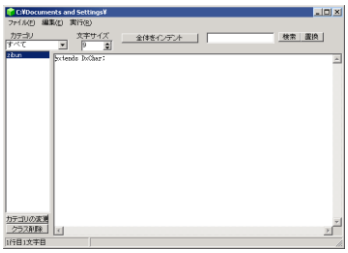
６．マップは、「ウィンドウ」→「マップ作成」で、設定してから、使えるようになる。
７．その時の、マップファイルの名前は、ページ名と同じで、最後だけ「.map」となる。
８．マップの情報は、マップファイルにあり、ページには、マップファイルの名前が書いてある。
１１．グラフィックのリストは、ページファイルに、保存されている。
１３．音のリストは、デフォルトファイルに、保存されている。
１７．例えば、$tama $tama_1 $tama_2 $teki $teki_1 $teki_2 など。
２０．変更する場合は、一回クラスを削除しなければならない。
２１．「オブジェクト」→「名前の変更」・右クリック→「名前の変更」で、変更できる。
２５．この名前は、絶対変わらない。
２６．音のリストは、ページファイルには、書かれてないので、注意！
２９．Tonyuには、開発版と、ランタイム版がある。
開発版でも、実行画面で遊べるが、開く作業が面倒。
ランタイム版は、実行ファイル(Player.exe)を、起動するだけで遊べる。

プログラムの基本

・プログラムの開き方

設計中に、画面のオブジェクトをダブルクリックすると、そのオブジェクトのプログラムが出る。
このウィンドウは、「エディタ」という。これで、プログラミングする。

ダブルクリック



エディタ

・プログラムの意味

・最初から書いてある命令

クラスを作成すると、最初から命令が１行書かれています。これは、クラスの種類です。
extendsの次の文が、クラスの種類です。
(Tonyuでは、半角の大文字・小文字は区別しません)
この命令は、あまりいじりません。

SpriteChar	：	スプライト
DxChar	：	ＤＸスプライト
TextChar	：	テキスト
PanelChar	：	パネル
SecretChar	：	シークレット
FrameManager	：	フレームマネージャー

・簡単な文（printメソッド）

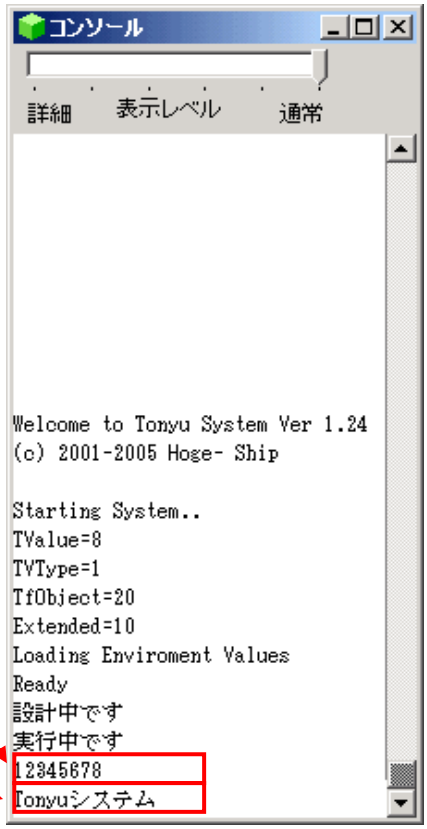
Tonyuで、いちばん簡単なのは、「文字を表示すること」です。
print();と書いて、()内に数字や文字を入れると、
()内のものがコンソールに表示されます。

・数字を表示

例えば、12345678という数字を、表示したい時は、
print(12345678);と書きます。

・文字を表示

文字を表示したい時は、""でくくります。
例えば、Tonyuシステムという文字を、表示したい時は、
print("Tonyuシステム");と書きます。



print(12345678);と、print("12345678");は、
同じように、表示されます。
しかし、print(12345678);は、数字扱いになり、
print("12345678");は、文字扱いになります。
この違いは、後で説明します。

・メソッド

Tonyuでは、ある命令のことを、メソッドといいます。
メソッドは、ほとんどが、～(～);のような形です。
()がついてたら、メソッドだと思ってもいいです。（例外もある）
例：　print();は、printメソッドという。

printメソッドは、処理結果を表示するのに使います。
なので、Tonyuを学習するのに、便利な命令です。
これからは、この命令を使っていきます。

プログラムの基本２

・計算

Tonyuでは、計算ができます。計算方法は、四則演算・比較演算・論理演算の３つです。

四則演算	：	＋－×÷を使う。
比較演算	：	< <= == => > != ！ の７つを使う。
論理演算	：	&& || の２つを使う。

（これらの処理結果も、printメソッドで表示できるのです）

・四則演算

これは、算数と同じ計算ができます。
プログラムでは、右の表の黄色い部分を、使います。
なので、右の表のように、×は*　÷は/　になる。

あと、10/(3+2)のように、()も使えます。
この計算では、()が先に計算されます。（答え：2）
優先順位は、右下の表を見てください。

四則演算の文法

文法	記号	意味
+	＋	足し算。
-	－	引き算。
*	×	掛け算。
/	÷	割り算。
%		割り算の余り。

＜例＞

[プログラム]		[コンソール]
print(1+2);	：	3
print(10+5+11);	：	26
print(10-2);	：	8
print(100+200-50);	：	250
print(9*9);	：	81
print(10+10*2);	：	30
print((10+10)*2);	：	40
print(10/2);	：	5
print(10%2);	：	0
print(10/3);	：	3.33333325386047
print(10%3);	：	1
print(123%100);	：	23

計算順位

順番	文法など	
1	()	（四則演算）
2	* / %	（四則演算）
3	+ -	（四則演算）
4		（比較演算）
5	&&	（論理演算）
6	||	（論理演算）

四則演算の計算順位は、
ほとんど算数と、同じです。

（Tonyuの小数は、誤差がある）

%は、整数まで割り算をして、余りを出します。
例えば、10%3は、まず10/3をしています。/のときは、商は3.3333…になりますが、
%の時は、商3　余り1　になり、余りを結果に出すので、1を表示します。

・比較演算

２つの数字や式を比較します。条件が合えば１になり、合わなければ０になります。
これは、例えば、3<5、5>=7、5>=3+4のように、両辺に数字や式を入れて使います。

比較演算の文法
（aとbは、数字や式が入る）

a<b	：	aがbより小さい。
a<=b	：	aがb以下。
a==b	：	aはbと等しい。
a!=b	：	aはbと違う。
a>=b	：	aがb以上。
a>b	：	aがbより大きい。
!a	：	aは偽。

条件に合う	1	真
条件に合わない	0	偽

Tonyuでは、1は真。0は偽。
このように、分類されます。

＜例＞

[プログラム]	[コンソール]	
print(3<5);	1	3は5より小さいので1
print(3>5);	0	3は5より大きくないので0
print(10<10+1);	1	足し算が先に計算される
print(5==5);	1	5と5は同じなので1
print(5!=5);	0	5と5は違くないので0
print(5!=6);	1	5と6は違うので1
print(5>=5);	1	5は5以上なので1
print(5>=6);	0	5は6以上ではないので0
print(!0);	1	0は偽なので1
print(!1);	0	1は偽ではないので0

プログラムの基本３

・論理演算

これは、比較演算と組み合わせて使います。
例えば、1<3&&7>=5のように、比較演算の間に、論理演算を書くのが普通です。
計算順番は、比較演算が先で、論理演算が後です。

論理演算の文法

&&	：	～かつ～	両端が真の時のみ、真になる。
||	：	～または～	１つでも真があれば、真になる。

論理演算の計算結果

計算		結果	計算		結果
0&&0	→	0	0||0	→	0
1&&0	→	0	1||0	→	1
0&&1	→	0	0||1	→	1
1&&1	→	1	1||1	→	1

計算順位は、&&が先で、||は後です。
詳しくは、プリント8の計算順位を見てください。

＜例＞

[プログラム]	[コンソール]
print(1==0&&1==0);	0
print(1==1&&1==0);	0
print(1==1&&1==1);	1
print(1==1&&1==1&&1==1);	1
print(1==1||1==0&&1==1);	0
print(1==1&&1==0||1==1);	0
print(9<=7||8<=7||7<=7);	1
print(5>3&&5>4&&5>5);	0
print(1!=1||1<3&&2<=1);	0
print(1&&1||1&&0);	0
print(!1&&!1||!1&&!0);	0
print(1&&(0||!0)&&1);	1

・文字

さっきは、数字を使ってました。次は、文字を使います。
文字は必ず、""で囲まれています。""で囲まれているものは、計算できません。
例えば、print(200-100);できますが、print("200"-100);はできません。（無視されて「200」とでます）

でも、print("200"+"100");はできます。これの結果は、「200100」です。
これを、文字結合といいます。

・文字結合

文字と文字の間に、「+」を書くと、文字がくっつきます。
例えば、print("Tonyu"+"システム");は、「Tonyuシステム」と表示されます。

また、文字と数字の組合せでも、文字結合になります。
例えば、print(12+"月"+31+"日");だと、「12月31日」と表示されます。
文字と数字の、計算順位は同じです。

＜例＞

[プログラム}	[コンソール]	
print("今日は"+"晴れ");	今日は晴れ	
print("時間　"+20+":"+45);	時間　20:45	
print("1"+2+"3"+4+"5");	12345	
print("1+5="+(1+5));	1+5=6	()内が、先に計算されてから、結合する。
print("1+5="+1+5);	1+5=15	「"1+5=1"+5」→「"1+5=15"」になる。
print(1+1+1+"1"+1+1);	3111	最初は計算だが、文字の後は文字結合になる。
print(1+1-1+"2"+1-1*1/1+1);	1211	文字の後の、「+」以外は無視される。

・文字比較

２つの文字が、同じだったら１、違ったら０になります。
例えば、print("tonyu"=="tonyu");だと、「1」が表示されます。
大文字・小文字の区別有り！
なので、print("Tonyu"=="tonyu");は「0」になります。
ただし、print("99"==99);の場合は、「1」になります。
しかし、print(99=="99");の場合は、「0」になります。 文字と数字の組合せは、普通使いません。

[プログラム]	[コンソール]
print("a"=="a");	1
print("a"=="A");	0
print("7"=="7");	1
print("99"=="9"+9);	1

プログラムの基本４

今まで、計算、文字結合、文字比較などは、使いどころがなく、無意味でした。
でも、変数が使えると、使いどころが多くなり、便利になります。
変数がすぐに、使いこなせるように、あえて計算などを先にやらせました。

・変数

これは、ある数字や、文字を入れとく場所です。

・代入

例えば、プログラムで、

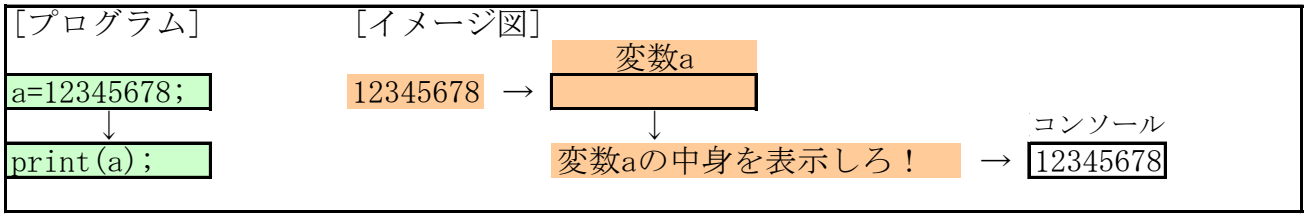
```
a=12345678;
print(a);
```

と書くと、「12345678」と表示されます。

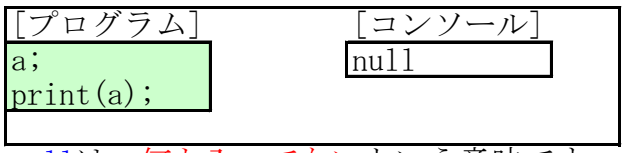
プログラムは、左から右、上から下へ、順番に動いています。　→　→

```
a=12345678;
↓
print(a);
```

まず、a=12345678;は、aという変数に「12345678」を入れてます。
次に、print(a);は、aの内容を表示しています。（print("a");とは違う）



変数を新しく作る時は、その変数名を書いたところから使えます。
ただし、最初は、「null」という、文字でも、数字でもないものが入ってます。



nullは、何も入ってないという意味です。
「=」を使うと、変数の内容が書き換えられます。このことを、代入といいます。
例を見てば、変数の便利さに気づくはず！！

＜例＞

[プログラム]		[コンソール]
a=12345678;	print(a);	12345678
a=1+5;	print(a);	6
a="あいう"+"えお";	print(a);	あいうえお
a="1+5="+(1+5);	print(a);	1+5=6
a=12;b=31;	print(a+"月"+b+"日");	12月31日
a=7;b=9;	print(a+"+"+b+"="+(a+b));	7+9=16
hi="明日";ten="晴れ";	print(hi+"の天気は"+ten+"です");	明日の天気は晴れです
a=10;b=3;	print(a+"の"+b+"倍は、"+(a*b)+"です");	10の3倍は、30です
a=1;b=3;	print(a+"<"+b+"を比較演算したら"+(a<b));	1<3を比較演算したら1
a=10;b=3;	print(a+"÷"+b+"の余りは"+(a%b));	10÷3の余りは1

・変数の名前

半角英語や、全角も使えます。ただし、最初の１文字が、半角数字だとダメ。（「_」以外の記号は使えない）
例（a　hp　mp　a1　a2　a_1　a_a　aa　体力　攻撃力　変数１　ＨＥＮＳＵ１　∂　など）

プログラムの基本５

・代入の種類

・演算子

代入は、「=」で行っていました。この「=」のことを、演算子といいます。
計算などで使った記号も、全て演算子といいます。
例（+ - * / % < <= == != >= > ! && ||）

代入には、いくつか種類があります。表を見ながら、例を見てください。

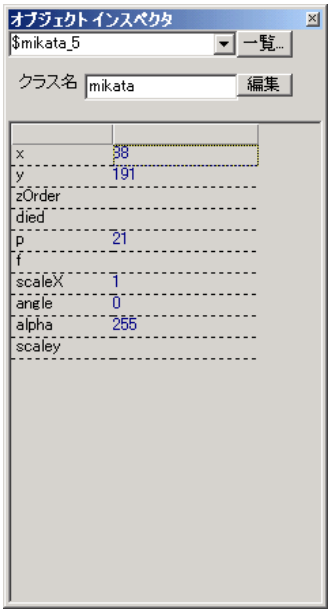
演算子	例	例の説明
=	a=2;	aに2を代入。
+=	a+=2;	aに、a+2を代入。
-=	a-=2;	aに、a-2を代入。
++	a++;	aに、a+1を代入。
--	a--;	aに、a-1を代入。

＜例＞

[プログラム]	[コンソール]
a=2;print(a);	2
a=2;a+=5;print(a);	7
a=2;a-=5;print(a);	-3
a=2;a++;print(a);	3
a=2;a--;print(a);	1

だいたいの、代入は、「=」だけでもできます。
しかし、「=」以外の演算子を使うと、便利で処理が速くなります。（速さはあまり大差ない）
とりあえず、便利な代入もできると、覚えてください。

・オブジェクト変数



今まで使ってきた変数は、「オブジェクト変数」といいます。
この変数は、オブジェクトごとに所持しています。
なので、同じ変数名でも、ｵﾌﾞｼﾞｪｸﾄが違えば、別の変数扱いになります。

例えば、$mikataのxは「48」、$mikata_1のxは「52」。
このように、同じ変数名でも、オブジェクトが違えば、
別物です。

この変数は、最初からある程度作られています。
「オブジェクトインスペクタ」で、オブジェクト変数のリストが見れます。
右のリストでは、１０個の変数が最初から作られてました。

例えば、プログラムで、新たにaという変数を作ったら、
右のリストに、aという欄ができます。

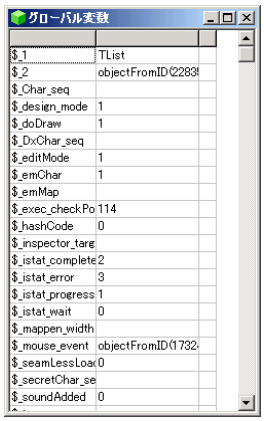
・グローバル変数

オブジェクト変数は、同じ名前の変数が、複数存在しました。
しかし、グローバル変数は、１つしか存在しません。
つまり、どのオブジェクトからも、同じ変数を参照できます。

この変数は、最初に「$」が付きます。例（$a $hp $mp $あああ $aaa_1 ）
後は、オブジェクト変数と同じようにつかえます。

＜例＞

[プログラム]	[コンソール]
$a=5;print($a);	5
$a=5;$a+=2;print($a);	7
$a=5;$a++;print("$aの内容は"+$a);	$aの内容は6



この変数も、最初から、いくつか作られています。
Tonyu自身が動くのに必要な変数や、ある機能を使うための変数などがあります。
「ウィンドウ」→「グローバル変数」から、グローバル変数のリストが見れます。

実は、オブジェクト名も、グローバル変数です。
例えば、「$mikata」というオブジェクトが、あったとします。
print($mikata);とやると、「mikata@17771128」と表示されます。（17771128の部分は毎度変わる）

なにやら、数字でも、文字でも、nullでもないものが、表示されました。
これは、$mikataのアドレス（住所）です。
アドレスは、「～@～」のように、真中に「@」がつきます。

変数 $mikata
mikata@17771128

あくまでも、$mikata自体は変数で、オブジェクトではない。
変数の中身も、アドレス（住所）なので、オブジェクトではない。
オブジェクト自体は、Tonyuが管理していて、プログラマは、直接操作できない。

例えば、$mikata=0;と書くと、$mikata変数の内容が「0」になるだけで、
オブジェクト自体は、消えない。そのまま動作を続ける。

・オブジェクトの変数参照

例えば、$mikataというオブジェクトがあって、変数xが50だったとする。
$mikataの変数xが知りたいときは、print($mikata.x);と書く。
表示結果は、「50」になります。

[ｵﾌﾞｼﾞｪｸﾄｲﾝｽﾍﾟｸﾀ]

$mikata	
x	50
…	…
…	…

このように、あるオブジェクトから、変数を参照する時は、「.」を使う。
使い方は、「変数.変数」で、左の変数の内容が、アドレスでないとエラーになる。
上の例の、意味は、「mikataオブジェクトの、変数x」という感じです。

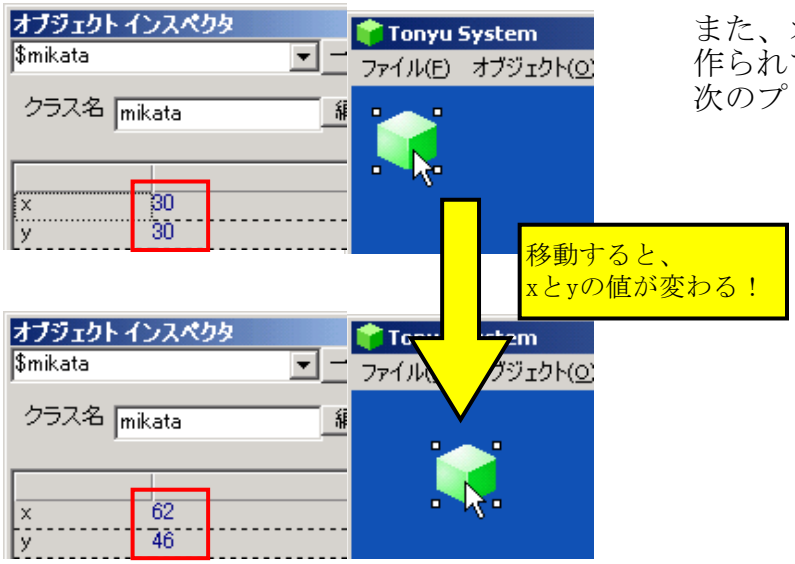
・オブジェクトを操作

ここから、やっとゲーム的なことが、できようになってきます。

・オブジェクト変数の初期値

オブジェクト変数は、最初から、いくつか作られていました。
これらの変数は、最初から、使う目的が決まっているからです。

例えば、xとyは、必ず作られます。
xは、オブジェクトの横の位置です。
yは、オブジェクトの縦の位置です。
設計画面中に、オブジェクトを移動すると、ｵﾌﾞｼﾞｪｸﾄｲﾝｽﾍﾟｸﾀの、xとyの値が変わります。



また、オブジェクトの種類によっても、
作られてる変数が、違います。
次のプリントで紹介します。

・最初からある変数
オブジェクトには、主に６種類ありました。また、それぞれ、用意されてる変数も違います。

ｽﾌﾟﾗｲﾄ
x
y
z0rder
died
p
f

DXｽﾌﾟﾗｲﾄ
x
y
z0rder
died
p
f
scaleX
angle
alpha
scaley

ﾃｷｽﾄ
x
y
z0rder
died
text
col
size
font
maxTextWidth
bold
italic
underLine

ﾊﾟﾈﾙ
x
y
z0rder
died
width
height
angle
alpha
scaleX
col
font
panel
text
bold
itaric
underLine
size

ｼｰｸﾚｯﾄ・ﾌﾚｰﾑﾏﾈｰｼﾞｬｰ
x
y
z0rder
died
pat

黄色の部分は、６種類の中では、共通です。

x	：	オブジェクトの、横の位置に使われていて、小さいほど左へ、大きいほど右へ行きます。
y	：	オブジェクトの、縦の位置に使われていて、小さいほど上へ、大きいほど下へ行きます。
z0rder	：	この値が、大きいときは、オブジェクトが後ろに表示され、 小さいときは、オブジェクトが手前に表示されます。 例えば、$aのz0rderが0、$a_1のz0rderが1、の場合は、$aが$a_1より前に表示されます。
died	：	オブジェクトがあるときは0、消えると1になります。 この変数が、1になると、残りのオブジェクト変数がすべて0になります。 この変数は、通常使いません。dieというメソッドを使います。

ｵﾌﾞｼﾞｪｸﾄｲﾝｽﾍﾟｸﾀで、変数名の右側に、数字や文字が書いてあるものもあります。
これは、初期値といって、実行する時は、この値が入った状態で、実行されます。

・スプライト
これは、画面に表示するオブジェクトです。ＤＸスプライトよりも、機能が少なく、
横反転くらいしか、ありません。特別な表示をしない時は、これにします。

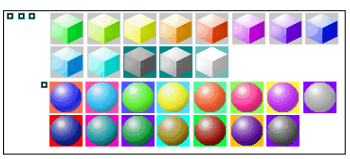
x	：	xが0の時は、このオブジェクトの中心が、画面の左端にきます。
y	：	yが0の時は、このオブジェクトの中心が、画面の上端にきます。
p	：	pは、どの画像を表示するかを、決める変数です。 Tonyuでは、あらかじめ0～31に画像が、用意されています。 例えば、pが3の時は、「緑の立体」が表示されます。
f	：	fが真の時は、画像が横反転します。

xが0、yが0

pが31

fが偽

fが真


pの0～31は、
こんな画像が使われている。

・真偽について
実は、真と偽は、1と0に限らず、
0とnullのことを偽といい、
それ以外を真といいます。

なので、fに0かnullを入れると、そのまま。
0とnull以外のものを入れると、反転します。

・ＤＸスプライト

これは、スプライトの機能に、回転・半透明・拡大縮小の機能が追加された、オブジェクトです。
特別な表示がしたいときは、これを使います。

このオブジェクトは、スプライトの変数に、さらに４つの変数が追加されています。

scaleX	：	縦横の拡大比、または、横の拡大比を入れます。 例えば、1の時は、そのまま。2の時は、２倍の大きさになります。
angle	：	角度を入れます。0～360の数字を入れますが、超えてもいい。 例えば、0の時は、そのまま。45の時は、時計回りに45度傾きます。
alpha	：	半透明度を入れます。0～255の数字を入れます。255の時は、そのまま。 0の時は、表示されません。その間の数字にすると、半透明になります。
scaley	：	縦の比率、または、使わない時もあります。 0やnullの時は、scaleXだけで、拡大縮小します。 例えば、scaleXが2、scaleyが0の時は、横２倍・縦２倍になります。 scaleXが2、scaleyが1の時は、横２倍・縦１倍になります。


scaleXが2


angleが45

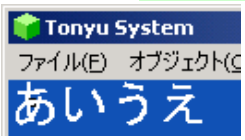

alphaが100


scaleXが2、scaleyが1

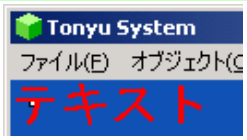
・テキスト

これは、文字を表示するための、オブジェクトです。
特に、上の３つは、よく使います。

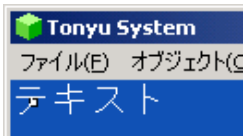
text	：	表示したい文字を入れます。文字を入れる時は""を忘れずに！
col	：	文字の色を入れます。 これは、色の変数を入れたり、colorメソッドを使います。 ｵﾌﾞｼﾞｪｸﾄｲﾝｽﾍﾟｸﾀには、変数。プログラムにはcolorメソッドを使います。
size	：	文字のサイズを入れます。
font	：	文字のフォント名を入れます。
maxTextWidth	：	文字の長さが、あるところまでいったら、改行します。
bold	：	1の時は、太字になります。
italic	：	1の時は、斜体になります。
underLine	：	1の時は、下線がつきます。



textが"あいうえ"



colが$clRed



sizeが16

colorメソッド
color(～,～,～);
～の部分に0～255の数字を入れます。
左から、赤,緑,青の量を入れます。
この３色で、色を作ります。

例えば、col=color(255,255,0);
この場合は、黄色の文字になります。

色の変数

$clRed	赤
$clYellow	黄色
$clGreen	緑
$clAqua	水色
$clBlue	青
$clPink	ﾋﾟﾝｸ
$clWhite	白
$clBlack	黒

・シークレット、フレームマネージャー

これらは、実行中は、表示を行いません。設計中は、スプライトのように表示します。
ただし、pはpatになり、fはありません。

pat	：	設計中に、表示する画像を入れます。 ｽﾌﾟﾗｲﾄなどの、変数pのように、0～31の数字を入れます。

プログラムの基本９

・オブジェクトを消させない

さっきの、オブジェクト変数を使えば、いろんなことができるのはわかりました。
でも、実際に実行してみると、何も写りません。
実は、実行直後の、最初の１コマしか、表示されてません。
２コマ目からは、オブジェクトは、消えてしまいます。(deidの値が1になる)

実は、プログラムが全部終わると、オブジェクトは消えてしまいます。
なので、プログラムを終わらないようにすればいいのです。

そこで、「while文」というものを使います。(メソッドではありません)
例えば、

```
while(1){
  update();
}
```

(１番上の、extends文の、次の行に書いてください)
こう書くと、オブジェクトは消えません。

whileの()内が、真の時は、{}内が繰り返され、偽の時は、{}内を通らないで、}の次へ行きます。
この場合は、1は真なので、updateメソッドが、呼ばれつづけます。

updateメソッドは、「ここまでが、１コマ分です」という命令です。
１秒間に、およそ60コマ分の処理が、行われています。
つまり、while分の{}内が、１秒間に、60回繰り返されています。

もし、while文に、updateがなかったら、{}内が、何万回も繰り返され、処理が重くなります。
通常は、while文の中には、updateメソッドを入れます。

・オブジェクトを使いこなす

さっきの、while文さえあれば、オブジェクトを使いこなすことができます。
例えば、ＤＸスプライトで、

```
while(1){
  angle++;
  update();
}
```

このように書くと、オブジェクトが、時計回りにゆっくりと回転し始めます。
でも、もちろんこれだけじゃ、ゲームはできません。
これからは、いろんな文やメソッドを、紹介します。

プログラム（if文）

・if文 (メソッドではない)

while文では、プログラムの流れが、変わりました。上から下へ流れるはずが、
}から上に戻ってしまいました。

この文も、プログラムの流れが変わります。例えば、

```
a=7;
if(a==7){print("if文を通りました");}
```

この場合、「if文を通りました」と、表示されます。
でも、a=7;をa=8;にすると、何も表示されません。

つまり、if文の()内が、真の時、{}内を通り、偽の時は、{}内を通り過ぎます。
while文や、if文の()内は、主に比較演算や、論理演算を使います。
比較演算や、論理演算は、このためにあったのです。

プログラム（if文・break文・for文）

また、if文には、elseというものも、使えます。（メソッドではない）

```
a=8;
if(a==7){print("if文を通りました");}else{print("こっちを通りました");}
```

この場合は、「こっちを通りました」と、表示されます。
()内が、真の時は、左の{}内を通り、偽の時は、右の{}内を通ります。
両方の{}内を、通ることはありません。

また、{}がなくてもいい場合があります。それは、命令が１つの時です。

```
a=2;
if(a)print(1);else print(0);
```

でも、命令が２つ以上の時は、必ず{}が必要です。

```
a=2;b=0;
if(a){print(1);print(a);b=1;}else{print(0);print(a);b=0;}
```

さらに、if文の中のif文というのも、できます。

```
a=10;b=20;
if(a==10){
  if(b==20)print("２つの条件が合いました");
  else print("a==10の条件が合いました");
}else print("条件は合いませんでした");
```

最初に、a==10が真なら、{}内へ入り、b==20を比べています。
a==10が偽の時は、b==20は、比べません。

・break文

この文は、while文の{}内を、強制的に抜け出したい時に、使います。

```
a=0;
while(1){
  if(a==10)break;
  print(a);
  a++;
}
print("終了"+a);
```

breakが呼ばれたとたん、{}内を抜け出します。
コンソールには、0～9が、表示された後、「終了10」と表示されます。

updateメソッドがないので、１コマで処理が終わります。
{}内に、updateメソッドを書くと、１０コマ分、時間がかかってしまいます。

・for文

この文は、while文のように、繰り返し（ループ）ができます。
でも、()内が、while文とは違います。

```
for(a=0;a<10;a++){
  print(a);
}
print("終了");
```

[for文の書き方]
for(初期値;繰返し条件;増加量){}

```
a=0;
while(a<10){
  print(a);
  a++;
}
print("終了"+a);
```

さっきの、プログラムをfor文で、やっています。

まず、a=0;で、aを0にして、a<10を比べて、真だったので、{}内に入りました。
}まできたら、a++をして、a<10を比べて、真だったら、{に戻ります。
a<10が、偽になったら、{}を抜け出します。

左のプログラムを、while文でやったのが、右のプログラムです。左の方が、コンパクトにかけます。

プログラム（new文・appearメソッド・crashToメソッド）

また、for文でも、break文が使えます。
つまり、for文の{}内を抜け出したい時は、breakを使います。

・new文、appearメソッド

new文は、新しいオブジェクトを、実行中に作ります。
例えば、mikataクラスのオブジェクトを、x=50, y=100, p=3, f=1で、作るときは、

```
new mikata(50,100,3,1);
```

書き方は、new クラス名(値,値,値,…);です。()内の書き方は、クラスによって変わります。

・()内の書き方（パネルは、ここでは書きません）（灰色の部分は、省略可能です）

スプライト	(x,y,p,f,zOrder)
ＤＸスプライト	(x,y,p,f,zOrder,scaleX,angle,alpha)
テキスト	(x,y,text,col,size)
シークレット	()

フレームマネージャーは普通、実行中には作りません。

また、以前から紹介してる、６種類のオブジェクトは、appearメソッドが必要です。
もし、mikataクラスが、この６種類中のものだったら、appearメソッドを付け加えます。

```
appear(new mikata(50,100,3,1));
```

このように、appearメソッドの中に、new文を書きます。上の６種類は、appearメソッドがないと、動きません。
６種類以外は、ほとんど、appearメソッドはいりません。

```
a=new mikata(50,100,3,1);
```

または、

```
a=appear(new mikata(50,100,3,1));
```

この場合、aには、mikataオブジェクトの、アドレスが入ります。
なので、この場合は、オブジェクト変数を参照できます。例えば、print(a.x);など。

また、例えば、mikataクラスで、mikataを作ると、ゲームが止まってしまうので、気をつけてください。

・dieメソッド

これは、オブジェクトを消す時に使います。

```
v=0;
while(1){
  if(v==60)die();
  v++;
  update();
}
```

この文では、60コマ（１秒）たったら、オブジェクトが消えます。これは、あの６種類のみ使えます。

・crashToメソッド

これは、自分があるオブジェクトに当たってるか、調べるものです。
例えば、$mikataｵﾌﾞｼﾞｪｸﾄが、$tekiｵﾌﾞｼﾞｪｸﾄに当たってるかどうかを、調べるには、$mikataｵﾌﾞｼﾞｪｸﾄに、

```
crashTo($teki)
```

と書きます。$tekiに当たってる時は1、そうでない時は0を返します。

```
while(1){
  if(crashTo($teki){
    die();
  }
  update();
}
```

応用すると、こんな書き方もできます。
これは、$tekiに当たると、自分が死にます。

Tonyu System

プログラム（getkeyメソッド・マップオブジェクト）

・getkeyメソッド

これは、キーボードや、マウスのボタンが押されてるか、調べるものです。

```
getkey(番号)
```

*番号*のところに、ボタンの番号を入れます。（右の表を参考）

```
print(getkey(32));
```

スペースキーを押していない時は0ですが、
押しつづけると、1から1つずつ、増えていきます。
つまり、１秒間押してると、60が返されます。

```
while(1){
  if(getkey(37))x--;
  if(getkey(38))y--;
  if(getkey(39))x++;
  if(gekkey(40))y++;
  update();
}
```

このように書くと、
カーソルキーでｵﾌﾞｼﾞｪｸﾄが、
移動します。

［キー番号］

1	マウスの左ボタン
2	マウスの右ボタン
4	マウスのホイールボタン
13	エンターキー
16	シフトキー
32	スペースキー
37	カーソルキー←
38	カーソルキー↑
39	カーソルキー→
40	カーソルキー↓
48～57	数字キー（０～９）
65～90	Ａ～Ｚキー
96～105	テンキー(0～9)

・マップオブジェクト

これは、マップ作成すると、Tonyuで自動的に作られます。
オブジェクト名は、「$map」です。「$map.*メソッド名*」と書くと、$mapのメソッドが使えます。

・背景色

背景の色を変えます。

```
$map.setBGcolor(色)
```

・画面移動

画面が、マップ上を移動します。（スクロールという）

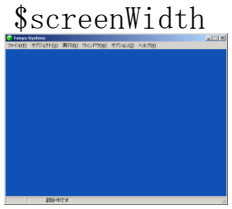
```
$map.scrollTo(横,縦)
```

*横*と*縦*の単位は、ドットです。
$map.scrollTo(10,10);と書くと、画面が、右に10ドット、下に10ドット移動します。

・画面の変数

これを使うと、画面の大きさがわかります。

$screenWidth	画面の横の大きさ
$screenHeight	画面の縦の大きさ



主人公などに、$map.scrollTo(x-$screenWidth/2,y-$screenHeight/2);と書くと、
常に、主人公が真中に表示されます。

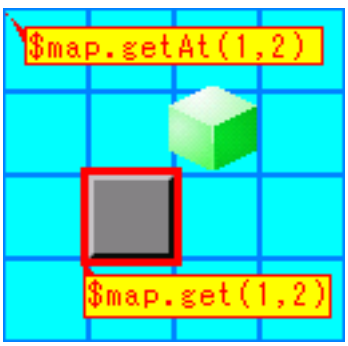
・マスの画像を得る

あるマスの画像を得ます。これには、２つの方法があります。

```
$map.get(横,縦)
```
マス単位で、求める。

```
$map.getAt(横,縦)
```
ドット単位で、求める。

例えば、getメソッドと、getAtメソッドには、右のような
違いがあります。



それぞれの基準は、
get(0,0)は、画面左上の１マス。
getAt(0,0)は、画面左上の１ドット。

・配列

配列とは、変数のように、データを入れれますが、複数入れることができます。
それを使うには、まず、配列オブジェクトを作らなければなりません。

```
変数名=new Array();
```

*変数名*の部分に、好きな変数名を書いてください。
この文では、配列オブジェクト（Arrayｵﾌﾞｼﾞｪｸﾄ）が作られ、そのアドレスが、*変数*に入ります。
配列のメソッドを使うときは、「*変数名.メソッド名*」のように書きます。

これから配列のメソッドを紹介していきます。

・addメソッド

ある要素を追加します。

```
a=new Array();
a.add(123);
a.add("ああ");
a.add("いい");
```

こうすると、右上の[例]のようになります。

[例] 配列a

123
"ああ"
"いい"

・setメソッド

ある要素を代入します。
例えば、さっきのプログラムの下に、

```
a.set(1,"うう");
```

と書くと、配列の中身は、右のようになります。

番号	配列a
0	123
1	"ああ"
2	"いい"

→

番号	配列a
0	123
1	"うう"
2	"いい"

番号は、0から数えます。つまり、このプログラムでは、1番を指しています。
1番の内容が、"ああ"から"うう"に変わりました。
※ 存在しない番号を書くとエラー

・insertメソッド

ある要素を挿入します。

```
a.insert(1,"うう");
```

最初のﾌﾟﾛｸﾞﾗﾑにこう書くと、配列の中身は、右のようになります。
"ああ"の前に、"うう"が挿入されます。
※ この場合、番号は、0～3にしないとエラー

"うう" →

番号	配列a
0	123
1	"うう"
2	"ああ"
3	"いい"

・getメソッド

ある要素を取得します。()内の番号の中身を取得します。

```
t=a.get(2);
```

この場合、tの内容が"いい"になります。
※ 存在しない番号を書くとエラー

番号	配列a
0	123
1	"ああ"
2	"いい"

→ 変数t: "いい"

・indexOfメソッド

ある要素を取得します。()内と同じ内容があれば、その番号を取得します。

```
t=a.indexOf("いい");
```

"いい"があるのは、2番なので、tは2になります。

```
t=a.indexOf("うう");
```

"うう"は、ないので、tは-1になります。
このように、()内の内容が、配列にないと、-1を返します。

番号	配列a
0	123
1	"ああ"
2	"いい"

→ 変数t: 2

```
t2=b.indexOf("ああ");
```

同じ要素があった時は、最小の番号を返します。
右の場合は、1番と2番にあるので、t2は1になります。

番号	配列b
0	123
1	"ああ"
2	"ああ"

→ 変数t: 1

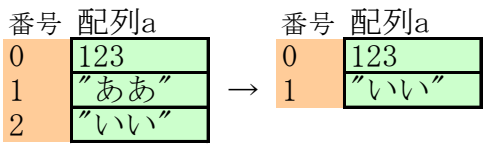
プログラム（配列２）

・deleteメソッド

ある要素を消去します。()内は番号を入れます。

```
a.delete(1);
```

1番を消去します。
番号も変わるので、注意してください。

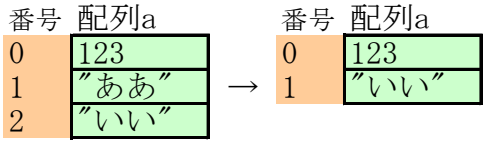


・removeメソッド

ある要素を消去します。()内は内容を入れます。

```
a.remove("ああ");
```

"ああ"は、1番にあるので、1番が消えます。
番号も変わるので、注意してください。

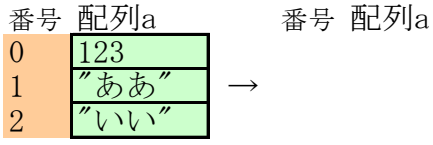


・clearメソッド

全部の要素を消去します。

```
a.clear();
```

配列aの要素が全部消えます。

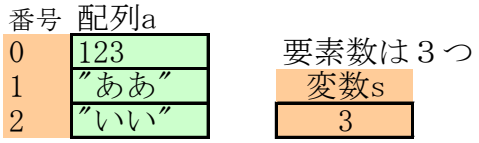


・sizeメソッド

配列の要素数を取得します。

```
s=a.size();
```

配列aの要素数は、3なので、sは3になります。

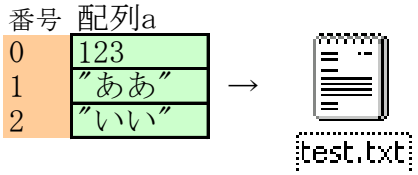


・saveメソッド

配列の内容をセーブ（保存）します。

```
a.save("test.txt");
```

filesフォルダに、testというtxtファイルが作られます。

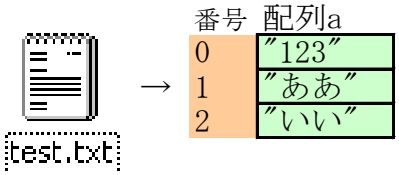


・loadメソッド

あるファイルから、配列の内容をロード（読み込み）します。

```
a.load("test.txt");
```

上で、test.txtを作った後に、ロードすると、
配列の内容が、元に戻ります。
しかし、全て文字になってしまいます。

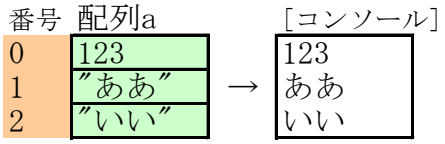


・for(*変数名* in *配列名*){}

for文は、もう１つ書き方があって、これは、配列専用の命令です。（メソッドではない）

```
for(t in a){
  print(t);
}
```

こう書くと、コンソールに配列の内容が表示されます。



動きは、下と同じです。

```
for(i=0;i<a.size();i++){
  t=a.get(i);
  print(t);
}
```

配列の内容を、１つ１つ、tに入れています。
配列の中身が知りたいときなどに使います。
要素数が変わっても、大丈夫です。

```
for(t in $chars){
  if(t is teki||crashTo(t))die();
}
```

これは、tekiクラスのｵﾌﾞｼﾞｪｸﾄに、当たってるかどうかです。
$charsは配列で、全ｵﾌﾞｼﾞｪｸﾄのアドレスが入っています。
is文は、tのアドレスのｵﾌﾞｼﾞｪｸﾄが、tekiクラスかどうか調べてて、条件が合えば真になります。

・描画系メソッド

これらは、ｵﾌﾞｼﾞｪｸﾄを作らないで、画像などを表示したい時に使います。
ただし、毎回命令を出さないと、消えてしまいます。
スプライト・ＤＸスプライト・テキストなどは、
Tonyuが自動的に、これらのメソッドを使って表示しています。
でも、それだけだと、足りない時は、これらのメソッドを使います。

- drawSprite(*横*, *縦*, *画像番号*, *反転*, *表示順位*)
  スプライト表示します。スプライトと同じ表示ができます。

- drawDXSprite(*横*, *縦*, *画像番号*, *反転*, *表示順位*, *回転*, *透明度*, *拡大率X*, *拡大率Y*)
  ＤＸスプライト表示します。ＤＸスプライトと同じ表示ができます。

- drawText(*横*, *縦*, *文字*, *色*, *サイズ*, *表示順位*)
  テキスト表示します。テキストと同じ表示ができます。

- centerText(*横*, *縦*, *文字*, *色*, *サイズ*, *表示順位*)
  テキスト表示します。これは、*横*で指定した座標を中心に、テキストが表示されます。
  *縦*は、drawTextメソッドと変わりません。

- drawLine(*横1*, *縦1*, *横2*, *縦2*, *色*, *表示順位*)
  線を表示します。
  横1・*縦1*で指定した座標から、*横2*・*縦2*で指定した座標まで、線を引きます。

- drawRect(*横1*, *縦1*, *横2*, *縦2*, *色*, *表示順位*)
  枠を表示します。
  横1・*縦1*で指定した座標から、*横2*・*縦2*で指定した座標まで、枠を表示します。

- fillRect(*横1*, *縦1*, *横2*, *縦2*, *色*, *表示順位*)
  長方形を表示します。
  横1・*縦1*で指定した座標から、*横2*・*縦2*で指定した座標まで、長方形を表示します。

・色を作るメソッド・変数

色は、表示系メソッドで、色とかかれたところに、メソッドや変数を書きます。

```
drawText(0,0,"色の実験",color(255,0,0),20);
drawText(0,0,"色の実験",$clRed,20);
```

この場合は、どちらも、赤が表示されます。
色には、ただの数字が使われています。0～16777215の数字で表されています。

```
drawText(0,0,"色の実験",16777215,20);
```

こういう書き方も、できます。これは、白になります。

- color(*赤*, *緑*, *青*)
  この３つの引数は、0～255の数字を入れます。そして、それに対する数値が返されます。

- colorHSL(*色相*, *彩度*, *明度*)
  これも、色を作るメソッドです。colorメソッドと、作り方が違います。
  色相　　：色合いです。0～239で指定。
  彩度　　：カラー度です。0～240で指定。
  明度　　：明るさです。0～240で指定。

プログラム（計算系・音・ページのメソッド）

・計算系メソッド

計算は、演算子(+ - * / …)でできました。でも、ほかの事をしたい時は、これらのメソッドを使います。

・rnd(*範囲*)

ランダムな整数を返します。*範囲*には、整数を入れます。
例えば、rnd(5);と書くと、0～4の値のどれかを返します。
*範囲*は省略可能ですが、普通は、省略しません。

・randomize(*番号*)

ランダムの順番を設定します。*番号*は、0～　　の数字を入れます。
*番号*を省略すると、毎回変わった順番になります。
これで、rndメソッドが返す数値が変わります。

・abs(*数値*)

絶対値を返します。*数値*に入れた整数は、すべて＋の数で返ってきます。
つまり、－の値を入れても、＋になって返ってきます。

・trunc(*実数*)

小数点を切り捨てます。これは、小数点より小さい数字を切り捨てます。

・floor(*実数*)

小数点を切り捨てます。これは、*実数*の値を超えないように切捨てします。

・valueOf(*文字*)

*文字*を数字にして、返します。
例えば、*文字*に"123"を入れると、数字の123を返します。

実数	trunc	floor
-2	-2	-2
-1.5	-1	-2
-1	-1	-1
-0.5	0	-1
0	0	0
0.5	0	0
1	1	1
1.5	1	1
2	2	2

・音楽のメソッド

音楽や効果音を鳴らすには、$mplayerオブジェクトにお願いします。
つまり、$mplayerのメソッドを使います。

・$mplayer.play(*曲*,*繰返し*,*音量*)

音楽の演奏や、効果音を鳴らします。
*曲*には、音リストに書いてある、$se_…を書きます。鳴らしたい音を入れます。
*繰り返し*には、音楽を繰り返したい時に、1を入れます。それ以外は0です。
*音量*には、効果音の音量を0～128の数字で入れます。音楽の音量は設定できません。

・$mplayer.stop()

音楽を止めます。

・ページのメソッド

ページのメソッドは、$projectManagerオブジェクトのメソッドを使います。

・$projectManager.loadPage(*ページ名*)

ページを移動します。*ページ名*には、ページリストの、$page_…を書きます。

・$projectManager.getCurrenPageName()

今のページ名を返します。文字で返します。例（"$page_index"）