# 目的

・Connect STG のプログラムの打ち方と実行の仕方
・ゲーム画面の座標を学ぶ
・条件分岐を学ぶ(if 文)
・自機を移動させる

ぬぬぬ。ゲームを作るって、ゴチャゴチャしていて難しいっ!
どっぐ博士・・・もっと、わかり易い物ないの!?!?

**知ろうと・ぱんだ　君**

ぱんだ君は、プログラミングに挑戦してるんだね。それなら、今日は、シューティングゲーム（STG）を使ってプログラミングを勉強しよう。

**ど偉い・どっぐ　博士**

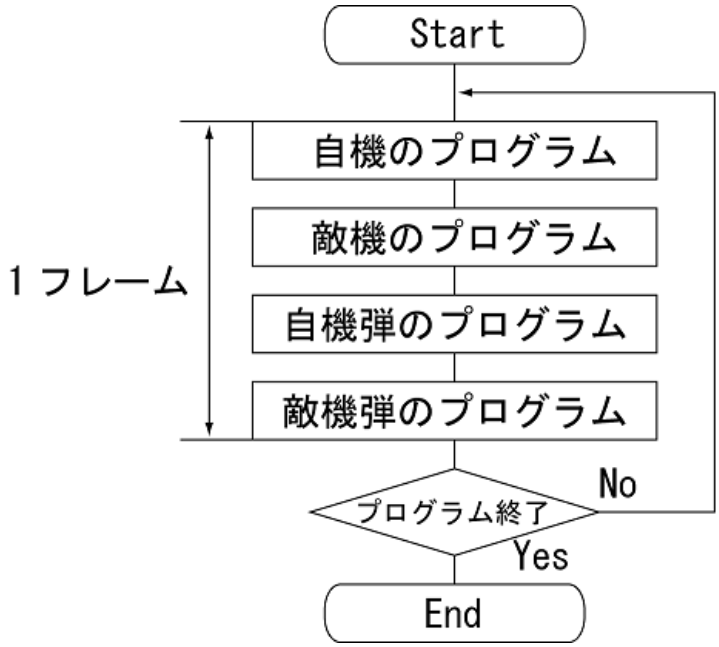
STG は、自機や敵機、そしてたくさんの弾幕が魅力だよね。
おいらも遊ぶのは好きだけど、作るのは難しいんじゃないの？

ぱんだ君の言う通り、キャラクターとしては、自機、敵機、自機弾、敵機弾が主になるよね。
この STG に特化したプログラミング環境 Connect STG を使えば、キャラクター毎にプログラムを書き分けることができるので、それぞれの動きが理解しやすくなるよ。

つまり魅力的な STG を作りながら、プログラミングの理解できるってことなのか（半信半疑）。

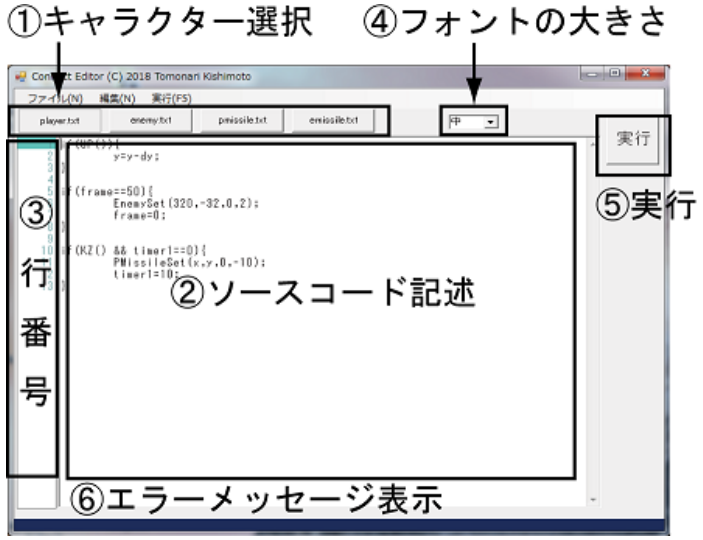
# 1． Connect STG とは？

Connect STG は、STG に特化したプログラミング学習環境です。STG のキャラクターである自機、敵機、自機弾、敵機弾にプログラムを記述し、１秒間に 60 フレーム（60 回）で動作します。１フレームのプログラムの流れ図を示します。




# 2．エディタの使い方

①キャラクターの選択（自機、敵機、自機弾、敵機弾）
②ソースコード（プログラム）の記述
③行番号の表示
④フォントの大きさの調整（小さい場合は大きくできる）
⑤プログラムの実行（プログラムは自動セーブされる）
⑥構文エラーの場合、その付近のエラーが表示される



# 3．　実行画面と自機の移動

横が x 軸、縦が y 軸となります。

y軸は下に行くほど数値が大きくなることに注意


自機を上に移動させるには、どうすればいいの？

自機を、少し上の位置に表示すればいいんだよ。
表示位置は、プログラムの変数 x,y の値に対応している。

差はいくつ？

自機を今の位置から少し上の位置を考えてみよう

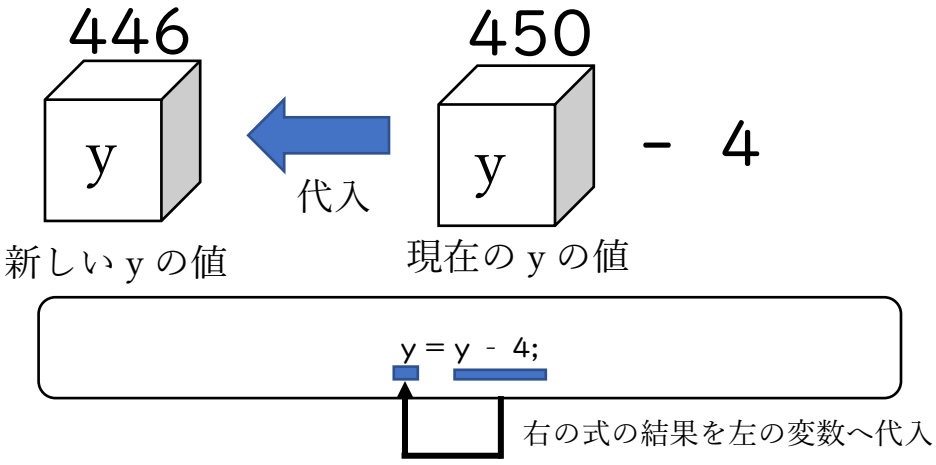
今　x 軸：320　y 軸：450

次　x 軸：320　y 軸：446

x ：

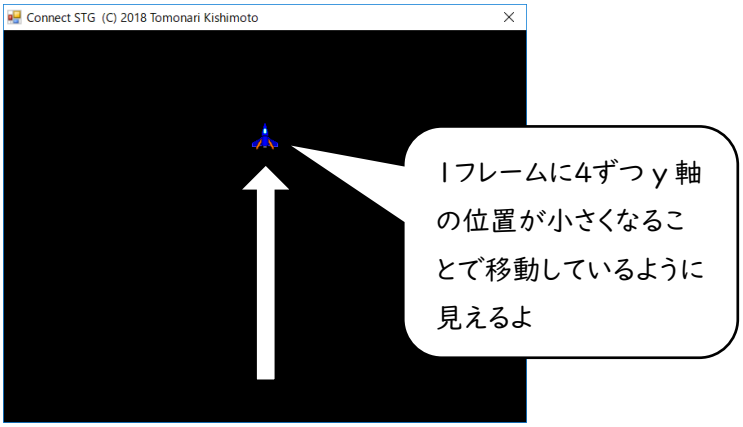
y ：

次の位置との差（変化）がわかれば、プログラムには、次のように位置が変化する様子を記入するよ。




Player.txt に記述

| | |
|---|---|
| y = y − 4; | **y 軸の値（変数 y）を 4 小さくして代入する。（値が更新される）** |



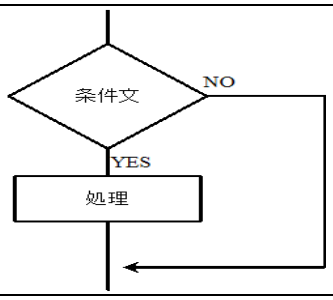

## 4． 自分で操作する

どっぐ博士!!
これ自分で動かせたら、マジ面白い!!

ぱんだ君は、いい所に気が付くね!
もちろん、条件分岐とボタンの判断の命令を使えば動くよ。

# 条件分岐とは？

次の条件分岐を使えば、条件文が真（Yes,true）の時だけ命令を実行できるよ。

| | | |
|---|---|---|
| if( 条件文 ){<br>　　処理;<br>} |  | 条件文に書かれたものが Yes の場合、{ }でくくられた処理を行う。 |

# 条件文に入れるものは？

変数や数値を比較することができます。
ボタンを押しているかを調べる特別な命令も使えるよ。

一般的なもの（比較命令）

| | |
|---|---|
| a < b | a は b より小さい |
| a > b | a は b より大きい |
| a >= b | a は b より小さいか等しい |
| a <= b | a は b より大きいか等しい |
| a == b | a と b は等しい |
| a != b | a と b は等しくない |

ボタンを押しているかを調べる特別な命令

| | |
|---|---|
| UP() | 上キーを押しているか |
| DOWN() | 下キーを押しているか |
| LEFT() | 左キーを押しているか |
| RIGHT() | 右キーを押しているか |
| KZ() | Z キーを押しているか |
| KX() | X キーを押しているか |
| KC() | C キーを押しているか |
| KV() | V キーを押しているか |

上下左右は、英語に直しただけだね。
Z,X,C,V キーは、前に K を付けるだけなので覚えやすい。

では、条件分岐である if 文、これの条件文の中に「上キーを押しているか」をいれてプログラムを作ってみよう。

Player.txt に変更記述

| | |
|---|---|
| if(UP()){ | 上キーを押した時 |
| y=y−4; | y 軸の値（変数 y）を 4 小さくして代入する |
| } | |

Tab キーでインデント（字下げ）すると見やすい

さっきのプログラムに、if 文を追記するだけで動くなんて･･･
プログラムって命令の組み合わせでできているんだ!!

**課題**

残り DOWN(),LEFT(),RIGHT()についても同様の処理を行いましょう。ただし、 x 軸は、変数 x を用いましょう。

UP 用のプログラムをコピーし、貼り付けを行うと、早くプログラムが作れるよ。

## 5．変数 x,y に増減させる数値を変更する

じゃぁ、ぱんだ君････。
そろそろ、自機の速さを変えてみようか。

どうすればいいかわかりません。

プログラムは、キャラクターの 1 フレームの変化を記述しているんだ。
「小さい変化」と「大きい変化」だと、同じ時間経過した場合、どちらが多く進むかな??

「大きい変化」の方が、多く進みそう････
･･･そうか･･･それって、速く移動するってことなのか!!

Player.txt に変更記述

```
if(UP()){
        y=y-8;
}
if(DOWN()){
        y=y+8;
}
if(LEFT()){
        x=x-8;
}
if(RIGHT()){
        x=x+8;
}
```

数値を書き換えて、速さが変わるのを体感しよう。

# 6．移動量を変数で制御しよう

「4」という数字を「8」に変更すると速さが変わった。
この数値を自由に変化させれば速さが制御できそう

自由に変化させるために、その部分を変数として取り扱おう。
y軸は「dy」、x軸は「dx」として、プログラムを書いてみるよ。
ちなみに、dx，dy 共に「4」の値が初期値として設定されているよ

Player.txt に変更記述

```
if(UP()){
        y=y-dy;
}
if(DOWN()){
        y=y+dy;
}
if(LEFT()){
        x=x-dx;
}
if(RIGHT()){
        x=x+dx;
}
```

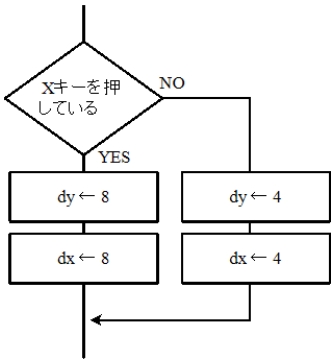
変更

| |
|---|
| 上キーを押した時 |
| y 軸の値（変数 y）を dy（4）小さくして代入 |
| する |
| 下キーを押した時 |
| y 軸の値（変数 y）を dy（4）大きくして代入 |
| する |
| 左キーを押した時 |
| x 軸の値（変数 x）を dx（4）小さくして代入 |
| する |
| 右キーを押した時 |
| x 軸の値（変数 x）を dx（4）大きくして代入 |
| する |

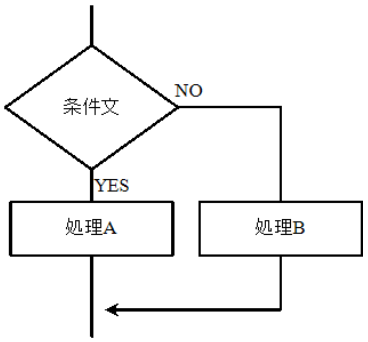
速さを制御するには、次のプログラムを考えてみよう。

X キーを押した時　→　8 移動する
X キー押していない時　→　4 移動する

X キーを押しているかを if 文で判断し
dx,dy に 8 を入れるか、4 をいれるか決める





| | | |
|---|---|---|
| if(条件文){<br>　　処理 A;<br>}<br>else{<br>　　処理 B;<br>} |  | 条件文に書かれたものが Yes の場合、{}でくくられた処理 A を行う。<br>そうでない場合、{}でくくられた処理 B を行う。 |

if 文はYesの処理を書く。
Noの処理は、else を書けば追加機能として加えられる。

Player.txt に変更記述

| | |
|---|---|
| if( KX() ){ | X キーを押した時 |
| dy = 8; | dy を 8 にする |
| dx = 8; | dx を 8 にする |
| } | |
| else{ | X キーを押していない時 |
| dy = 4; | dy を 4 にする |
| dx = 4; | dx を 4 にする |
| } | |
| if( UP() ){ | 上キーを押した時 |
| y = y − dy; | y 軸の値（変数 y）を dy 小さくして更新する |
| } | |
| ・・・(以下省略) | |

# 課題

次の状態で自機が移動するようにプログラムを改造しよう

| | | |
|---|---|---|
| X キーを押す | x 軸の変化量:4 | y 軸の変化量:8 |
| X キーを押さない | x 軸の変化量:8 | y 軸の変化量:4 |

長かったけど、使い方と自機の移動は、これで終わりです。
次は、自機から弾を出す方法です。