

# ネットワークゲームを作ろう

ネットワーク機能を使って、教室の中で友だちとゲームをするプログラムを作ってみよう。題材は、ch 01 pingpong で作ったピンポンゲームを対戦型に拡張したものである。

作成したゲームの画面を示す。画面は左側と右側のプログラムで異なるが、内容はほとんど同じであるため、左側のプログラムを例に説明する。ピンポンゲームと変わらない部分については、ch 01 pingpong を参照してほしい。



## 壁を作る（ステップ1）

最初に、ゲームに登場するオブジェクトを画面に置く。上下にある長方形は、ボールを跳ね返す壁である。左の小さいほうの長方形はボールを打ち返すパドルである。大きいほうの長方形は、パドルでボールを打ち返せなかったことを知るための壁である。ボールがこの壁にぶつくと、ゲームオーバーになる。



次のプログラムでは、4個の長方形を作成している。プログラムを簡単にするために、線の太さで太さ20の線を描くことで長方形とした。続いて、上壁を作ってから複製して下壁を作り、続いて左壁とパドルを作っている。

```
// 壁を作る（ステップ1）
かめた = タートル！作る。
かめた！（緑）線の色 20 線の太さ。
壁 = かめた！550 歩く 図形を作る -200 200 位置。
壁！作る -200 -220 位置。
かめた！90 左回り。
左壁 = かめた！（黄）線の色 440 歩く 図形を作る -210 -230 位置。
パドル = かめた！（青）線の色 120 歩く 図形を作る -190 -210 位置。
```

## パドルを動かす（ステップ2）

画面にボタンを表示して、パドルを上下に動かせるようにする。次のプログラムでは、画面の左側に「上ボタン」と「下ボタン」という名前の2つのボタンを表示し、ボタンが押されるか上下の矢印キーが押されたときにパドルを上下に50ずつ移動させている。



```
// パドルを動かす（ステップ2）
上ボタン = ボタン！"上" "UP" 作る -380 50 位置。
下ボタン = ボタン！"下" "DOWN" 作る -380 0 位置。
上ボタン：動作 = 「パドル！0 50 移動する」。
下ボタン：動作 = 「パドル！0 -50 移動する」。
```

## ボールの動きを定義する (ステップ3)

「かめた」はボールの役割をする。毎回少しずつ違った位置からゲームが始まるように、横の位置は左右の中央で、縦の位置は乱数を使い、-149 ~ 150の範囲でランダムになるようにした。壁にぶつかると、自然な角度で跳ね返る。

```
// ボールの動きを定義する (ステップ3)
かめた！ペンなし 45 向き。
かめた！0 (乱数 (300) -150) 位置。
かめた：衝突 = タートル：跳ね返る。
```



## 通信を準備する (ステップ4)

ゲームをする2つの画面の間では、ボールがどのような状態にあるかという情報を、お互いに交換して合わせておく必要がある。たとえば、ひとつの画面でパドルと衝突して跳ね返ったら、もうひとつの画面でも同様に向きを変える必要がある。

そこで、自分のボールの状態をサーバーに書き込む「書き込み」というメソッドと、相手のボールの状態をサーバーから読み出す「読み出し」というメソッドを定義することにした。

次のプログラムで、「書き込み」の中では、現在のボール（「かめた」）から位置と向きを取り出し、それぞれ `x` `y` `t` という名前でサーバーに保存している。「読み出し」の中では、サーバーから位置と向きの情報 `x` `y` `t` を取り出し、ボール（「かめた」）にセットしている。

ステップ4からステップ6までを実行するには、サーバーと相手の画面が必要になる。詳しくはステップ7で説明する。

```
// 通信を準備する (ステップ4)
かめた：書き込み = 「
  サーバ "x" [かめた！横の位置?] 書く。
  サーバ "y" [かめた！縦の位置?] 書く。
  サーバ "t" [かめた！向き?] 書く
」
かめた：読み出し = 「
  x = サーバ "x" 読む。
  y = サーバ "y" 読む。
  t = サーバ "t" 読む。
  かめた x y 位置 t 向き
」
```

## 通信しながらボールを動かす (ステップ5)

ステップ5は、このプログラムの中心的な部分である。行っている処理は画面上のボールを動かすことだが、ボールは両方の画面で同じように表示される必要があるため、サーバーを介した通信を行うことで、ボールの状態をお互いに合わせている。

次のプログラムでは、最初に、サーバーに接続する。続いて、ステップ4で定義した「書き込み」を実行して、ボールの位置をサーバーに書き込む。そしてタイマーを作成する。ここまでが、プログラムを

開始するときの動作である。

続いて、ゲームの処理を行う。本質的な動作は次の1行で記述できる。つまり、タイマーでボール役の「かめた」を前進させる動作である。

```
時計！「かめた！10 歩く」実行。
```

ここでは、他の画面とボールの状態を合わせるために、最初に「読み出し」でサーバーから相手のボールの状態を取得して、ボールを正しい位置に置く。続いてボールを移動し、移動後の新しい状態を「書き込み」でサーバーに書き込んでいる。

```
// 通信しながらボールを動かす（ステップ5）
サーバ□□"localhost" 接続。
かめた！書き込み。
時計 = タイマー！作る 60秒 時間 0.2秒 間隔。
時計！「
    かめた！読み出し。
    かめた！10 歩く。
    かめた！書き込み。
」実行。
```

## ゲームの勝敗を判定する（ステップ6）

最後に、タイマーが終了するまでゲームを続けられた場合には「ゲームクリア」というメッセージを表示する。この部分はピンポンゲームと同じであるため、詳しくはch□pingpong を参照されたい。

```
// ゲームの勝敗を判定する（ステップ6）
ゲームクリア = はい。
左壁：衝突 = 「：ゲームクリア = いいえ。時計！中断」。
時計！「
    「ゲームクリア = はい」！なら「
        ラベル！"ゲームクリア！"作る（青）文字色。
    」そうでなければ「
        ラベル！"ゲームオーバー！"作る（赤）文字色。
    」実行。
」最後に実行。
```

ステップ1からステップ6までの全体のプログラムを掲載しておく。これは左側の画面で実行するプログラムである。

```
// 壁を作る（ステップ1）
かめた = タートル！作る。
かめた！（緑）線の色 20 線の太さ。
壁 = かめた！550 歩く 図形を作る -200 200 位置。
壁！作る -200 -220 位置。
かめた！90 左回り。
左壁 = かめた！（黄）線の色 440 歩く 図形を作る -210 -230 位置。
パドル = かめた！（青）線の色 120 歩く 図形を作る -190 -210 位置。

// パドルを動かす（ステップ2）
上ボタン = ボタン！"上" "UP" 作る -380 50 位置。
下ボタン = ボタン！"下" "DOWN" 作る -380 0 位置。
上ボタン：動作 = 「パドル！0 50 移動する」。
下ボタン：動作 = 「パドル！0 -50 移動する」。
// (続き)
```

```
// ボールの動きを定義する (ステップ3)
かめた！ペンなし 45 向き。
かめた！0 (乱数 (300) -150) 位置。
かめた：衝突 = タートル：跳ね返る。

// 通信を準備する (ステップ4)
かめた：書き込み = 「
  サーバ[] [] "x" []かめた！横の位置?) 書く。
  サーバ[] [] "y" []かめた！縦の位置?) 書く。
  サーバ[] [] "t" []かめた！向き?) 書く
[] []
かめた：読み出し = 「
[]x=サーバ[] [] "x" 読む。
[]y[]サーバ[] [] "y" 読む。
[]t[]サーバ[] [] "t" 読む。
  かめた[] []x[] []y[] 位置[]t[]向き
[] []

// 通信しながらボールを動かす (ステップ5)
サーバ[] [] "localhost" 接続。
かめた！書き込み。
時計 = タイマー！作る 60秒 時間 0.2秒 間隔。
時計！「
  かめた！読み出し。
  かめた！10 歩く。
  かめた！書き込み。
」実行。

// ゲームの勝敗を判定する (ステップ6)
ゲームクリア = はい。
左壁：衝突 = 「：ゲームクリア = いいえ。時計！中断」。
時計！「
  「ゲームクリア = はい」！なら「
    ラベル！"ゲームクリア！"作る (青) 文字色。
  」そうでなければ「
    ラベル！"ゲームオーバー！"作る (赤) 文字色。
  」実行。
」最後に実行。
```

## プログラムを実行する (ステップ7)

この節で作成しているプログラムは、サーバーや相手の画面と協調して動作するため、単体では動かない。実行するための手順を説明する。

まず、サーバーを起動する。サーバーはどのコンピュータで起動してもよいが、ここでは左側の役割をする画面を実行するコンピュータで動かすことにする。サーバーを起動するには、ドリトルの編集画面 (プログラムを入力する画面) を開き、右側にある **server** をマウスでチェックする。すると、現在使用中のPCのIPアドレス、および、サーバに送られたデータの情報が、画面右側に表示される。

次に、右側の画面を実行するために、もうひとつドリトルの画面を起動する。右側の画面で実行するプログラムを示す。左側の画面との違いは、画面上のパドルなどが左右逆に配置されていることと、ボールの初期位置をサーバーに設定しないこと、そしてボールが両方の画面で自然な位置に表示されるように、ボールの位置を100だけ左にずらしてサーバーに書き込んでいることである。

このプログラムを左側の画面と同じコンピュータで実行する場合は変更の必要はないが、別のコンピュー

タで実行する場合には、左側の画面のコンピュータを実行しているコンピュータのホスト名またはIPアドレスを調べて、プログラム中の「localhost」の部分を書き換える必要がある。

```
// 壁を作る (ステップ1)
かめた = タートル! 作る。
かめた! (緑)線の色 20 線の太さ 180 右回り。
壁 = かめた! 550 歩く 図形を作る 200 200 位置。
壁! 作る 200 -220 位置。
かめた! 90 右回り。
左壁 = かめた! (黄)線の色 440 歩く 図形を作る 210 -230 位置。
パドル = かめた! (青)線の色 120 歩く 図形を作る 190 -210 位置。
// (続き)
// パドルを動かす (ステップ2)
上ボタン = ボタン! "上" "UP" 作る 230 50 位置。
下ボタン = ボタン! "下" "DOWN" 作る 230 0 位置。
上ボタン: 動作 = 「パドル! 0 50 移動する」。
下ボタン: 動作 = 「パドル! 0 -50 移動する」。

// ボールの動きを定義する (ステップ3)
かめた! ペンなし。
かめた! 0 (乱数 (300) -150) 位置。
かめた! 45 向き。
かめた: 衝突 = タートル: 跳ね返る。

// 通信を準備する (ステップ4)
かめた: 書き込み = 「
  サーバ[] [] "x" [] 100+[] かめた! 横の位置?) ) 書く。
  サーバ[] [] "y" [] かめた! 縦の位置?) 書く。
  サーバ[] [] "t" [] かめた! 向き?) 書く
[]
かめた: 読み出し = 「
[] x = サーバ[] [] "x" 読む。
[] y [] サーバ[] [] "y" 読む。
[] t [] サーバ[] [] "t" 読む。
  かめた [] [] x - 100 [] [] y [] 位置 [] t [] 向き
[]

// 通信しながらボールを動かす (ステップ5)
サーバ[] [] "localhost" 接続。
// かめた! 書き込み。
時計 = タイマー! 作る 60秒 時間 0.2秒 間隔。
時計! 「
  かめた! 読み出し。
  かめた! 10 歩く。
  かめた! 書き込み。
」実行。

// ゲームの勝敗を判定する (ステップ6)
ゲームクリア = はい。
左壁: 衝突 = 「: ゲームクリア = いいえ。時計! 中断」。
時計! 「
  「ゲームクリア = はい」! なら 「
    ラベル! "ゲームクリア!" 作る (青) 文字色。
  」そうでなければ 「
    ラベル! "ゲームオーバー!" 作る (赤) 文字色。
  」実行。
```

」最後に実行。

From:

<https://dolittle.eplang.jp/> - プログラミング言語「ドリトル」

Permanent link:

[https://dolittle.eplang.jp/ch\\_pingpong331?rev=1580458044](https://dolittle.eplang.jp/ch_pingpong331?rev=1580458044)

Last update: **2020/01/31 17:07**

