Hyper Pictogramming

（ハイパーピクトグラミング）

（2025 年 11 月 28 日版）

第１章　はじめに

ハイパーピクトグラミングはピクトグラムの作成を通じて，Web ページの作成技法について学ぶことができるアプリケーションです．

1.1　ピクトグラム

ピクトグラムとは日本語で絵記号，図記号と呼ばれるグラフィックシンボルで，意味するものの形状を使ってその意味概念を理解させる記号です．ピクトグラムは案内，安全，施設，機器等々，様々な用途で標準化されています．

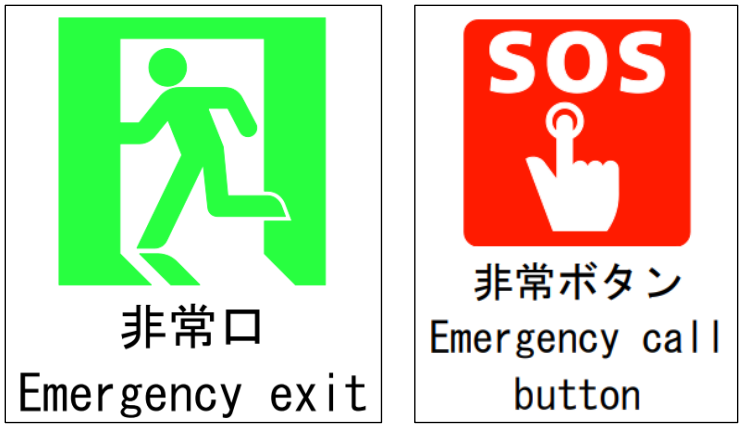


図　ピクトグラムの例（国土交通省ホームページより引用）

ピクトグラムは，世界共通の記号表現として世界中で用いられていますが，特に近年のグローバル化やその流れに伴う外国人観光客の急激な増加などの理由もあり，ピクトグラムを題材とする研究が盛んになっています．

1.2　HTML・CSS・JavaScript

Web ページを作成するには，HTML (HyperText Markup Language)，CSS (Cascading Style Sheets)，JavaScript という技術が使用されています．HTML は Web ページの構造を定義するマークアップ言語，CSS は Web ページの見た目を指定するスタイルシート言語，JavaScript は Web ページの動きを指定するプログラミング言語です．ハイパーピクトグラミングで，ピクトグラムを作りながら，楽しく学んでいきましょう．

第 2 章　アクセスと画面説明

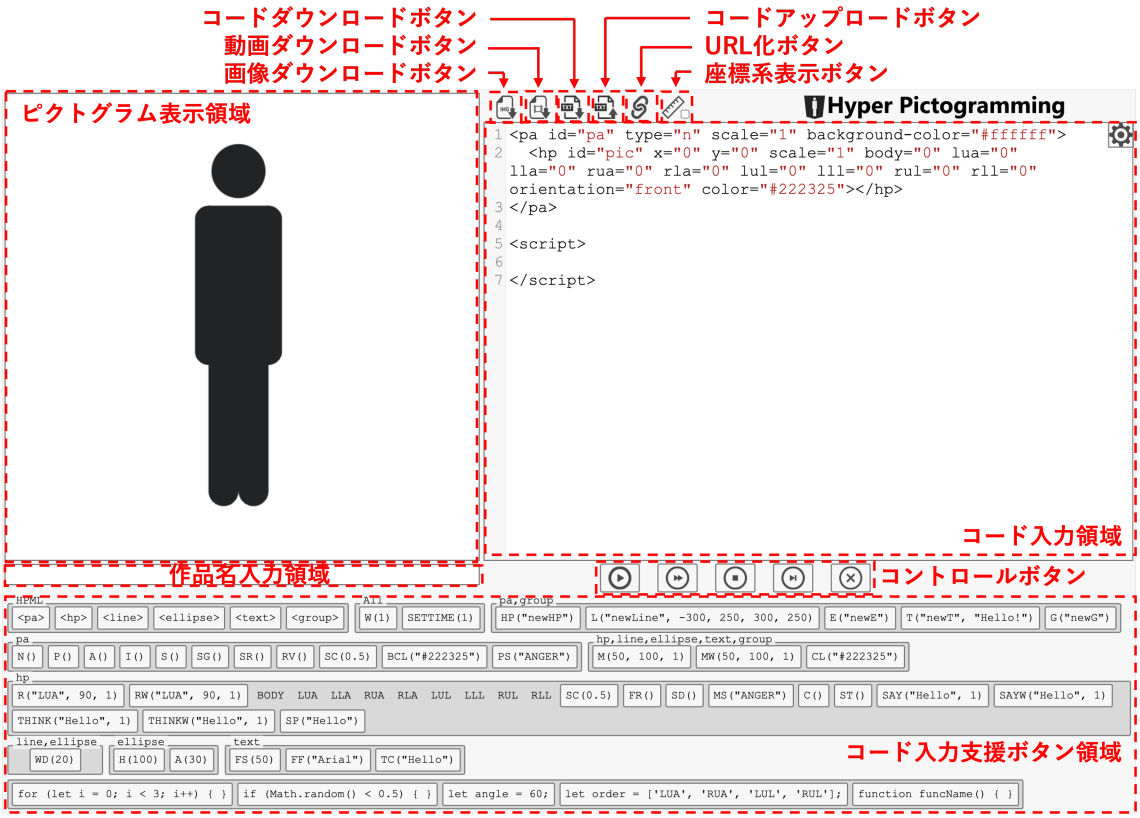
2.1　アクセス

以下の URL からハイパーピクトグラミングにアクセスしてください.

https://pictogramming.org/apps/hyperpictogramming/

または，検索エンジンで「ピクトグラミング」と検索していただき，ピクトグラミングシリーズのホームページにあるハイパーピクトグラミングの「はじめる」ボタンからアクセスすることも可能です.

2.2　画面説明



画面は主に 3 つの部分から構成されています．上左側は，コードの実行結果を表示するピクトグラム表示領域，上右側はコードを入力するコード入力領域，下側にはコードの入力を支援するコード入力支援ボタン領域が配置されています．

画面右中央部には，5 つのコントロールボタンがあります．

画像	名称	説明
	実行ボタン	コード入力領域のコードをピクトグラム表示領域で実行する．
	実行ボタン（早送り）	コード入力領域のコードをピクトグラム表示領域で，早送りで実行する．
	一時停止ボタン	一時停止する．
	再開ボタン	一時停止状態から再開する．
	クリアボタン	コード入力領域に入力されているプログラムを全て消去する．

また，画面右上部には 6 つの機能を使用するためのボタンがあります．

画像	名称	説明
	画像ダウンロードボタン	ピクトグラム表示領域を画像としてダウンロードする．
	動画ダウンロードボタン	ピクトグラム表示領域をアニメーションとしてダウンロードする．
	コードダウンロードボタン	コード入力領域のコードをダウンロードする．
	コードアップロードボタン	コード入力領域にコードをアップロードする．
	URL 化ボタン	コード入力領域と作品名入力領域の情報を含んだ URL をクリップボードにコピーする．
	座標系表示ボタン	ピクトグラム表示領域に座標系を表示する．

第 3 章　HPML

3.1　HTML とは

HTML (HyperText Markup Language)は要素を組み合わせることで Web ページの構造を定義するマークアップ言語です．以下に例として，リストを表示するための要素とその表示結果を示します．

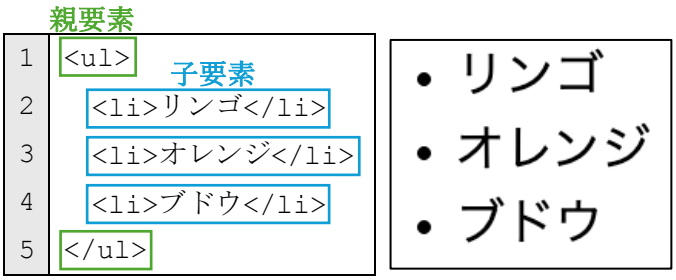


図　（左）HTML の例，（右）表示結果

要素は「<要素名>要素のコンテンツ</要素名>」というように記述されます．「<要素名>」の部分を開始タグ，「</要素名>」の部分を終了タグと言います．要素のコンテンツには，さらに要素を入れることができます．上記の例では，unordered list（順序なしリスト）を意味する `ul` 要素の中に，list item（リスト項目）を意味する `li` 要素が入っています．なお，このように `ul` 要素の中に `li` 要素を記述したとき，`ul` 要素を親要素，`li` 要素を子要素といいます．

上記の `ul` 要素や `li` 要素以外にも要素は用途に合わせて多くの量があり，初めて HTML を学習する際には混乱してしまいます．そこで，ハイパーピクトグラミングではピクトグラムを作成するための HPML という独自のマークアップ言語を定義しています．HPML について学んでいきましょう．

3.2　HPML (Hyper Pictogram Markup Language)

HPML とは Hyper Pictogram Markup Language の略で，ピクトグラムを作成するためのハイパーピクトグラミング独自のマークアップ言語です．ハイパーピクトグラミングにアクセスすると，コード入力領域には最初から以下のコードが入力されています．これは初期状態の人型ピクトグラムを表示するためのコードです．

```
<pa id="pa" type="n" scale="1" background-color="#ffffff">
  <hp id="pic" x="0" y="0" scale="1" body="0" lua="0" lla="0" rua="0" rla="0" lul="0" lll="0" rul="0" rll="0" color="#222325" orientation="front"></hp>
</pa>

<script>

</script>
```

「id="pa"」や「id="pic"」,「<script> </script>」に関しては，第 5 章で使用します．そのため，それまでは削除していただいても大丈夫です．

このコードは pa 要素と hp 要素で構成されています．pa は Pictogram Area（ピクトグラムエリア）を，hp は Human Pictogram（人型ピクトグラム）を意味しています．
「type="n"」といった記述は後ほど学習するため，一旦無視してください．

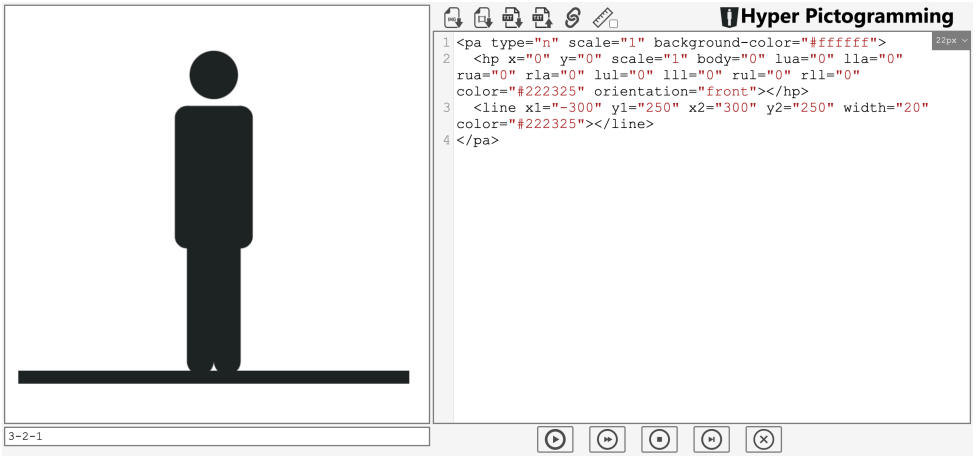
人型ピクトグラムを表示するためには，まず pa 要素を親要素として記述し，ピクトグラムエリアを設置します．そして，pa 要素の子要素に hp 要素を記述することで，人型ピクトグラムを表示させることができます．hp 要素だけで人型ピクトグラムを表示させることはできず，pa 要素が親要素である必要があります．

ハイパーピクトグラミングには pa 要素と hp 要素以外にも，line 要素，ellipse 要素，text 要素，group 要素があります．line 要素は線，ellipse 要素は楕円，text 要素はテキストを描画するためのものです．group 要素は子要素をまとめてグループ化するためのもので，group 要素の子要素にさらに group 要素を入れることも可能です．

次は，hp 要素の下に line 要素を入力してみましょう．なお，hp 要素の下にカーソルを合わせ，画面右下のコード入力支援ボタン領域の「<line>」ボタンを押すと簡単にコードを入力できます．line 要素を入力できたら，実行ボタンを押してください．

コード例 3-2-1（←クリックするとハイパーピクトグラミングでコード例を開きます）

```
<pa type="n" scale="1" background-color="#ffffff">
  <hp x="0" y="0" scale="1" body="0" lua="0" lla="0" rua="0" rla="0" lul="0" lll="0" rul="0" rll="0" color="#222325" orientation="front"></hp>
  <line x1="-300" y1="250" x2="300" y2="250" width="20" color="#222325"></line>
</pa>
```



すると，上の図のように線が描画されたと思います．ですが，人型ピクトグラムや線が初期状態のままです．そこで，次は属性を使い，ピクトグラムを変化させていきましょう．

表　HPML タグ要素集合の一覧

タグ要素名	処理
pa	Pictogram Area（ピクトグラムエリア）を設置する．
hp	pa 要素，または group 要素の子要素にすることで，Human Pictogram（人型ピクトグラム）を表示する．
line	pa 要素，または group 要素の子要素にすることで，線を描画する．
ellipse	pa 要素，または group 要素の子要素にすることで，楕円を描画する．
text	pa 要素，または group 要素の子要素にすることで，テキストを描画する．
group	pa 要素，または group 要素の子要素にし，子要素の hp 要素や line 要素，ellipse 要素，text 要素，group 要素らをグループ化する．

3.3　属性

属性とは，開始タグの中にある「属性名="属性値"」という記述のことです．例えば，pa 要素にある「type="n"」や hp 要素にある「x="0"」です．属性値を変更することで，様々なピクトグラムを作成できるようになります．

はじめに pa 要素の type 属性を見てみましょう．初期状態では「type="n"」となっています．"n"は Normal（標準）の略で，ピクトグラムエリアのタイプを Normal（標準）にするという意味です．"n"（Normal，標準）以外にも，"p"（Prohibit，禁止），"a"（Attention，注意），"i"（Instruction，指示），"s"（Safety，安全），"sg"（Safety Green，安全緑），"sr"（Safety Red，安全赤），"rv"（Reverse，反転）にすることもできます．なお，type 属性の値を変更すると，子孫要素の color 属性の初期値が変化する

場合があります.

type 属性の値を"p"にしてみましょう.

コード例 3-3-1

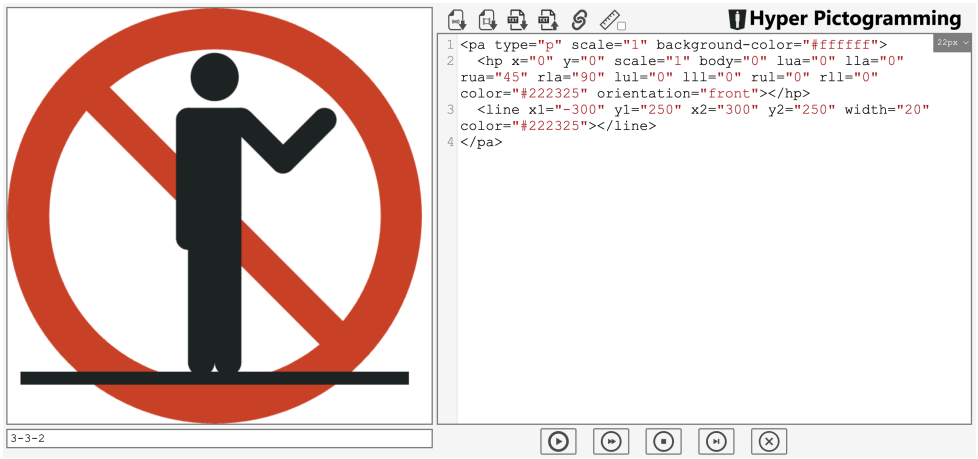
```
<pa type="p" scale="1" background-color="#ffffff">
  <hp x="0" y="0" scale="1" body="0" lua="0" lla="0" rua="0"
rla="0" lul="0" lll="0" rul="0" rll="0" color="#222325"
orientation="front"></hp>
  <line x1="-300" y1="250" x2="300" y2="250" width="20"
color="#222325"></line>
</pa>
```

すると，上の図のように禁止のピクトグラムになったと思います．pa 要素の type 属性を変更したことで，禁止のピクトグラムに変更できました．ですが，人型ピクトグラムが初期状態のままなので，何を禁止しているピクトグラムなのかわかりません．そのため，ここからは hp 要素と line 要素の属性を変化させ，歩きスマホ禁止のピクトグラムを作っていきます.

はじめに人型ピクトグラムが歩いているようにします．まず，スマホを持っている手を作成しましょう．rua 属性と rla 属性の値をそれぞれ"45"と"90"にしてください.

コード例 3-3-2

```
<pa type="p" scale="1" background-color="#ffffff">
  <hp x="0" y="0" scale="1" body="0" lua="0" lla="0" rua="45" rla="90" lul="0" lll="0" rul="0" rll="0" color="#222325" orientation="front"></hp>
  <line x1="-300" y1="250" x2="300" y2="250" width="20" color="#222325"></line>
</pa>
```



すると，上の図のように人型ピクトグラムの右肩と右肘の角度が変化したと思います．`rua` 属性は Right Upper Arm（右肩），`rla` 属性は Right Lower Arm（右肘）の角度を指定するための属性です．「`rua="45"`」，「`rla="90"`」としたことで右肩が 45 度，右肘が 90 度，反時計周りに回転しました．

以下に人型ピクトグラムのパーツの一覧を示します．体は `body` です．手足は英字 3 字で表現します．1 文字目は左（Left）か右（Right）か，2 文字目は上側（Upper）か下側（Lower）か，3 文字目は腕（Arm）か足（Leg）かを意味します．

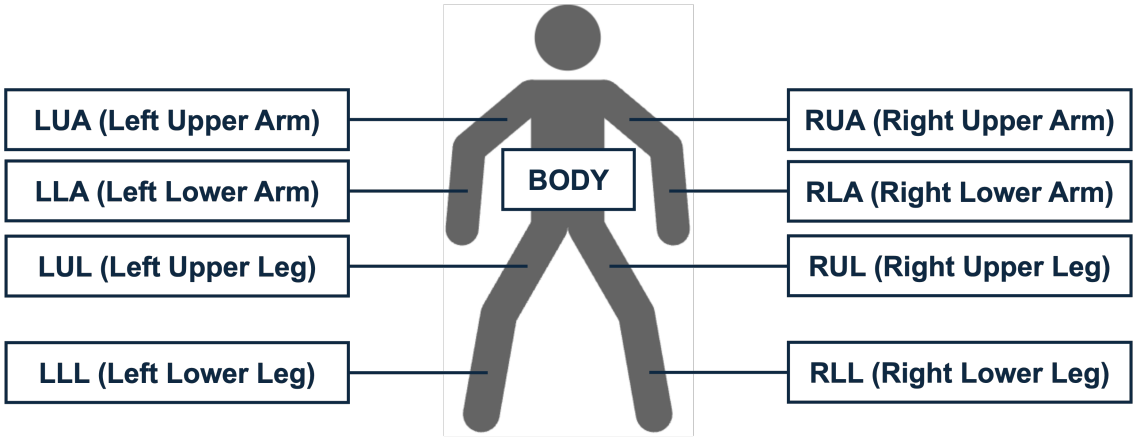
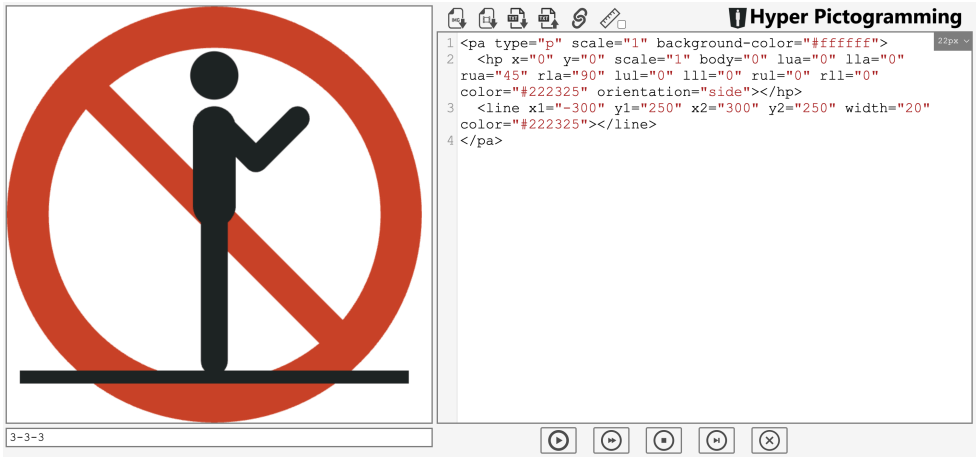


図　人型ピクトグラムのパーツ

続いては，人型ピクトグラムの向きを変化させる orientation 属性を変化させてみましょう．正面向きを意味する"front"から側面向きを意味する"side"に変更します．

コード例 3-3-3

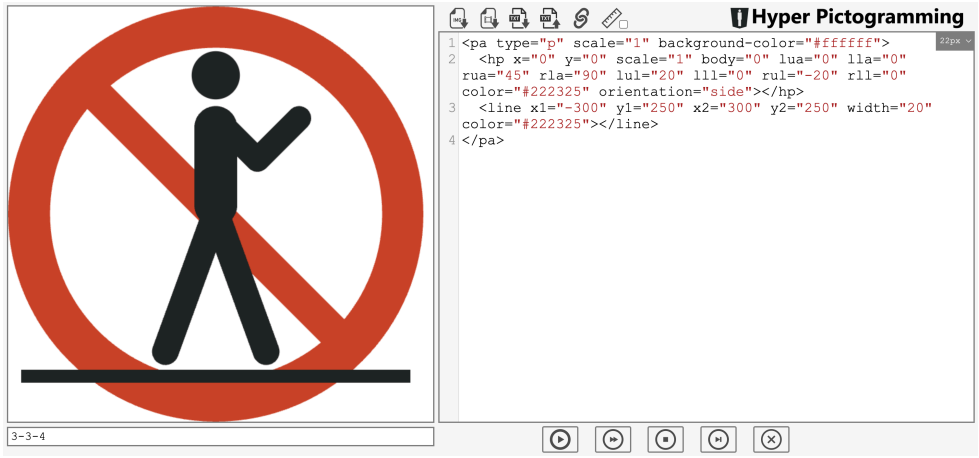
```
<pa type="p" scale="1" background-color="#ffffff">
  <hp x="0" y="0" scale="1" body="0" lua="0" lla="0" rua="45" rla="90" lul="0" lll="0" rul="0" rll="0" color="#222325" orientation="side"></hp>
  <line x1="-300" y1="250" x2="300" y2="250" width="20" color="#222325"></line>
</pa>
```



すると，上の図のように人型ピクトグラムが側面向きになったと思います．最後に，左股（Left Upper Leg）と右股（Right Upper Leg）の角度を変え，歩いているようにしましょう．lul 属性と rul 属性の値を変更します．

コード例 3-3-4

```
<pa type="p" scale="1" background-color="#ffffff">
  <hp x="0" y="0" scale="1" body="0" lua="0" lla="0" rua="45" rla="90" lul="20" lll="0" rul="-20" rll="0" color="#222325" orientation="side"></hp>
  <line x1="-300" y1="250" x2="300" y2="250" width="20" color="#222325"></line>
</pa>
```

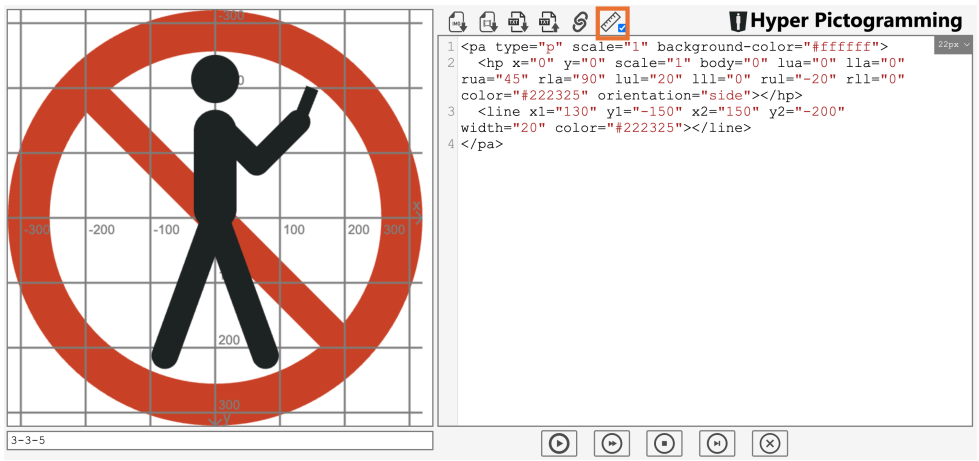


すると，上の図のように歩きスマホをしている人型ピクトグラムを作成することができたと思います.

次は line 要素の属性を変更し，スマートフォンを作成しましょう．x1 属性，y1 属性，x2 属性，y2 属性の値を変化させることで，線を描画する位置を変更します．なお，画面右上の座標系表示ボタンをクリックして座標系を表示すると便利です.

コード例 3-3-5

```
1  <pa type="p" scale="1" background-color="#ffffff">
2    <hp x="0" y="0" scale="1" body="0" lua="0" lla="0" rua="45"
   rla="90" lul="20" lll="0" rul="-20" rll="0" color="#222325"
   orientation="side"></hp>
3    <line x1="130" y1="-150" x2="150" y2="-200" width="20"
   color="#222325"></line>
4  </pa>
```



すると，上の図のようにスマートフォンを作成できたと思います．ここまでで，pa 要素，hp 要素，line 要素を使用して歩きスマホ禁止のピクトグラムを作成できました．

以下にそれぞれの要素の属性一覧を示します．必要になったときに参照してください．なお，値が初期値のまま属性に関しては，削除してしまっても同じ処理がされます．そのため，これ以降のコード例はそれらを削除したものを提示します．

color 属性や background-color 属性には，色キーワードや 16 進数文字列，RGB 関数などで色を指定できます．例えば，赤色を指定したい場合，色キーワードなら"red"，16 進数文字列なら"#ff0000"，RGB 関数なら"rgb(255, 0, 0)"となります．詳しくは，教科書や MDN のドキュメント等を参照してください．

text 要素の font-family 属性には，"Arial"などのフォントの名称を指定することができます．詳細は MDN のドキュメント等を参照してください．

表　pa 要素の属性一覧

属性	とりうる値	処理	初期値
type="arg1"	"n", "p", "a", "i", "s", "sg", "sr", "rv"	タイプを arg1 が"n"のとき Normal（標準），"p"のとき Prohibit（禁止），"a"のとき Attention（注意），"i"のとき Instruction（指示），"s"のとき Safety（安全），"sg"のとき Safety Green（安全緑），"sr"のとき Safety Red（安全赤），"rv"のとき Reverse（反転）にする．	"n"
scale="arg1"	正の数	幅，高さともに arg1×640 ピクセルにする．	"1"
background-color="arg1"	色キーワード，16 進数文字列，RGB 関数など	背景色を arg1 にする．type 属性の値が"n"，"p"，"a"，"i"，"s"のいずれかの場合のみ有効であり，"sg"，"sr"，"rv"の場合無効である．	"#ffffff"
ps="arg1"	"none", "anger", "disgust", "fear", "happiness", "sadness", "surprise"	arg1 が"anger"のとき怒り，"disgust"のとき嫌悪，"fear"のとき恐れ，"happiness"のとき喜び，"sadness"のとき悲しみ，"surprise"のとき驚きの仮面を装着する．arg1 が"none"のときは仮面を外す．	"none"

表　hp 要素の属性一覧

属性	とりうる値	処理	初期値
x="arg1", y="arg2"	数値	座標（arg1，arg2）を中心座標にする.	ともに"0"
scale="arg1"	正の数	大きさを標準（初期状態）の arg1 倍にする.	"1"
body="arg1", lua="arg2", lla="arg3", rua="arg4", rla="arg5", lul="arg6", lll="arg7", rul="arg8", rll="arg9"	数値	体（Body）を arg1 度，左肩（Left Upper Arm）を arg2 度，左肘（Left Lower Arm）を arg3 度，右肩（Right Upper Arm）を arg4 度，右肘（Right Lower Arm）を arg5 度，左股（Left Upper Leg）を arg6 度，左膝（Left Lower Leg）を arg7 度，右股（Right Upper Leg）を arg8 度，右膝（Right Lower Leg）を arg9 度反時計回りに回転させる.	全て"0"
orientation ="arg1"	"front"，または "side"	arg1 が"front"の場合は正面向きに，"side"の場合は横向きにする.	"front"
color="arg1"	色キーワード，16 進数文字列，RGB 関数など	色を arg1 にする.	pa 要素の type 属性の値が "n"，"p"，"a"のときは "#222325"，"i"，"sg"，"sr"，"rv"のときは "#eeedef"，"s"のときは "#009c64"
ms="arg1"	"none", "anger", "disgust", "fear", "happiness", "sadness", "surprise"	arg1 が"anger"のとき怒り，"disgust"のとき嫌悪，"fear"のとき恐れ，"happiness"のとき喜び，"sadness"のとき悲しみ，"surprise"のとき驚きの仮面を装着する．arg1 が"none"のときは仮面を外す.	"none"
sk=arg1	真偽値（arg1 は省略可能）	人型ピクトグラムをスケルトンモードに変更する.	false

表　line 要素の属性一覧

属性	とりうる値	処理	初期値
x1="arg1", y1="arg2", x2="arg3", y2="arg4"	数値	座標（arg1, arg2）から，座標（arg3, arg4）を描画範囲にする.	全て"0"
width="arg1"	正の数	太さを arg1 にする.	"20"
color="arg1"	色キーワード，16 進数文字列，RGB 関数など	色を arg1 にする.	pa 要素の type 属性の値が"n", "p", "a"のときは"#222325", "i", "sg", "sr", "rv"のときは"#eeedef", "s"のときは"#009c64"

表　ellipse 要素の属性一覧

属性	とりうる値	処理	初期値
x="arg1", y="arg2"	数値	座標（arg1, arg2）を中心座標にする.	ともに"0"
width="arg1", height="arg2"	正の数	幅を arg1，高さを arg2 にする.	ともに"100"
angle="arg1"	数値	arg1 度反時計回りに回転させる.	"0"
color="arg1"	色キーワード，16 進数文字列，RGB 関数など	色を arg1 にする.	pa 要素の type 属性の値が"n", "p", "a"のときは"#222325", "i", "sg", "sr", "rv"のときは"#eeedef", "s"のときは"#009c64"

表　text 要素の属性一覧

属性	とりうる値	処理	初期値
x="arg1", y="arg2"	数値	座標（arg1, arg2）を描画の左下の点にする.	ともに"0"
font-size="arg1"	正の数	フォントの大きさを arg1 にする.	"50"
font-family="arg1"	フォントファミリー	フォントファミリーを arg1 にする.	"Hiragino Kaku Gothic ProN"
color="arg1"	色キーワード，16 進数文字列，RGB 関数など	色を arg1 にする.	pa 要素の type 属性の値が"n", "p", "a"のときは"#222325", "i", "sg", "sr", "rv"のときは"#eeedef", "s"のときは"#009c64"

表　group 要素の属性一覧

属性	とりうる値	処理	初期値
x="arg1", y="arg2"	数値	座標（arg1, arg2）を中心座標にする.	ともに"0"
color="arg1"	色キーワード，16 進数文字列，RGB 関数など	全ての子要素の色を arg1 にする.	pa 要素の type 属性の値が"n", "p", "a"のときは"#222325", "i", "sg", "sr", "rv"のときは"#eeedef", "s"のときは"#009c64"

3.4　要素の追加

pa 要素の中には，要素をいくつも追加することができます．hp 要素と line 要素を追加し，歩きスマホをしている人型ピクトグラムを 2 人にしましょう．なお，追加するだけでは元々いる人型ピクトグラムとスマートフォンと重なってしまうので，hp 要素の x 属性，line 要素の x1 属性，x2 属性の値を変更し，位置をずらします.

コード例 3-4-1

```
1 <pa type="p">
2   <hp x="-130" rua="45" rla="90" lul="20" rul="-20"
  orientation="side"></hp>
3   <hp x="130" rua="45" rla="90" lul="20" rul="-20"
  orientation="side"></hp>
4   <line x1="0" y1="-150" x2="20" y2="-200"></line>
5   <line x1="260" y1="-150" x2="280" y2="-200"></line>
6 </pa>
```

すると，上の図のように，2 人で歩きスマホをしているようにできたと思います．line 要素と ellipse 要素，text 要素，group 要素に関してもいくつも追加できます．複数の要

素を使用することでピクトグラムの表現の幅が広がります．また，pa 要素を複数使用して，ピクトグラムを複数作成することも可能です．ぜひ試してみてください．

3.5　HPML と HTML

ここまではピクトグラムを作成するために HPML というハイパーピクトグラミング独自のマークアップ言語を使用してきました．ですが，ハイパーピクトグラミングでは HPML を HTML と組み合わせることも可能です．

歩きスマホ禁止のピクトグラムに見出しを意味する h1 要素を組み合わせてみましょう．

コード例 3-5-1

```
<h1>歩きスマホ禁止</h1>
<pa type="p">
  <hp x="-130" rua="45" rla="90" lul="20" rul="-20" orientation="side"></hp>
  <hp x="130" rua="45" rla="90" lul="20" rul="-20" orientation="side"></hp>
  <line x1="0" y1="-150" x2="20" y2="-200"></line>
  <line x1="260" y1="-150" x2="280" y2="-200"></line>
</pa>
```

すると，上の図のようにピクトグラムと見出しを同時に表示できたと思います．

HTML には h1 要素以外にもたくさんの要素があります．興味がある方はぜひ試してみてください．

3.6　練習問題

(1) HPML を使用して，サッカーを禁止するピクトグラムを作成してください．

(2) HPML を使用して，何かに注意を促すピクトグラムを作成してください．

第 4 章　CSS

4.1　CSS とは

CSS (Cascading Style Sheets)とは，HTML に対して，スタイルを指定するために使用します．以下にリストの文字の色を緑色にするための CSS の例を示します．

```
1  <ul>
2    <li>リンゴ</li>
3    <li>オレンジ</li>
4    <li>ブドウ</li>
5  </ul>
6  <style>
7    li {
8      color: #008c79;
9    }
10 </style>
```

セレクター　プロパティ　値

図　（左）CSS の例，（右）表示結果

CSS を適用するには，まず style 要素を用意します（CSS 専用のファイルを作るなど他にも方法がありますが，ハイパーピクトグラミングでは style 要素を使用します)．そして，そのコンテンツに以下のようにして CSS を記述していきます．

```
セレクター {
  プロパティ: プロパティ値;
}
```

セレクターとは，どの HPML 要素に対して，CSS を適用するか指定するものです．先ほどの例では，li がセレクターです．そして，プロパティとはスタイルの種類のことです．
CSS には多くのプロパティがありますが，ハイパーピクトグラミングでは CSS の学習を短時間で行えるようにするため，HPML にのみ使用できるプロパティを用意しています．
HPML のプロパティとプロパティ値は，HPML の属性名と属性値と同様で，処理も同様です．

4.2　pa 要素に CSS を適用

さっそく pa 要素に対して CSS を使用してみましょう．まず，HPML の属性を指定していると CSS が適用できないので，pa 要素の「type="p"」という記述を削除します．そして，style 要素の中にセレクターを pa，プロパティと値を type と"p"として CSS を記述します．

コード例 4-2-1

```
1  <pa ~~type="p"~~>
2    <hp x="-130" rua="45" rla="90" lul="20" rul="-20"
   orientation="side"></hp>
3    <hp x="130" rua="45" rla="90" lul="20" rul="-20"
   orientation="side"></hp>
4    <line x1="0" y1="-150" x2="20" y2="-200"></line>
5    <line x1="260" y1="-150" x2="280" y2="-200"></line>
6  </pa>
7  <style>
8    pa {
9      type: p;
10   }
11 </style>
```

すると，上の図のように CSS で pa 要素を装飾できたことがわかったと思います．

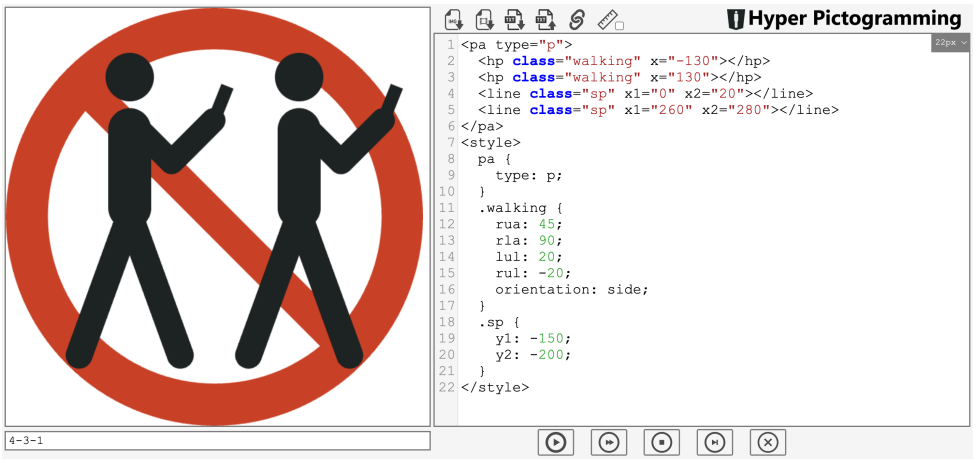
4.3　class

次は class 属性を使用して，CSS を適用していきます．class 属性は複数の要素に対して一度に CSS を適用するために使用できます．

まずは 2 つの hp 要素に対して walking，2 つの line 要素に対して sp という class 属性をつけます．そして，class 名の前に"."（ピリオド）をつけることでセレクターにし，先ほどと同様に CSS を適用していきます．

コード例 4-3-1

```
<pa type="p">
  <hp class="walking" x="-130" rua="45" rla="90" lul="20" rul="-20" orientation="side"></hp>
  <hp class="walking" x="130" rua="45" rla="90" lul="20" rul="-20" orientation="side"></hp>
  <line class="sp" x1="0" y1="-150" x2="20" y2="-200"></line>
  <line class="sp" x1="260" y1="-150" x2="280" y2="-200"></line>
</pa>
<style>
  pa {
    type: p;
  }
  .walking {
    rua: 45;
    rla: 90;
    lul: 20;
    rul: -20;
    orientation: side;
  }
  .sp {
    y1: -150;
    y2: -200;
  }
</style>
```



すると，上の図のように class 属性を使用して CSS を適用できたと思います．class 属性を使用することで，短いコードで CSS を適用できるようになります．

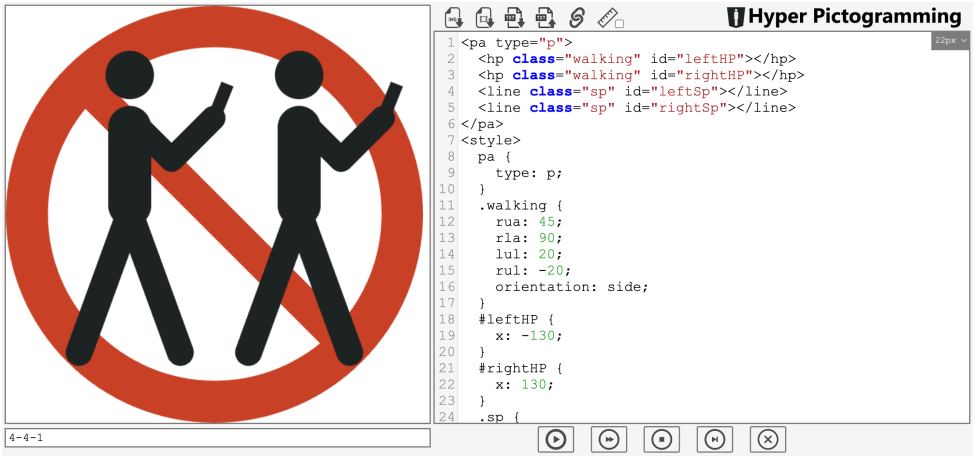
4.4 id

次は id 属性を使用して CSS を適用します．id 属性は，特定の要素を一意に識別するための属性です．class 属性とは異なり，同じ id 属性の値を複数の要素に付けることはできません．つまり，id 属性は一意である必要があります．

まず，1 つ目の hp 要素は leftHP，2 つ目の hp 要素に対しては rightHP，1 つ目の line 要素には leftSp，2 つ目の line 要素に対しては rightSp とします．そして，id 名の前に"#"（ハッシュ）をつけることでセレクターにし，CSS を適用します．

コード例 4-4-1

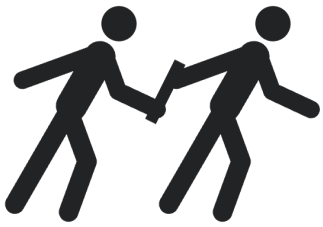
```
<pa type="p">
  <hp class="walking" id="leftHP" x="-130"></hp>
  <hp class="walking" id="rightHP" x="130"></hp>
  <line class="sp" id="leftSp" x1="0" x2="20"></line>
  <line class="sp" id="rightSp" x1="260" x2="280"></line>
</pa>
<style>
  pa {
    type: p;
  }
  .walking {
    rua: 45;
    rla: 90;
    lul: 20;
    rul: -20;
    orientation: side;
  }
  #leftHP {
    x: -130;
  }
  #rightHP {
    x: 130;
  }
  .sp {
    y1: -150;
    y2: -200;
  }
  #leftSp {
    x1: 0;
    x2: 20;
  }
  #rightSp {
    x1: 260;
    x2: 280;
  }
</style>
```



すると，上の図のように id 属性と class 属性を使用し，CSS で歩きスマホ禁止のピクトグラムを作成できたと思います.

4.5　練習問題

(1) HPML と CSS を使用して，バトンパスをしているピクトグラムを作成してください.



(2) HPML と CSS を使用して，何かを指示するピクトグラムを作成してください.

第 5 章　JavaScript

5.1　JavaScript とは

この章からはピクトグラムを JavaScript で作成していきます．JavaScript とは，例えば Web ページでボタンがユーザーによって押されたときに何かを実行するといったように，Web ページのインタラクションを担当するプログラミング言語です．

プログラミングの学習を始めると難しいと感じる人も少なくないかもしれません．ですが，ハイパーピクトグラミングではピクトグラムを操作するための JavaScript を定義しています．ピクトグラムを作成しながら，楽しくプログラミングを学んでいきましょう．

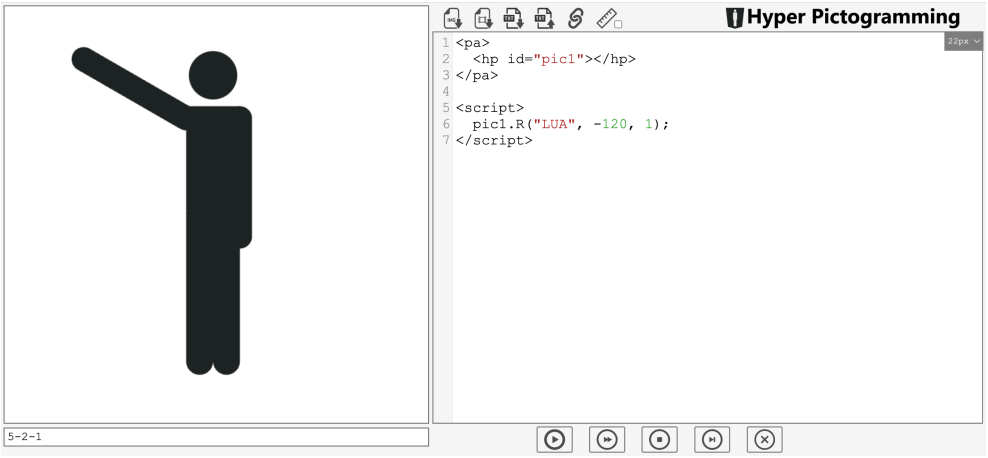
5.2　逐次実行と並列実行

ハイパーピクトグラミングで JavaScript を実行するには，まず `script` 要素を用意し（JavaScript 専用のファイルを作るなどの方法もありますが，ハイパーピクトグラミングでは `script` 要素を使用します），その中に処理を書いていきます．

ここからは，バイバイのピクトグラムを作りながら JavaScript を学習していきます．まずは，`hp` 要素の `id` 属性の値を `pic1` にします．そして，`script` 要素の中に「`pic1.R("LUA", -120, 1);`」と記述します．

コード例 5-2-1

```
1 <pa>
2   <hp id="pic1"></hp>
3 </pa>
4
5 <script>
6   pic1.R("LUA", -120, 1);
7 </script>
```



すると，人型ピクトグラムの左肩が回転したと思います．「pic1.R("LUA", -120, 1);」は「pic1 が 1 秒かけて LUA（Left Upper Arm）を反時計周りに-120 度回転させる」という意味だからです．左から順に説明していきます．

```
                arg1   arg2 arg3
pic1.R("LUA", -120, 1);
識別子 メソッド     引数
```

pic1 は人型ピクトグラムを示す識別子（名称）です．HPML 要素の id 属性の値をそのまま記述します．なお，通常は HTML 要素の id 属性の値の中に"-"（ハイフン）を使用することができますが，ハイパーピクトグラミングでは id 属性の値がそのまま識別子になるという仕様上，"-"を使用することができません．例えば，id 属性の値を pic-1 のようにすると，エラーが出てしまいます．

識別子に対して何か処理を実行するには"."（ピリオド）を記述します．そして，その後に R メソッドが続きます．メソッドとは，識別子に対して実行することができる処理のようなものです．R は回転（Rotate）という意味です．メソッド名の後には"()"と記述します．なお，この中に引数（argument）がある場合があります．引数とはメソッドを使用するときに渡される値のことです．引数は複数ある場合もあり，その場合は","（カンマ）で区切ります．なお，ここからは左から 1 つ目の引数のことを arg1，2 つ目の引数のことを arg2，…，N つ目の引数のことを argN と表記します．

R メソッドは「arg3 秒かけて arg1 で指定される体の部位を，反時計回りに arg2 度回転する」という意味です．そのため，この 1 文は「pic1 が 1 秒かけて LUA（Left Upper Arm）を反時計周りに-120 度回転させる」という意味になります．なお，JavaScript では文の終わりには";"（セミコロン）を書くことが推奨されているためメソッドの最後には

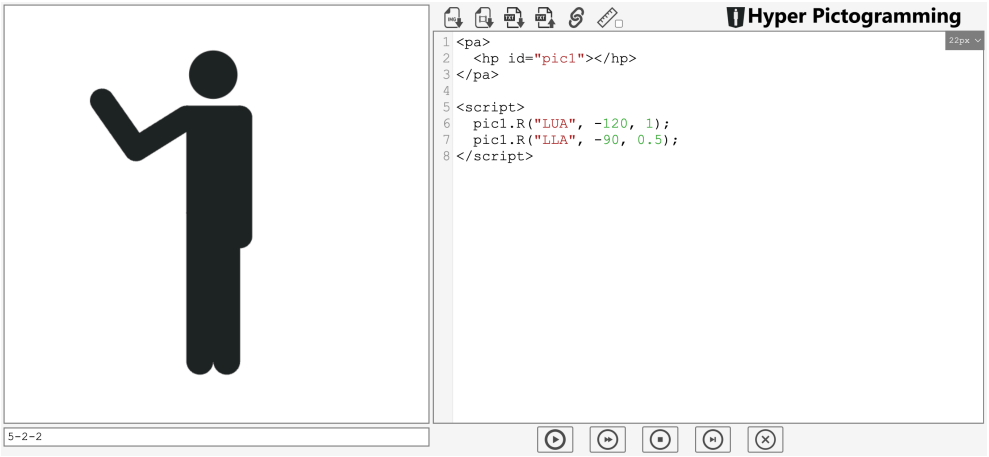
";"と記述しています.

ちなみに，メソッド名やパーツは大文字でも小文字でも同じ意味です．つまり，「pic1.R("LUA", -120, 1);」は「pic1.r("lua", -120, 1);」としても同様に処理されます.

次は，LUA（Left Upper Arm）を回転が終わった後に LLA（Left Lower Arm）を回転させたいです．先ほどの文の後に，「pic1.R("LLA", -90, 0.5);」という文を追加してください.

コード例 5-2-2

```
<pa>
  <hp id="pic1"></hp>
</pa>

<script>
  pic1.R("LUA", -120, 1);
  pic1.R("LLA", -90, 0.5);
</script>
```



すると，LUA（Left Upper Arm）の回転が終わった後に LLA（Left Lower Arm）が回転するのではなく，LUA（Left Upper Arm）の回転と同時に LLA（Left Lower Arm）が回転してしまいました．これは R メソッドが複数の処理を同時に行う並列実行のためのメソッドのためです.

LUA（Left Upper Arm）の回転が終わった後に LLA（Left Lower Arm）を回転させるには，並列実行の R メソッドではなく，処理を 1 つずつ順番に行う逐次実行の RW メソッドを使用します．RW は回転待ち（Rotate Wait）という意味です．R メソッドとは，回転が終

了するまで他のメソッドは実行しないという違いがあります.

先ほどの 2 文を R メソッドから RW メソッドに変更します.

コード例 5-2-3

```
<pa>
  <hp id="pic1"></hp>
</pa>

<script>
  pic1.RW("LUA", -120, 1);
  pic1.RW("LLA", -90, 0.5);
</script>
```

すると，LUA（Left Upper Arm）の回転が終わった後に LLA（Left Lower Arm）を回転させることができたと思います.

続いては，RW メソッドの文を追加して，バイバイをしているようにしましょう．LLL（Left Lower Arm）を回転させ，元に戻すということを繰り返します.

コード例 5-2-4

```
<pa>
  <hp id="pic1"></hp>
</pa>

<script>
  pic1.RW("LUA", -120, 1);

  pic1.RW("LLA", -90, 0.5);
  pic1.RW("LLA", 90, 0.5);

  pic1.RW("LLA", -90, 0.5);
  pic1.RW("LLA", 90, 0.5);

  pic1.RW("LLA", -90, 0.5);
  pic1.RW("LLA", 90, 0.5);

  pic1.RW("LUA", 120, 1);
</script>
```

すると，3 回手を振るバイバイのピクトグラムにすることができたと思います．

ハイパーピクトグラミングには，要素ごとにいくつかのメソッドが存在します．以下にそれぞれのメソッドの一覧を示します．必要に応じて参照してください．なお，識別子は"*"としていますが，id 属性の値に書き換えてください．

表　pa（ピクトグラムエリア）要素が使用可能なメソッド一覧

メソッドの様式	処理
*.N();	タイプを Normal（標準）にする.
*.P();	タイプを Prohibit（禁止）にする.
*.A();	タイプを Attention（注意）にする.
*.I();	タイプを Instruction（指示）にする.
*.S();	タイプを Safety（安全）にする.
*.SG();	タイプを Safety Green（安全緑）にする.
*.SR();	タイプを Safety Red（安全赤）にする.
*.RV();	タイプを Reverse（反転）にする.
*.SC(arg1);	幅，高さともに arg1×640 ピクセルにする.
*.BCL(arg1);	背景色を arg1 にする.
*.PS(arg1);	arg1 が"anger"のとき怒り，"disgust"のとき嫌悪，"fear"のとき恐れ，"happiness"のとき喜び，"sadness"のとき悲しみ，"surprise"のとき驚きの仮面を装着する．arg1 が"none"のときは仮面を外す.
*.HP(arg1, arg2, arg3, arg4, arg5, arg6, arg7, arg8, arg9, arg10, arg11, arg12, arg13, arg14, arg15);	子要素に id=arg1，x=arg2，y=arg3，scale=arg4，body=arg5，lua=arg6，lla=arg7，rua=arg8，rla=arg9，lul=arg10，lll=arg11，rul=arg12，rll=arg13，orientation=arg14，color=arg15 の hp 要素を追加する．arg1 が省略されたときは，arg1 に""（空文字列）が，arg2 から arg15 が省略されたときは，それぞれ対応する属性の初期値が入力されているものとして取り扱う.
*.L(arg1, arg2, arg3, arg4, arg5, arg6, arg7);	子要素に id=arg1，x1=arg2，y1=arg3，x2=arg4，y2=arg5，width=arg6，color=arg7 の line 要素を追加する．arg1 が省略されたときは，arg1 に""（空文字列）が，arg2 から arg7 が省略されたときは，それぞれ対応する属性の初期値が入力されているものとして取り扱う.
*.E(arg1, arg2, arg3, arg4, arg5, arg6, arg7);	子要素に id=arg1，x=arg2，y=arg3，width=arg4，height=arg5，angle=arg6，color=arg7 の ellipse 要素を追加する．arg1 が省略されたときは，arg1 に""（空文字列）が，arg2 から arg7 が省略されたときは，それぞれ対応する属性の初期値が入力されているものとして取り扱う.
*.T(arg1, arg2, arg3, arg4, arg5, arg6, arg7);	子要素に id=arg1，テキストコンテンツが arg2，x=arg3，y=arg4，font-size=arg5，font-family=arg6，color=arg7 の text 要素を追加する．arg1，arg2 が省略されたときは，いずれも""（空文字列）が，arg3 から arg7 が省略されたときは，それぞれ対応する属性の初期値が入力されているものとして取り扱う.
*.G(arg1, arg2, arg3, arg4);	子要素に id=arg1，x=arg2，y=arg3，color=arg4 の group 要素を追加する．arg1 が省略されたときは，arg1 に""（空文字列）が，arg2 から arg4 が省略されたときは，それぞれ対応する属性の初期値が入力されているものとして取り扱う.

表　hp（人型ピクトグラム）要素が使用可能なメソッド一覧

メソッドの様式	処理
*.M(arg1, arg2, arg3, arg4);	arg4 秒後に arg3 秒かけて水平方向に arg1，垂直方向に arg2 だけ全体を等速直線移動する．arg4 が省略されたときは，arg4 に 0 が，arg3，arg4 の両方が省略されたときは，いずれも 0 が入力されているものとして取り扱う．
*.MW(arg1, arg2, arg3);	arg3 秒かけて水平方向に arg1，垂直方向に arg2 だけ全体を等速直線移動する．直線移動が終了するまで次のメソッドは実行されない．arg3 が省略されたときは，arg3 に 0 が入力されているものとして取り扱う．
*.R(arg1, arg2, arg3, arg4);	arg4 秒後に arg1 で指定される体の部位を反時計回りに arg2 度だけ arg3 秒かけて支点を中心に等速回転する．arg4 が省略されたときは，arg4 に 0 が，arg3，arg4 が省略されたときは，いずれも 0 が入力されているものとする．
*.RW(arg1, arg2, arg3);	arg1 で指定される体の部位を反時計回りに arg2 度だけ arg3 秒かけて支点を中心に等速回転する．回転が終了するまで次のメソッドは実行されない．arg3 が省略されたときは，arg3 に 0 が入力されているものとする．
*.SC(arg1);	大きさを標準（初期状態）の arg1 倍にする．
*.FR();	正面向きにする．
*.SD();	側面向きにする．
*.CL(arg1);	色を arg1 にする．
*.MS(arg1);	arg1 が"anger"のとき怒り，"disgust"のとき嫌悪，"fear"のとき恐れ，"happiness"のとき喜び，"sadness"のとき悲しみ，"surprise"のとき驚きの仮面を装着する．arg1 が"none"のときは仮面を外す．
*.C();	初期状態にする．
*.ST();	メソッドが実行される時点での人型ピクトグラムを複製（スタンプ）する．
*.SAY(arg1, arg2, arg3);	arg3 秒後に arg1 で指定される値を arg2 秒だけ吹き出しで表示する．arg3 が省略されたときは，arg3 に 0 が，arg2，arg3 の両方が省略されたときは，いずれも"0"が入力されているものとして取り扱う．
*.SAYW(arg1, arg2);	arg1 で指定される値を arg2 秒だけ吹き出しで表示する．吹き出しの表示が終了するまで次のメソッドは実行されない．arg2 が省略されたときは，arg2 に 0 が入力されているものとして取り扱う．
*.THINK(arg1, arg2, arg3);	arg3 秒後に arg1 で指定される値を arg2 秒だけ思考吹き出しで表示する．arg3 が省略されたときは，arg3 に 0 が，arg2，arg3 の両方が省略されたときは，いずれも"0"が入力されているものとして取り扱う．
*.THINKW(arg1, arg2);	arg1 で指定される値を arg2 秒だけ思考吹き出しで表示する．吹き出しの表示が終了するまで次のメソッドは実行されない．arg2 が省略されたときは，arg2 に 0 が入力されているものとして取り扱う．
*.SP(arg1, arg2);	言語設定を arg2 にし，文字列 arg1 を発話する．arg2 が省略されたときは，ブラウザの言語設定が日本語の場合は"ja-JP"，そのほかの場合は"en-US"として取り扱う．

表　line（線）要素が使用可能なメソッド一覧

メソッドの様式	処理
*.M(arg1, arg2, arg3, arg4);	arg4 秒後に arg3 秒かけて水平方向に arg1，垂直方向に arg2 だけ全体を等速直線移動する．arg4 が省略されたときは，arg4 に"0"が，arg3，arg4 の両方が省略されたときは，いずれも 0 が入力されているものとして取り扱う．
*.MW(arg1, arg2, arg3);	arg3 秒かけて水平方向に arg1，垂直方向に arg2 だけ全体を等速直線移動する．直線移動が終了するまで次のメソッドは実行されない．arg3 が省略されたときは，arg3 に 0 が入力されているものとして取り扱う．
*.WD(arg1);	幅を arg1 にする．
*.CL(arg1);	色を arg1 にする．

表　ellipse（楕円）要素が使用可能なメソッド一覧

メソッドの様式	処理
*.M(arg1, arg2, arg3, arg4);	arg4 秒後に arg3 秒かけて水平方向に arg1，垂直方向に arg2 だけ全体を等速直線移動する．arg4 が省略されたときは，arg4 に 0 が，arg3，arg4 の両方が省略されたときは，いずれも 0 が入力されているものとして取り扱う．
*.MW(arg1, arg2, arg3);	arg3 秒かけて水平方向に arg1，垂直方向に arg2 だけ全体を等速直線移動する．直線移動が終了するまで次のメソッドは実行されない．arg3 が省略されたときは，arg3 に 0 が入力されているものとして取り扱う．
*.WD(arg1);	幅を arg1 にする．
*.H(arg1);	高さを arg1 にする．
*.A(arg1);	角度を arg1 度にする．
*.CL(arg1);	色を arg1 にする．

表　text（テキスト）要素が使用可能なメソッド一覧

メソッドの様式	処理
*.M(arg1, arg2, arg3, arg4);	arg4 秒後に arg3 秒かけて水平方向に arg1，垂直方向に arg2 だけ全体を等速直線移動する．arg4 が省略されたときは，arg4 に 0 が，arg3，arg4 の両方が省略されたときは，いずれも 0 が入力されているものとして取り扱う．
*.MW(arg1, arg2, arg3);	arg3 秒かけて水平方向に arg1，垂直方向に arg2 だけ全体を等速直線移動する．直線移動が終了するまで次のメソッドは実行されない．arg3 が省略されたときは，arg3 に 0 が入力されているものとして取り扱う．
*.FS(arg1);	フォントサイズを arg1 にする．
*.FF(arg1);	フォントファミリーを arg1 にする．
*.CL(arg1);	色を arg1 にする．
*.TC(arg1);	テキストコンテンツを arg1 にする．

表　group（グループ）要素が使用可能なメソッド一覧

メソッドの様式	処理
*.M(arg1, arg2, arg3, arg4);	arg4 秒後に arg3 秒かけて水平方向に arg1，垂直方向に arg2 だけ全体を等速直線移動する．arg4 が省略されたときは，arg4 に 0 が，arg3，arg4 の両方が省略されたときは，いずれも 0 が入力されているものとして取り扱う．
*.MW(arg1, arg2, arg3);	arg3 秒かけて水平方向に arg1，垂直方向に arg2 だけ全体を等速直線移動する．直線移動が終了するまで次のメソッドは実行されない．arg3 が省略されたときは，arg3 に 0 が入力されているものとして取り扱う．
*.CL(arg1);	全ての子要素の色を arg1 にする．
*.HP(arg1, …, arg15);	子要素に id=arg1，x=arg2，y=arg3，scale=arg4，body=arg5，lua=arg6，lla=arg7，rua=arg8，rla=arg9，lul=arg10，lll=arg11，rul=arg12，rll=arg13，orientation=arg14，color=arg15 の hp 要素を追加する．arg1 が省略されたときは，arg1 に""（空文字列）が，arg2 から arg15 が省略されたときは，それぞれ対応する属性の初期値が入力されているものとして取り扱う．
*.L(arg1, …, arg7);	子要素に id=arg1，x1=arg2，y1=arg3，x2=arg4，y2=arg5，width=arg6，color=arg7 の line 要素を追加する．arg1 が省略されたときは，arg1 に""（空文字列）が，arg2 から arg7 が省略されたときは，それぞれ対応する属性の初期値が入力されているものとして取り扱う．
*.E(arg1, …, arg7);	子要素に id=arg1，x=arg2，y=arg3，width=arg4，height=arg5，angle=arg6，color=arg7 の ellipse 要素を追加する．arg1 が省略されたときは，arg1 に""（空文字列）が，arg2 から arg7 が省略されたときは，それぞれ対応する属性の初期値が入力されているものとして取り扱う．
*.T(arg1, …, arg7);	子要素に id=arg1，テキストコンテンツが arg2，x=arg3，y=arg4，font-size=arg5，font-family=arg6，color=arg7 の text 要素を追加する．arg1，arg2 が省略されたときは，いずれも""（空文字列）が，arg3 から arg7 が省略されたときは，それぞれ対応する属性の初期値が入力されているものとして取り扱う．
*.G(arg1, …, arg4);	子要素に id=arg1，x=arg2，y=arg3，color=arg4 の group 要素を追加する．arg1 が省略されたときは，arg1 に""（空文字列）が，arg2 から arg4 が省略されたときは，それぞれ対応する属性の初期値が入力されているものとして取り扱う．

表　すべての要素が使用可能なメソッド一覧

メソッドの様式	処理
*.W(arg1);	次のメソッドを実行せずに arg1 秒待つ．
*.SETTIME(arg1);	現在の時間を arg1 秒にする．
*.SEND_MESSAGE(arg1, arg2);	arg2 がオブジェクトの場合は arg2 に，arg2 が配列の場合は arg2 の中のすべてのオブジェクトに，メッセージ arg1 を送信する．
*.SEND_MESSAGE_TO_ALL(arg1);	すべてのオブジェクトにメッセージ arg1 を送信する．
*.RECEIVE_MESSAGE(arg1, arg2);	メッセージ arg1 を受信したとき，arg2 を実行する．

5.3　変数と定数

ここで，もう少し速く手を振りたいとします．そのためには手を振る RW メソッドの 3 つ目の引数の値を変更する必要があります．しかし，RW メソッドはすでに 6 文書いており，全て変更するのは面倒です．今回は 6 箇所変更するだけのためなんとかなりますが，場合によっては数十箇所，数百箇所変更する必要が出てきてしまうかもしれません．そこで，定数や変数を使用します．

様式	処理
const arg1 = arg2	定数 arg1 に arg2 を代入する．
let arg1 = arg2	変数 arg1 に arg2 を代入する．

定数は後から変更することができない値，一方，変数は後から変更することができる値のことです．定数は const，変数は let を用いて定義することができます．そして，値の代入には定数，変数ともに"="を使用します．

手を振る時間などを定数で定義します．

コード例 5-3-1

```
 1 <pa>
 2   <hp id="pic1"></hp>
 3 </pa>
 4
 5 <script>
 6   const raiseAngle = 120;
 7   const raiseTime = 1;
 8   const waveAngle = 90;
 9   const waveTime = 0.5;
10
11   pic1.RW("LUA", -raiseAngle, raiseTime);
12
13   pic1.RW("LLA", -waveAngle, waveTime);
14   pic1.RW("LLA", waveAngle, waveTime);
15
16   pic1.RW("LLA", -waveAngle, waveTime);
17   pic1.RW("LLA", waveAngle, waveTime);
18
19   pic1.RW("LLA", -waveAngle, waveTime);
20   pic1.RW("LLA", waveAngle, waveTime);
21
22   pic1.RW("LUA", raiseAngle, raiseTime);
23 </script>
```

すると，定数を使用して手を振るピクトグラムにすることができたと思います．定数を使用したことで，もし手を振る速さを変えたくなっても，waveTime に代入している値の 1 箇所を変更すればよくなります．

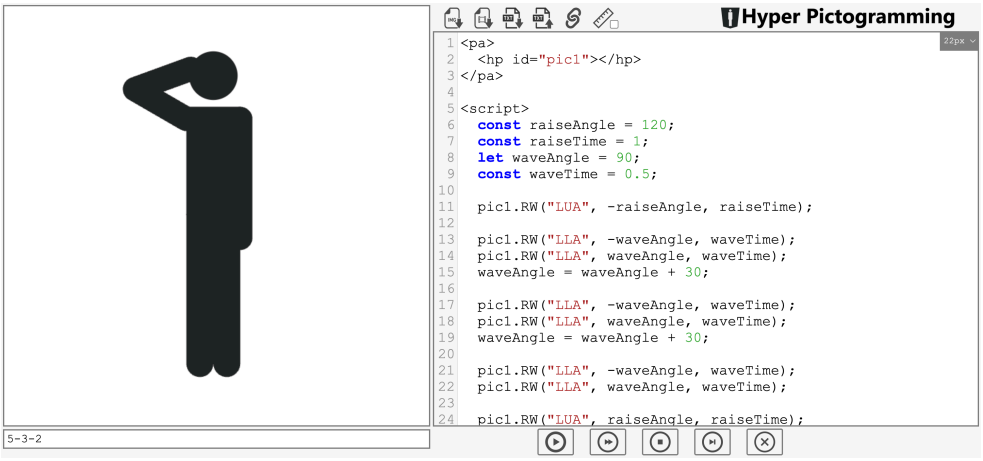
次は，手を振る角度をだんだん大きくしたいです．そのためには，算術演算子を使用します．算術演算子とは，計算のために用いる演算を表す記号のことです．

様式	処理
A + B	A と B を足す．
A - B	A と B を引く．
A * B	A と B を掛ける．
A / B	A を B で割る．
A % B	A を B で割った余り．
A ** B	A を B でべき乗する．

waveAngle を変数から定数にしておきます．そして，"+"を使用して，waveAngle の値をだんだん増加させていきます．

コード例 5-3-2

```
<pa>
  <hp id="pic1"></hp>
</pa>

<script>
  const raiseAngle = 120;
  const raiseTime = 1;
  let waveAngle = 90;
  const waveTime = 0.5;

  pic1.RW("LUA", -raiseAngle, raiseTime);

  pic1.RW("LLA", -waveAngle, waveTime);
  pic1.RW("LLA", waveAngle, waveTime);
  waveAngle = waveAngle + 30;

  pic1.RW("LLA", -waveAngle, waveTime);
  pic1.RW("LLA", waveAngle, waveTime);
  waveAngle = waveAngle + 30;

  pic1.RW("LLA", -waveAngle, waveTime);
  pic1.RW("LLA", waveAngle, waveTime);

  pic1.RW("LUA", raiseAngle, raiseTime);
</script>
```



すると，手を振る角度がだんだん大きくなっていったと思います．

5.4　繰り返し

ここまでで，バイバイのピクトグラムを作成することができました．しかし，同じような文を何回も繰り返しています．現在は 3 回手を振るだけなのでなんとかなっています

が，仮に 100 回手を振るようにしたいとすると，コードが非常に長くなってしまい大変です．そこで，for 文で繰り返しの処理を行います．

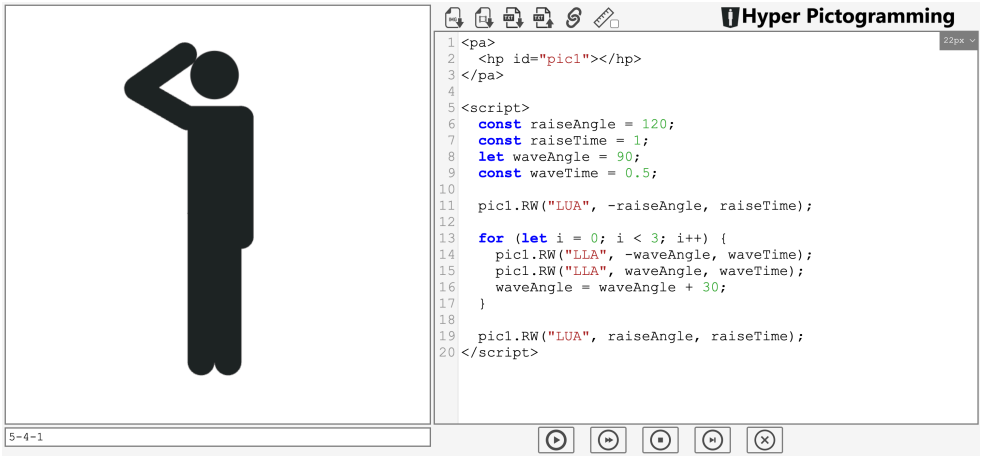
様式	処理
`for (let i = 0; i < arg1; i++) { }`	対応する処理群を arg1 回繰り返す．

「let i = 0」は繰り返しが始まる前に行われる初期化，「i < arg1」はそれぞれの繰り返しの始めに評価される式，「i++」はそれぞれの繰り返しの最後に評価される式です．つまり，変数 i に 0 を代入し，変数 i が arg1 より小さい間，処理を繰り返し，毎回最後に変数 i を 1 増やすという流れです．

for 文を使用すると，以下のように短いコードでバイバイをすることができます．

コード例 5-4-1

```
<pa>
  <hp id="pic1"></hp>
</pa>

<script>
  const raiseAngle = 120;
  const raiseTime = 1;
  let waveAngle = 90;
  const waveTime = 0.5;

  pic1.RW("LUA", -raiseAngle, raiseTime);

  for (let i = 0; i < 3; i++) {
    pic1.RW("LLA", -waveAngle, waveTime);
    pic1.RW("LLA", waveAngle, waveTime);
    waveAngle = waveAngle + 30;
  }

  pic1.RW("LUA", raiseAngle, raiseTime);
</script>
```



すると，for 文を使って短いコードでだんだん大きく手を振ることができたと思います．

5.5　条件分岐

ここまでは左手でバイバイをしてきました．ですが，左手か右手，どちらかランダムにバイバイをするようにしたいです．そこで，if 文で条件分岐を使用します．

様式	処理
`if (exp1) { }`	対応する処理群を arg1 回繰り返す．

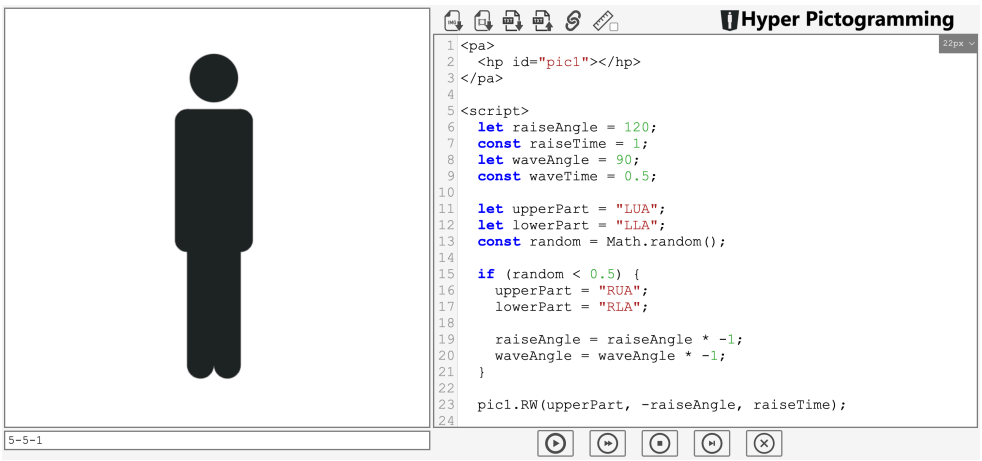
exp1 には，比較演算子を使用した条件式を記述します．

様式	評価
`A > B`	A が B より大きい．
`A >= B`	A が B より大きいか等しい．
`A < B`	A が B より小さい．
`A <= B`	A が B より小さいか等しい．
`A == B`	A が B と等しい．
`A != B`	A が B と等しくない．
`A++, ++A`	A の値を 1 増やす（A=A+1，A+=1 と同じ）．
`A--, --A`	A の値を 1 減らす（A=A-1，A-=1 と同じ）．

50%の確率で左手，50%の確率で右手でバイバイをしたいです．その条件式を作るために，Math.random()を使用します．これは 0 以上 1 未満のランダムな値（乱数）を返すものです．もし，Math.random()で生成した値が 0.5 以下ならば，右手でバイバイをするように変更します．なお，Math.random()は暗記する必要はありません．必要になったときに，「JavaScript 乱数」のように，キーワードで検索すれば大丈夫です．

コード例 5-5-1

```
<pa>
  <hp id="pic1"></hp>
</pa>

<script>
  let raiseAngle = 120;
  const raiseTime = 1;
  let waveAngle = 90;
  const waveTime = 0.5;

  let upperPart = "LUA";
  let lowerPart = "LLA";
  const random = Math.random();

  if (random < 0.5) {
    upperPart = "RUA";
    lowerPart = "RLA";

    raiseAngle = raiseAngle * -1;
    waveAngle = waveAngle * -1;
  }

  pic1.RW(upperPart, -raiseAngle, raiseTime);

  for (let i = 0; i < 3; i++) {
    pic1.RW(lowerPart, -waveAngle, waveTime);
    pic1.RW(lowerPart, waveAngle, waveTime);
  }

  pic1.RW(upperPart, raiseAngle, raiseTime);
</script>
```



すると，50%ずつの確率で左手か右手でバイバイすることができたと思います．

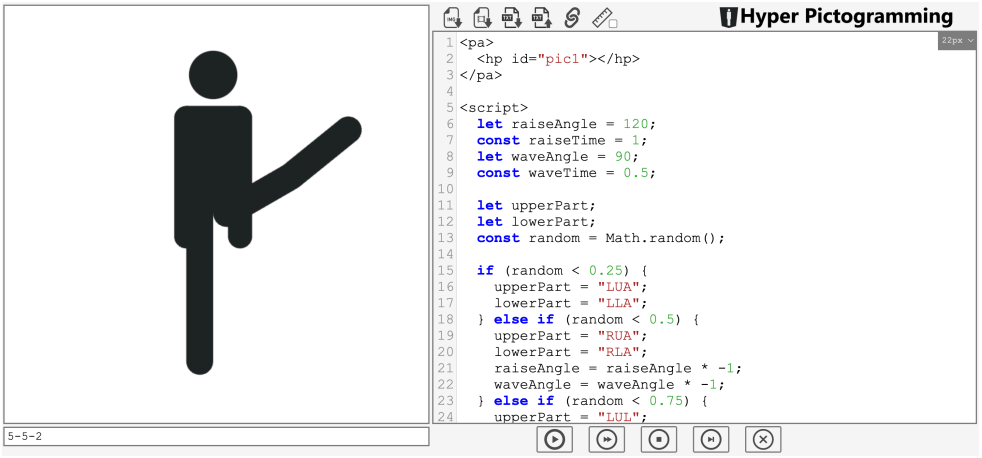
`else if` や `else` も使用して，より複雑な条件分岐にすることも可能です．

様式	処理
`if (exp1) {}`	もし `exp1` が真ならば，対応する処理群を実行する．
`else if (expN) {}`	もし対応する先述の `if` または `else if` の条件式が全て偽で，かつ `expN` が真ならば，対応する処理群を実行する．
`else {}`	もし対応する先述の `if` または `else if` の条件が全て偽ならば，対応する処理群を実行する．

左手，右手，左足，右足，それぞれ25%ずつの確率でバイバイするようにします．

コード例 5-5-2

```
<pa>
  <hp id="pic1"></hp>
</pa>

<script>
  let raiseAngle = 120;
  const raiseTime = 1;
  let waveAngle = 90;
  const waveTime = 0.5;

  let upperPart;
  let lowerPart;
  const random = Math.random();

  if (random < 0.25) {
    upperPart = "LUA";
    lowerPart = "LLA";
  } else if (random < 0.5) {
    upperPart = "RUA";
    lowerPart = "RLA";
    raiseAngle = raiseAngle * -1;
    waveAngle = waveAngle * -1;
  } else if (random < 0.75) {
    upperPart = "LUL";
    lowerPart = "LLL";
  } else {
    upperPart = "RUL";
    lowerPart = "RLL";
    raiseAngle = raiseAngle * -1;
    waveAngle = waveAngle * -1;
  }

  pic1.RW(upperPart, -raiseAngle, raiseTime);

  for (let i = 0; i < 3; i++) {
    pic1.RW(lowerPart, -waveAngle, waveTime);
    pic1.RW(lowerPart, waveAngle, waveTime);
  }

  pic1.RW(upperPart, raiseAngle, raiseTime);
</script>
```



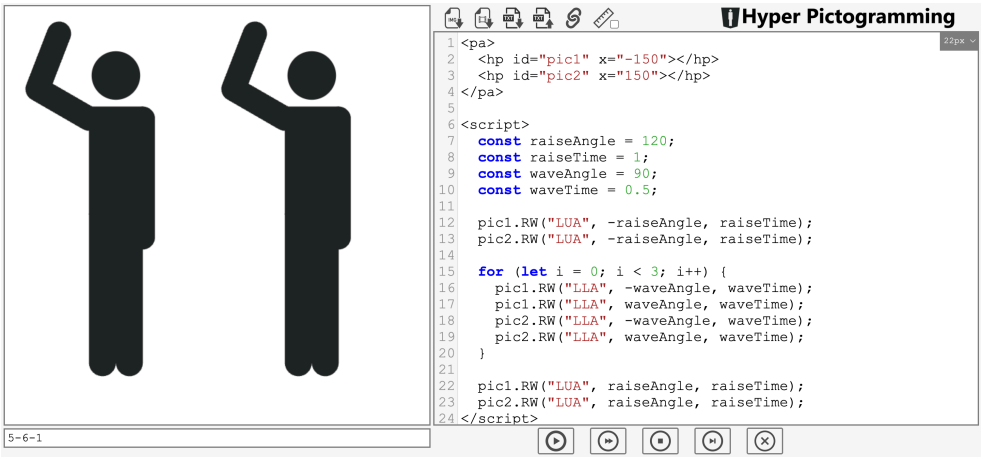
すると，左手，右手，左足，右足，それぞれ 25%の確率でバイバイできたと思います.

5.6　手続き・関数

次は，2 人でバイバイするようにします．まず，id 属性の値が pic2 の hp 要素を追加します．x 属性の値を変更し，2 人が重ならないようにします．そして，pic2 に対しても，pic1 と同じようにすると，バイバイができます.

コード例 5-6-1

```
<pa>
  <hp id="pic1" x="-150"></hp>
  <hp id="pic2" x="150"></hp>
</pa>

<script>
  const raiseAngle = 120;
  const raiseTime = 1;
  const waveAngle = 90;
  const waveTime = 0.5;

  pic1.RW("LUA", -raiseAngle, raiseTime);
  pic2.RW("LUA", -raiseAngle, raiseTime);

  for (let i = 0; i < 3; i++) {
    pic1.RW("LLA", -waveAngle, waveTime);
    pic1.RW("LLA", waveAngle, waveTime);
    pic2.RW("LLA", -waveAngle, waveTime);
    pic2.RW("LLA", waveAngle, waveTime);
  }

  pic1.RW("LUA", raiseAngle, raiseTime);
  pic2.RW("LUA", raiseAngle, raiseTime);
</script>
```



すると，2 人でバイバイができたと思います．なお，RW メソッドで待ちが発生するのは，その識別子だけです．そのため，pic1 に対して RW メソッドを使用しても，pic2 のメソッドは並列して実行されます．

しかし，現在は人型ピクトグラムが 2 人だけですが，仮に 10 人でバイバイしたいとなるとコードが長くなってしまい，大変です．それに加えて，このコードは何をするためのコードかわかりにくいです．そこで，手続き（Procedure）を使用します．

手続きとは一連の処理をまとめ，後から呼び出して実行することができる再利用可能なコードのことです．コードが短くすっきりするだけではなく，一連の処理に対して名前を定義できるため，コードがわかりやすくなります．

様式	処理
`function name(arg1, …, argN) { }`	arg1 から argN までの N 個の引数を伴う手続き name を定義する．

コード例 5-6-2

```
<pa>
  <hp id="pic1" x="-150"></hp>
  <hp id="pic2" x="150"></hp>
</pa>

<script>
  const raiseAngle = 120;
  const raiseTime = 1;
  const waveAngle = 90;
  const waveTime = 0.5;

  function byebye() {
    this.RW("LUA", -raiseAngle, raiseTime);

    for (let i = 0; i < 3; i++) {
      this.RW("LLA", -waveAngle, waveTime);
      this.RW("LLA", waveAngle, waveTime);
    }

    this.RW("LUA", raiseAngle, raiseTime);
  }

  pic1.byebye = byebye;
  pic1.byebye();
  pic2.byebye = byebye;
  pic2.byebye();
</script>
```

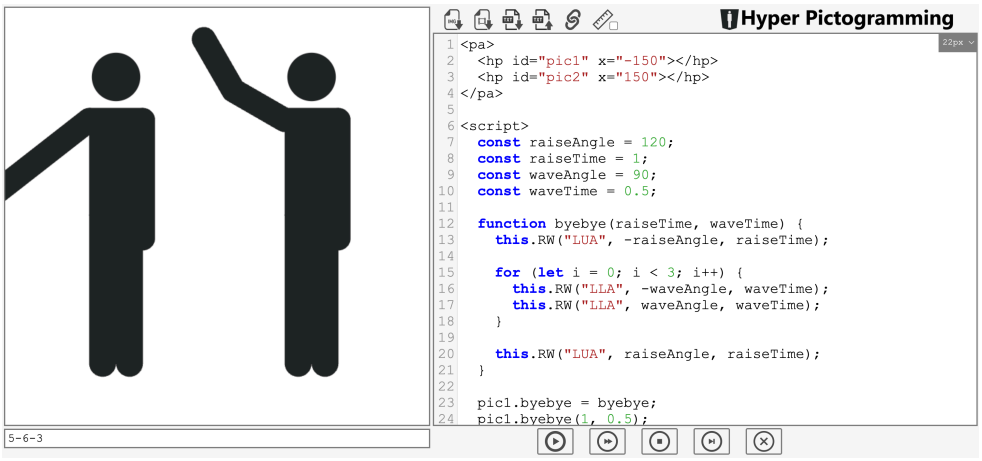
すると，先ほどと同じように 2 人でバイバイができたと思います．「picN.byebye = byebye」というのは手続き byebye を picN の byebye メソッドとして割り当てるという意味です．そして，「picN.byebye()」とすることで実行できます．byebye の中の this はその byebye メソッドを呼び出した識別子，つまり，ここでは picN を指します．

手続き byebye を使用することで，複数人でも短いコードでバイバイできるようになりました．次は，手続き byebye に引数を伴わせてみます．

先ほどまでは raiseTime と waveTime を定数で定義していましたが，この 2 つを引数にします．手続きを呼び出すときに，raiseTime と waveTime に入れたい値を記述します．なお，pic1 と pic2 で値を変え，バイバイの速さを変化させてみます．

コード例 5-6-3

```
<pa>
  <hp id="pic1" x="-150"></hp>
  <hp id="pic2" x="150"></hp>
</pa>

<script>
  const raiseAngle = -120;
  const raiseTime = 1;
  const waveAngle = 90;
  const waveTime = 0.5;

  function byebye(raiseTime, waveTime) {
    this.RW("LUA", -raiseAngle, raiseTime);

    for (let i = 0; i < 3; i++) {
      this.RW("LLA", -waveAngle, waveTime);
      this.RW("LLA", waveAngle, waveTime);
    }

    this.RW("LUA", raiseAngle, raiseTime);
  }

  pic1.byebye = byebye;
  pic1.byebye(1, 0.5);
  pic2.byebye = byebye;
  pic2.byebye(0.2, 0.1);
</script>
```



すると，2 人で速さの異なるバイバイができたと思います．

ここまでで，function を使用して手続きを使用してきました．なお，function は，戻り値というものを返すものがあります．そのようなものは手続きではなく，関数と呼ばれます．手続きは一連の処理をまとめることが目的なのに対して，関数は値を返すことが目的です．

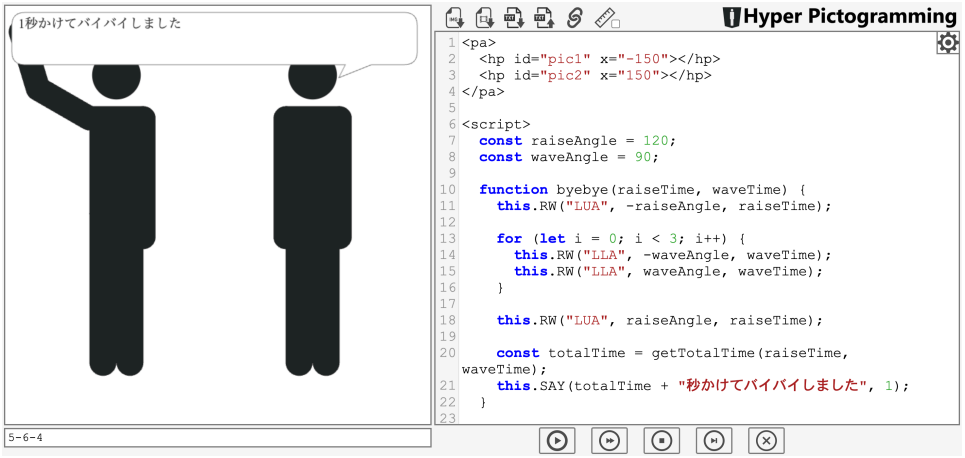
様式	処理

function name(arg1, …, argN) {
  return result;
}
	arg1 から argN までの N 個の引数を伴い，result を返す関数 name を定義する．

raiseTime と waveTime を使用して，バイバイするのに何秒かかったかを返す関数 getTotalTime を定義します．getTotalTime で取得した値を，バイバイの後に，SAY メソッドを使用して，表示します．SAY は言うという意味です．

コード例 5-6-4

```
 1 <pa>
 2   <hp id="pic1" x="-150"></hp>
 3   <hp id="pic2" x="150"></hp>
 4 </pa>
 5
 6 <script>
 7   const raiseAngle = 120;
 8   const waveAngle = 90;
 9
10   function byebye(raiseTime, waveTime) {
11     this.RW("LUA", -raiseAngle, raiseTime);
12
13     for (let i = 0; i < 3; i++) {
14       this.RW("LLA", -waveAngle, waveTime);
15       this.RW("LLA", waveAngle, waveTime);
16     }
17
18     this.RW("LUA", raiseAngle, raiseTime);
19
20     const totalTime = getTotalTime(raiseTime, waveTime);
21     this.SAY(totalTime + "秒かけてバイバイしました", 1);
22   }
23
24   function getTotalTime(raiseTime, waveTime) {
25     let totalTime = raiseTime * 2 + waveTime * 3 * 2;
26     return totalTime;
27   }
28
29   pic1.byebye = byebye;
30   pic1.byebye(1, 0.5);
31   pic2.byebye = byebye;
32   pic2.byebye(0.2, 0.1);
33 </script>
```



5.7　配列

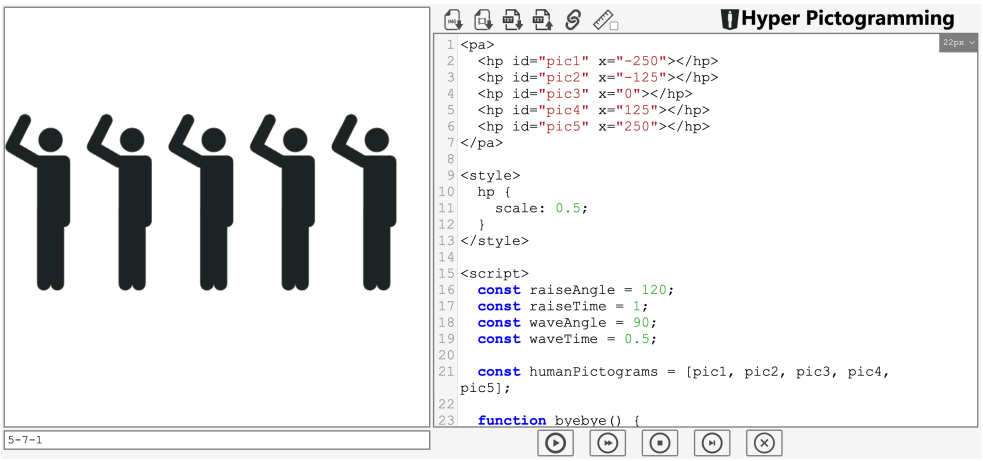
続いては，5 人でバイバイをするようにしたいです．先ほどと同じように「`picN.byebye = byebye; picN.byebye();`」としてもいいですが，5 人もいると少々大変です．仮に 100 人でバイバイしたいとなると，もっと大変になってしまいます．そこで，配列を使用します．

配列とは，複数の値を格納するためのデータ構造です．`"[]"`の中に，`","`で区切りながら値を格納できます．`humanPictograms` という配列を用意し，その中に，人型ピクトグラムの識別子を入れていきます．

配列の中の要素を取得するには `upperParts[0]` というように，配列名の後に`"[i]"`とします．なお，先頭の要素を取得するには 1 ではなくて 0 とする必要があるため気をつけてください．2 番目の要素は 1，3 番目の要素は 2，…というように続きます．また，配列の長さを取得するには，`humanPictograms.length` とします．

コード例 5-7-1

```
<pa>
  <hp id="pic1" x="-250"></hp>
  <hp id="pic2" x="-125"></hp>
  <hp id="pic3" x="0"></hp>
  <hp id="pic4" x="125"></hp>
  <hp id="pic5" x="250"></hp>
</pa>

<style>
  hp {
    scale: 0.5;
  }
</style>

<script>
  const raiseAngle = 120;
  const raiseTime = 1;
  const waveAngle = 90;
  const waveTime = 0.5;

  const humanPictograms = [pic1, pic2, pic3, pic4, pic5];

  function byebye() {
    this.RW("LUA", -raiseAngle, raiseTime);

    for (let i = 0; i < 3; i++) {
      this.RW("LLA", -waveAngle, waveTime);
      this.RW("LLA", waveAngle, waveTime);
    }

    this.RW("LUA", raiseAngle, raiseTime);
  }

  for (let i = 0; i < humanPictograms.length; i++) {
    humanPictograms[i].byebye = byebye;
    humanPictograms[i].byebye();
  }
</script>
```



配列と for 文を組み合わせることで，短いコードで複数人でのバイバイができるようになりました.

5.8　メソッドチェーン

最後にメソッドチェーンについて紹介します．メソッドチェーンとは，メソッドを次々と繋げて書く方法です.

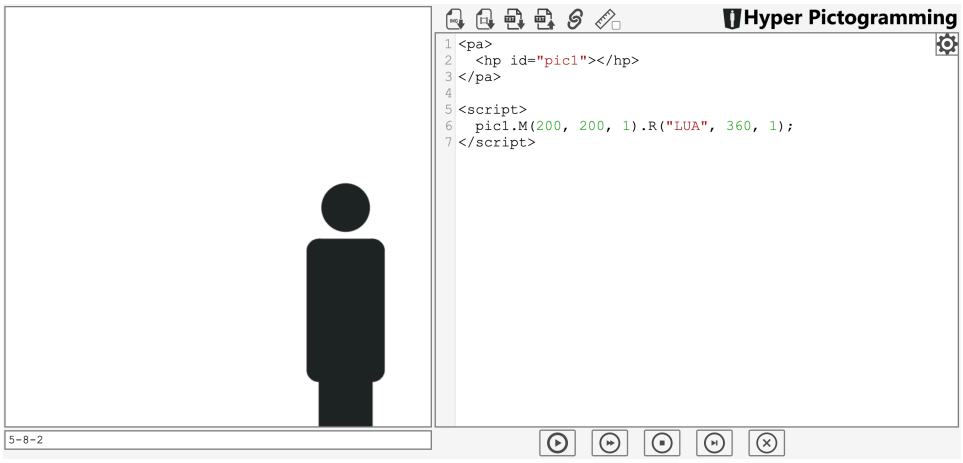
例えば，以下の 2 つのコードは書き方は違いますが，同じ結果になります.

コード例 5-8-1

```
<pa>
  <hp id="pic1"></hp>
</pa>

<script>
  pic1.M(200, 200, 1);
  pic1.R("LUA", 360, 1);
</script>
```

コード例 5-8-2

```
<pa>
  <hp id="pic1"></hp>
</pa>

<script>
  pic1.M(200, 200, 1).R("LUA", 360, 1);
</script>
```



メソッドチェーンを使用することで，コードが簡潔になったり，処理の流れがわかりやすくなったりします.

5.9 練習問題

(1) JavaScript を使用して，「第 3 章　HPML」と「第 4 章　HPML」で作成した歩きスマホ禁止のピクトグラムの中の人型ピクトグラムが実際に歩くようにしてください.

(2) HPML と JavaScript を使用して，動くピクトグラムを作成してください.

第 6 章　ピクトグラフィックス

この章ではピクトグラフィックスを扱います．ピクトグラフィックスは人型ピクトグラムにペンを持たせ，人型ピクトグラムが動くことで，ペンの軌跡を描画することができる機能です．

以下の URL にアクセスしてください．

```
https://pictogramming.org/apps/hyperpictogramming/?pictographics=true
```

今までの画面とほとんど同じですが，コード入力支援ボタン領域の hp のボタン部分が変化します．以下の画像のように，この章で扱うメソッドに対応したボタンが表示されます．

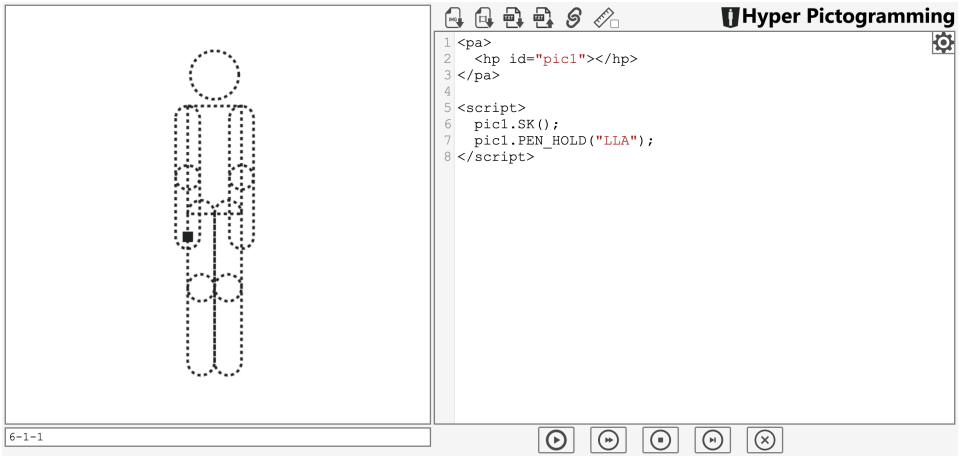
なお，これまでの URL でも本章のメソッドを使用できますし，この URL でも従来の URL に表示されていたメソッドを使用できます．

6.1　ペンを持つ

人型ピクトグラムにペンを持たせてみましょう．まずはペンの軌跡を見やすくするために，SK メソッドで，人型ピクトグラムをスケルトン状態にします．そして，PEN_HOLD メソッドで，ペンを持たせます．

コード例 6-1-1

```
1  <pa>
2    <hp id="pic1"></hp>
3  </pa>
4
5  <script>
6    pic1.SK();
7    pic1.PEN_HOLD("LLA");
8  </script>
```



左手にペンを持たせることができました．PEN_HOLD メソッドの arg1 には，人型ピクトグラムのパーツを指定します．R メソッドや RW メソッドで指定できるパーツに加え，HEAD も指定可能です．

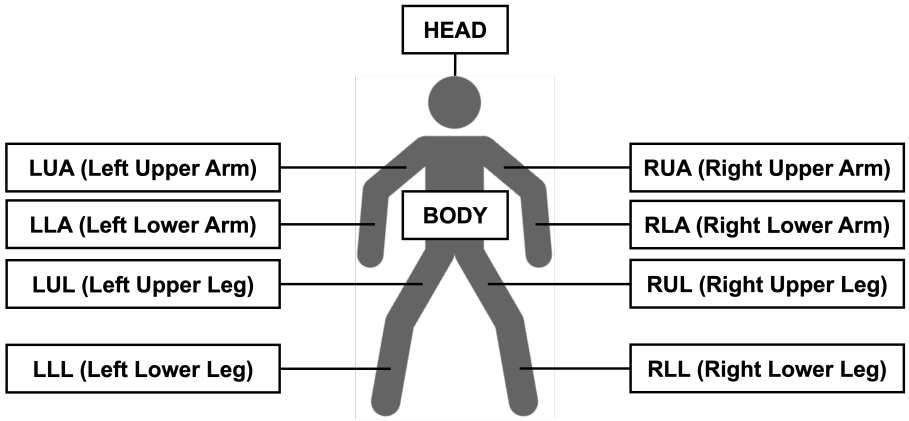


図　人型ピクトグラムのパーツ

ペンを離すための PEN_RELEASE メソッドや，今までペンで描画したものを削除する CS メソッドもあります．

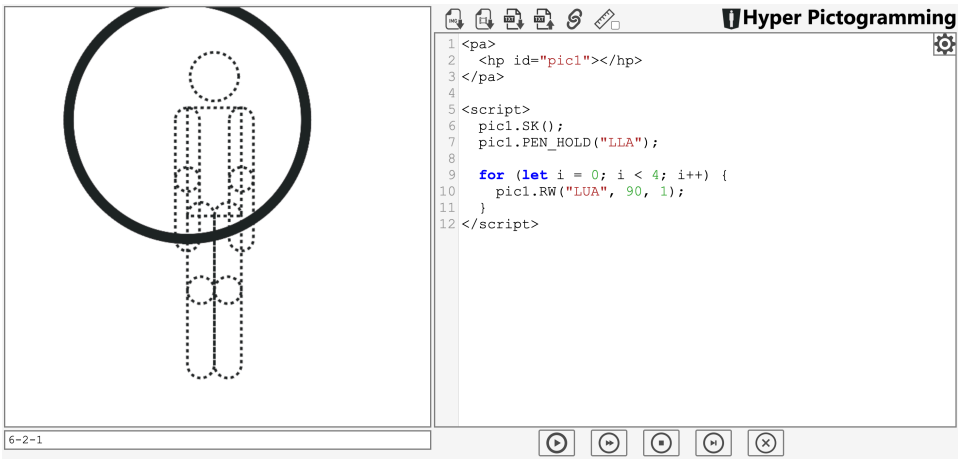
メソッドの様式	処理
*.SK(arg1);	arg1 が省略された場合は，人型ピクトグラムをスケルトンモードに変更する．arg1 が"None"の場合は，人型ピクトグラムを通常モードに変更する．
*.PEN_HOLD(arg1);	arg1 にペンを持つ．
*.PEN_RELEASE(arg1);	arg1 のペンを離す．
*.CS();	ペンによって描画された軌跡を削除する．

6.2　図形を描画する

続いては，人型ピクトグラムを動かして，図形を描画してみましょう．

コード例 6-2-1

```
<pa>
  <hp id="pic1"></hp>
</pa>

<script>
  pic1.SK();
  pic1.PEN_HOLD("LLA");

  for (let i = 0; i < 4; i++) {
    pic1.RW("LUA", 90, 1);
  }
</script>
```

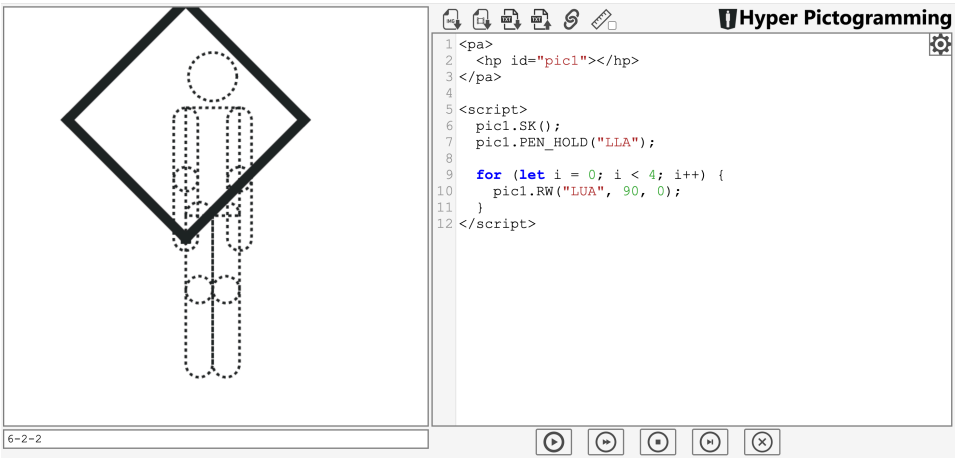


RW メソッドと組み合わせることで，円を描画することができました．

次は，RW メソッドの arg3 を 1 ではなく 0 にしてみましょう．

コード例 6-2-2

```
<pa>
  <hp id="pic1"></hp>
</pa>

<script>
  pic1.SK();
  pic1.PEN_HOLD("LLA");

  for (let i = 0; i < 4; i++) {
    pic1.RW("LUA", 90, 0);
  }
</script>
```

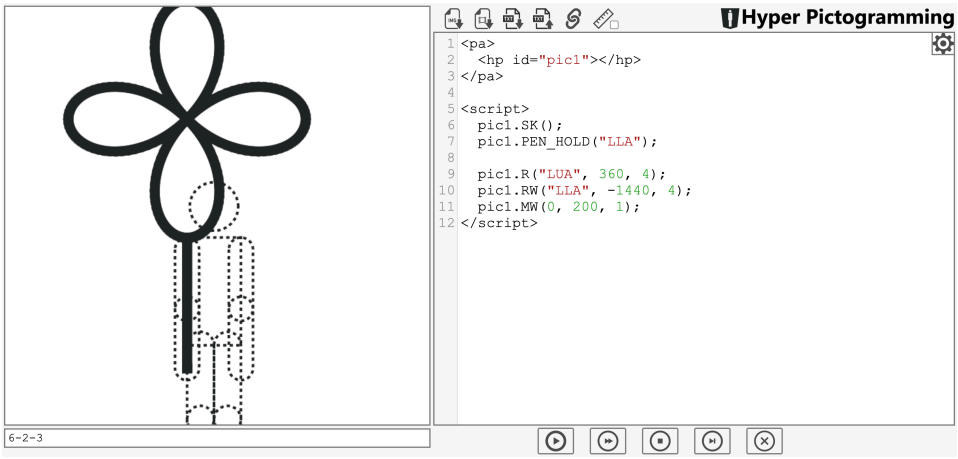


すると，円ではなく四角形が描画されました．これは，arg3 を 0 にすると瞬間移動になるためです．瞬間移動した場合は，移動元と移動先を両端とする直線を描きます．そのため，コード例 7-2-2 は四角形が描画されました．

R，RW，M，MW メソッドなどを使用することで，様々な図形を描画することが可能です．

コード例 6-2-3

```
<pa>
  <hp id="pic1"></hp>
</pa>

<script>
  pic1.SK();
  pic1.PEN_HOLD("LLA");

  pic1.R("LUA", 360, 4);
  pic1.RW("LLA", -1440, 4);
  pic1.MW(0, 200, 1);
</script>
```



四葉のクローバーを描画することができました.

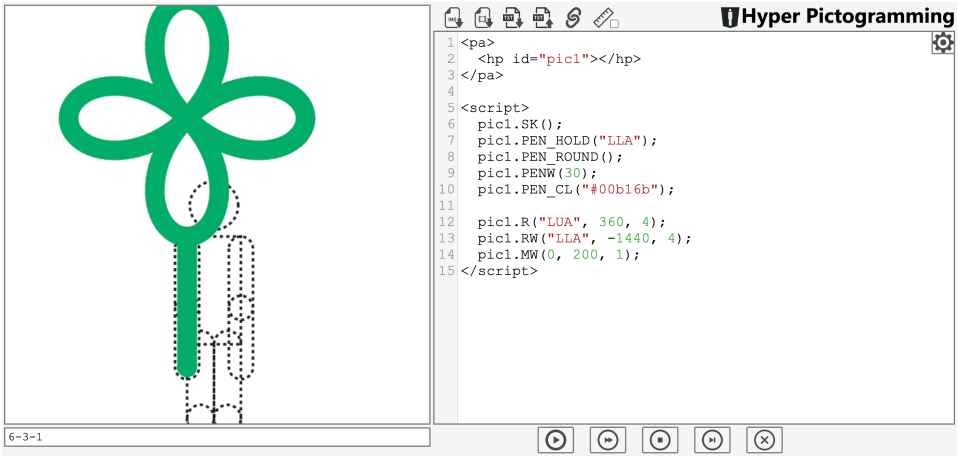
6.3　ペンをカスタマイズする

次はペンをカスタマイズする方法についてです. PEN_SQUARE メソッド, PEN_ROUND メソッド, PEN_BUTT メソッドで線の両端の形状, PENW メソッドでペンの太さ, PEN_CL メソッドで色を変更することが可能です.

先ほどの四葉のクローバーの例をカスタマイズしていきます. 線の両端を半円, ペンの太さを 30, 色を緑色にしてみます.

コード例 6-3-1

```
<pa>
  <hp id="pic1"></hp>
</pa>

<script>
  pic1.SK();
  pic1.PEN_HOLD("LLA");
  pic1.PEN_ROUND();
  pic1.PENW(30);
  pic1.PEN_CL("#00b16b");

  pic1.R("LUA", 360, 4);
  pic1.RW("LLA", -1440, 4);
  pic1.MW(0, 200, 1);
</script>
```



より四葉のクローバーらしくすることができました.

メソッドの様式	処理
*.PEN_SQUARE();	以後描画する線の両端の形状を四角にする.
*.PEN_ROUND();	以後描画する線の両端の形状を半円にする.
*.PEN_BUTT();	以後描画する線の両端の形状を無しにする.
*.PENW(arg1);	ペンの太さ（幅）を arg1 にする. 初期値は 15 である.
*.PEN_CL(arg1);	ペンの色を arg1 にする. 初期値はピクトグラムエリアの type によって異なる.

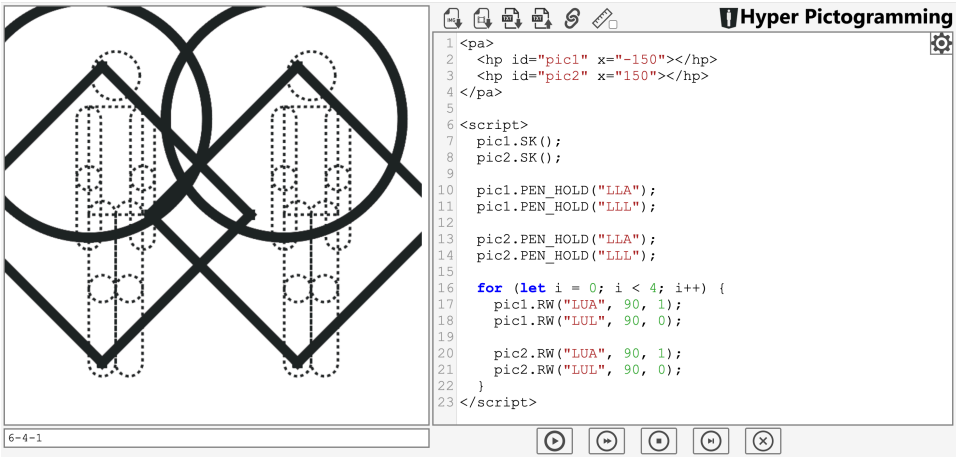
6.4 2 人以上で図形を描画する

ここまでは 1 人の人型ピクトグラムで図形を描画してきましたが，2 人以上でペンを持って図形を描画することもできます. 加えて，左手以外にもペンを持って同時に複数の図形を描画することも可能です.

pic1 と pic2 がそれぞれ，左手で円を，左足で四角形を描画するようにします.

コード例 6-4-1

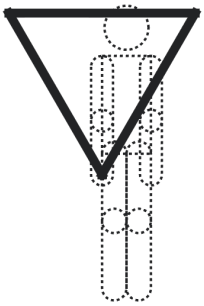
```
<pa>
  <hp id="pic1" x="-150"></hp>
  <hp id="pic2" x="150"></hp>
</pa>

<script>
  pic1.SK();
  pic2.SK();

  pic1.PEN_HOLD("LLA");
  pic1.PEN_HOLD("LLL");

  pic2.PEN_HOLD("LLA");
  pic2.PEN_HOLD("LLL");

  for (let i = 0; i < 4; i++) {
    pic1.RW("LUA", 90, 1);
    pic1.RW("LUL", 90, 0);

    pic2.RW("LUA", 90, 1);
    pic2.RW("LUL", 90, 0);
  }
</script>
```



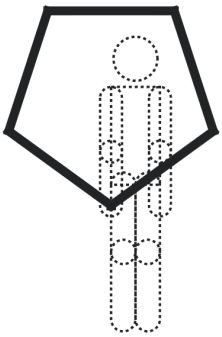
2 人で円と四角形を描画することができました.

6.5　練習問題

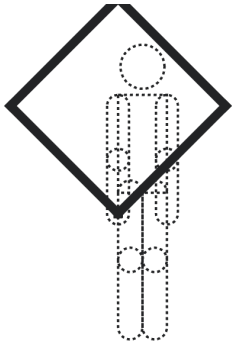
(1) 正三角形を描画してみましょう.

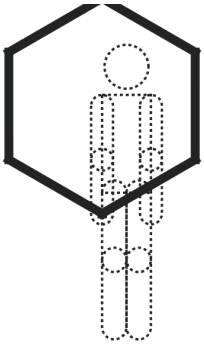


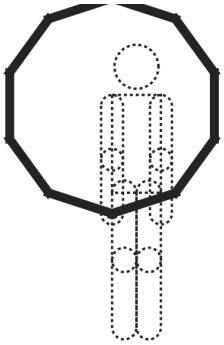
(2) 正五角形を描画してみましょう.



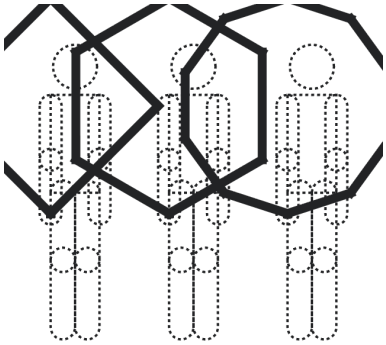
(3) 引数 n の値によって，正 n 角形を描画するための手続きを定義し，使用してみましょう.



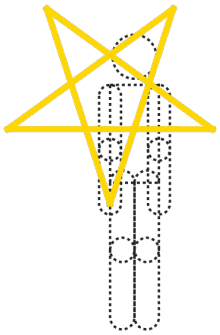




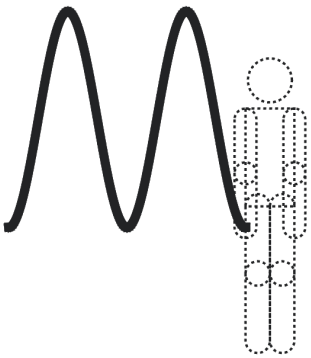
(4) (3)のコードを改良し，複数人で正 n 角形を同時に描画してみましょう．



(5) 星を描画してみましょう．



(6) サインカーブを描画してみましょう．



(7) 自由にペンで図形を描画してみましょう．

第 7 章　ピクトスイミング

この章では，ピクトスイミングを扱います．ピクトスイミングはその名の通り，人型ピクトグラムを泳がせる機能です．

以下の URL にアクセスしてください．

```
https://pictogramming.org/apps/hyperpictogramming/?pictoswimming=true
```

今までの画面とほとんど同じですが，コード入力支援ボタン領域の hp のボタン部分が変化します．以下の画像のように，この章で扱うメソッドに対応したボタンが表示されます．

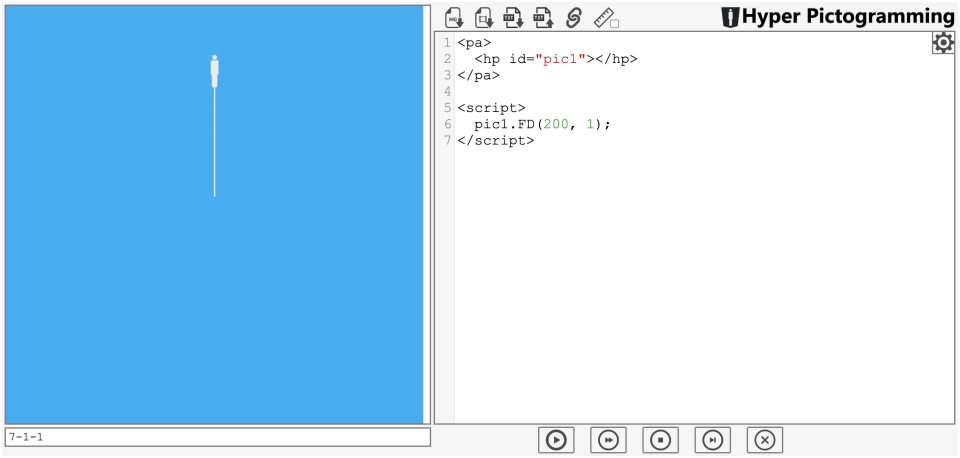
なお，これまでの URL でも本章のメソッドを使用できますし，この URL でも従来の URL に表示されていたメソッドを使用できます．

7.1　泳ぐ

前に向かって泳いでみます．前に向かって泳ぐには，FD メソッドか FDW メソッドを使用します．FD は Forward の略です．どちらのメソッドも arg1 が泳ぐ距離，arg2 が泳ぐのにかかる時間です．

コード例 7-1-1

```
<pa>
  <hp id="pic1"></hp>
</pa>

<script>
  pic1.FD(200, 1);
</script>
```



人型ピクトグラムが 1 秒かけて 100 前向きに泳ぎました.

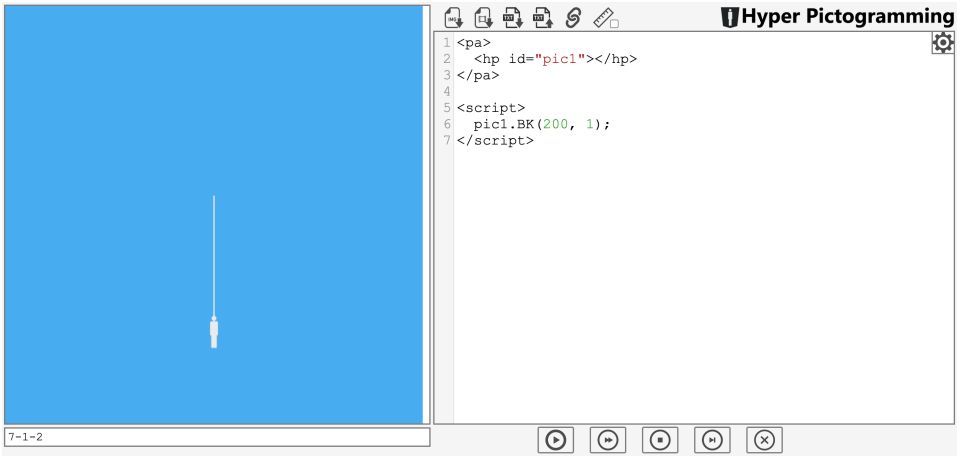
R メソッドと RW メソッドの関係と同様に，FD メソッドは泳ぎと並行して次の命令も実行される，FDW メソッドは泳ぎが完了するまで次の命令は実行されないという違いがあります.

メソッドの様式	処理
*.FD(arg1, arg2);	人型ピクトグラムを進行方向に arg2 秒かけて距離 arg1 だけ等速で進める．arg2 が 省略された時は，arg2 に 0 が入力されているものとして取り扱う.
*.FDW(arg1, arg2);	人型ピクトグラムを進行方向に arg2 秒かけて距離 arg1 だけ等速で進める．移動が終了するまで次の命令は実行されない．arg2 が 省略された時は，arg2 に 0 が入力されているものとして取り扱う.

後ろ向きに泳ぐ BK メソッドと BKW メソッドもあります．BK は Backward の略です.

コード例 7-1-2

```
1 <pa>
2   <hp id="pic1"></hp>
3 </pa>
4
5 <script>
6   pic1.BK(200, 1);
7 </script>
```



後ろ向きに泳ぐことができました.

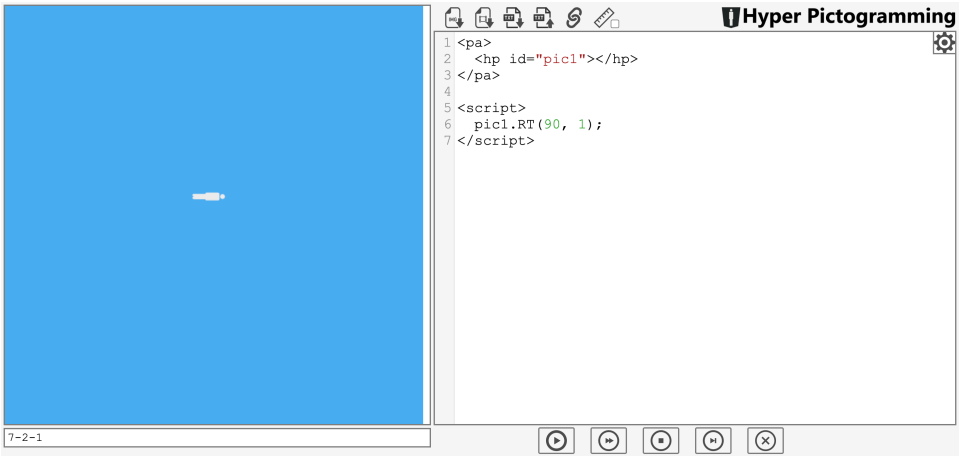
メソッドの様式	処理
*.BK(arg1, arg2);	人型ピクトグラムを進行方向と逆向きに arg2 秒かけて距離 arg1 だけ等速で進める. arg2 が 省略された時は, arg2 に 0 が入力されているものとして取り扱う.
*.BKW(arg1, arg2);	人型ピクトグラムを進行方向と逆向きに arg2 秒かけて距離 arg1 だけ等速で進める. 移動が終了するまで次の命令は実行されない. arg2 が 省略された時は, arg2 に 0 が入力されているものとして取り扱う.

7.2 回転する

次は人型ピクトグラムを回転させてみます. 左向きに回転するには LT メソッドか LTW メソッド, 右向きに回転するには RT メソッドか RTW メソッドを使用します. LT は Left Turn, RT は Right Turn の略です. いずれも arg1 が回転する角度, arg2 が回転するのにかかる時間です.

コード例 7-2-1

```
<pa>
  <hp id="pic1"></hp>
</pa>

<script>
  pic1.RT(90, 1);
</script>
```



1 秒で 90 度右向きに回転することができました.

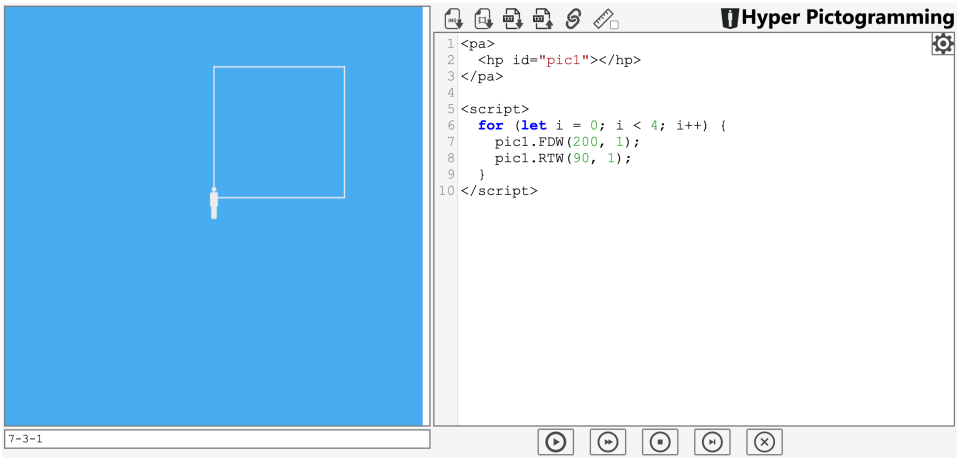
メソッドの様式	処理
*.RT(arg1, arg2);	人型ピクトグラムを arg2 秒かけて時計回り方向に角度 arg1 だけ等角速度で回転する．arg2 が 省略された時は，arg2 に 0 が入力されているものとして取り扱う.
*.RTW(arg1, arg2);	人型ピクトグラムを arg2 秒かけて時計回り方向に角度 arg1 だけ等角速度で回転する．回転が終了するまで次の命令は実行されない．arg2 が 省略された時は，arg2 に 0 が入力されているものとして取り扱う.
*.LT(arg1, arg2);	人型ピクトグラムを arg2 秒かけて反時計回り方向に角度 arg1 だけ等角速度で回転する．arg2 が 省略された時は，arg2 に 0 が入力されているものとして取り扱う.
*.LTW(arg1, arg2);	人型ピクトグラムを arg2 秒かけて反時計回り方向に角度 arg1 だけ等角速度で回転する．回転が終了するまで次の命令は実行されない．arg2 が 省略された時は，arg2 に 0 が入力されているものとして取り扱う.

7.3　泳ぎと回転を組み合わせる

続いてはここまでに扱った泳ぎと回転のメソッドを組み合わせてみます.

コード例 7-3-1

```
<pa>
  <hp id="pic1"></hp>
</pa>

<script>
  for (let i = 0; i < 4; i++) {
    pic1.FDW(200, 1);
    pic1.RTW(90, 1);
  }
</script>
```

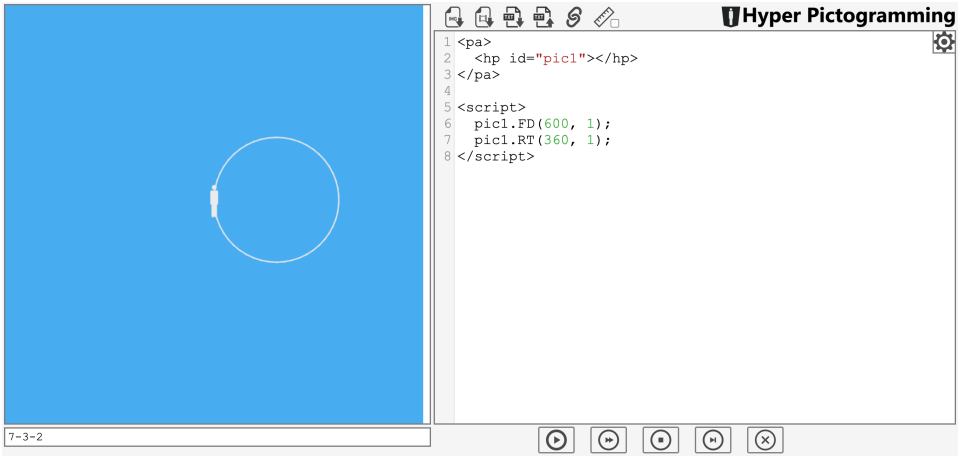


FDW メソッドと RTW メソッドを使用することで，正方形を描画することができました．

もう 1 つ例を見てみましょう．

コード例 7-3-2

```
<pa>
  <hp id="pic1"></hp>
</pa>

<script>
  pic1.FD(600, 1);
  pic1.RT(360, 1);
</script>
```



FD メソッドと RT メソッドを使用することで，円を描画することができました．泳ぐのと同時に回転しているため，このように円になります．

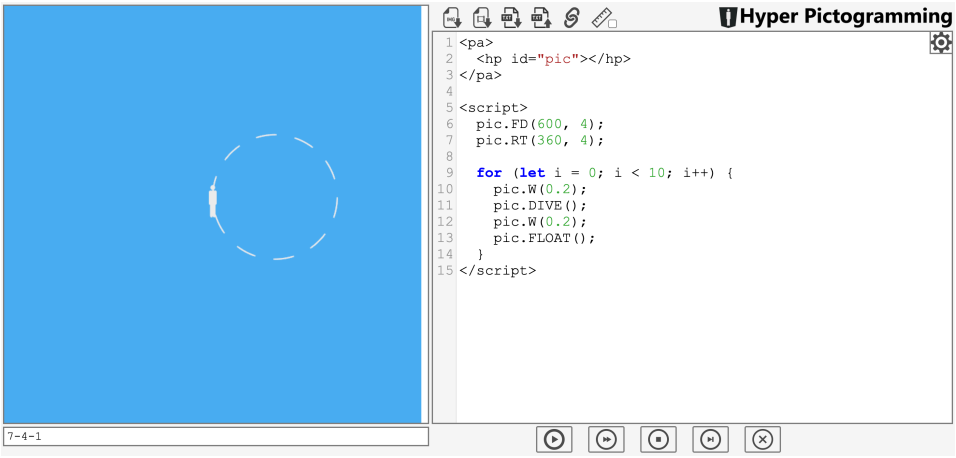
7.4　浮上する・潜る

人型ピクトグラムを水面に浮上させる FLOAT メソッド，水中に潜らせる DIVE メソッドも存在します．

FLOAT メソッドと DIVE メソッドを組み合わせ，点線の円を描画してみましょう．

コード例 7-4-1

```
<pa>
  <hp id="pic"></hp>
</pa>

<script>
  pic.FD(600, 4);
  pic.RT(360, 4);

  for (let i = 0; i < 10; i++) {
    pic.W(0.2);
    pic.DIVE();
    pic.W(0.2);
    pic.FLOAT();
  }
</script>
```



潜るのと浮上するのを繰り返すことで，点線の円を描画することができました．

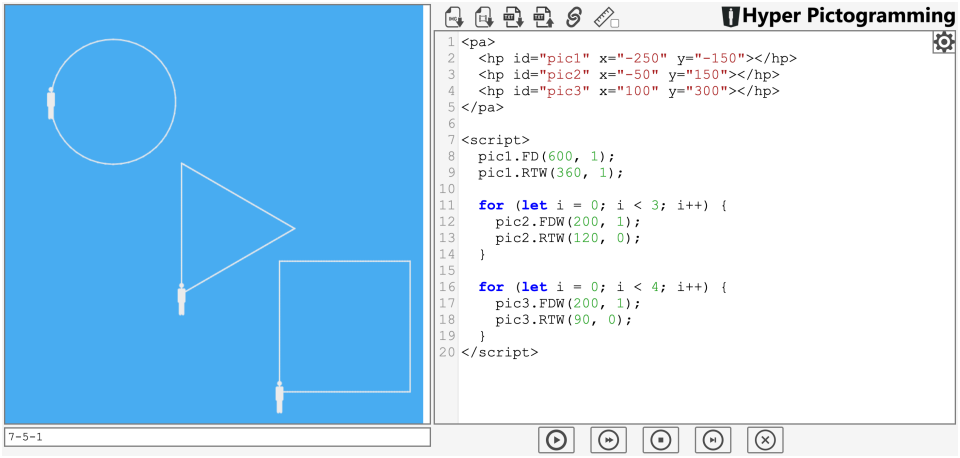
メソッドの様式	処理
*.FLOAT();	人型ピクトグラムを水面に浮上させる．
*.DIVE();	人型ピクトグラムを水中に潜らせる．

7.5　みんなで泳ぐ

ピクトスイミングに関しても複数人で同時に動くことが可能です．

コード例 7-5-1

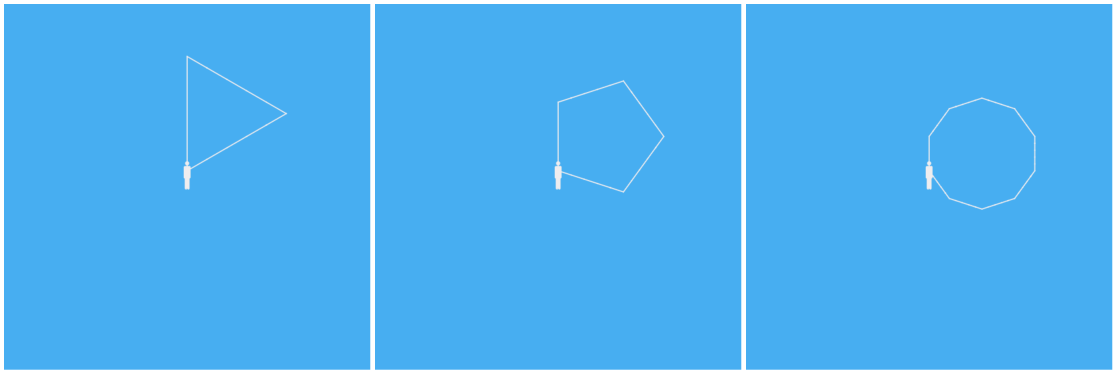
```
<pa>
  <hp id="pic1" x="-250" y="-150"></hp>
  <hp id="pic2" x="-50" y="150"></hp>
  <hp id="pic3" x="100" y="300"></hp>
</pa>

<script>
  pic1.FD(600, 1);
  pic1.RTW(360, 1);

  for (let i = 0; i < 3; i++) {
    pic2.FDW(200, 1);
    pic2.RTW(120, 0);
  }

  for (let i = 0; i < 4; i++) {
    pic3.FDW(200, 1);
    pic3.RTW(90, 0);
  }
</script>
```



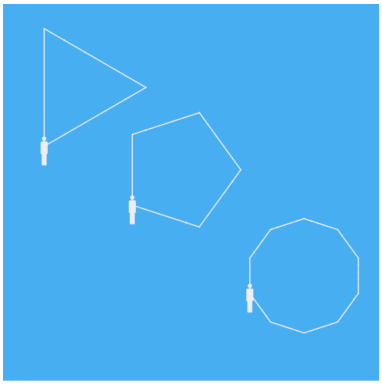
3 人でそれぞれ丸，三角形，正方形を描くことができました．

7.6　練習問題

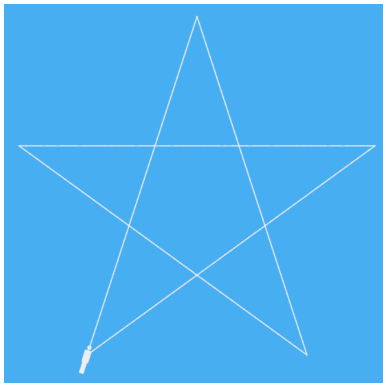
(1) 引数 n の値によって，正 n 角形を泳ぎの軌跡で描画するための手続きを定義し，使用してみましょう．なお，n の値によって 1 辺の長さが変わるようにしましょう．



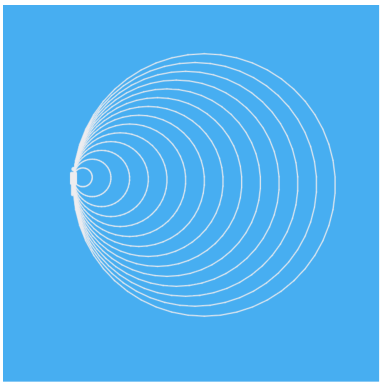
(2) (1)のコードを改良し，複数人で正 n 角形を同時に描画してみましょう．



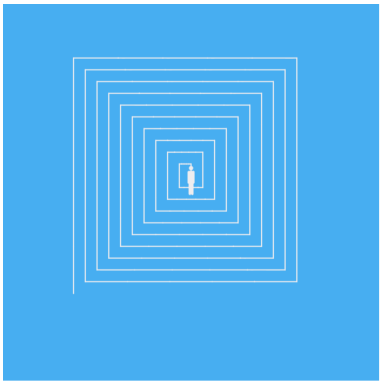
(3) 星を描画してみましょう．



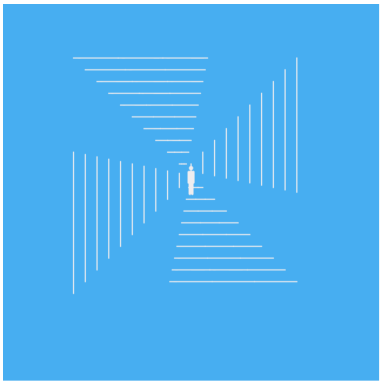
(4) だんだん円の大きさを小さく（あるいは大きく）しましょう．



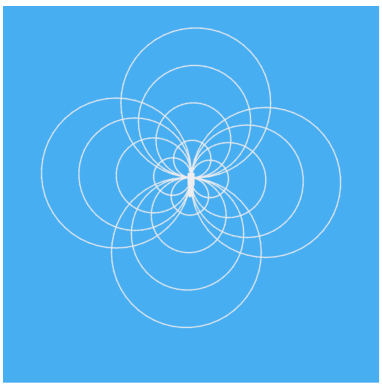
(5) 四角形の渦巻き模様を描画してみましょう．



(6) (5)のコードを改良し，以下のような図形を描画してみましょう．



(7) 4 人で花を描画してみましょう．



(8) 自由に泳いでみましょう．

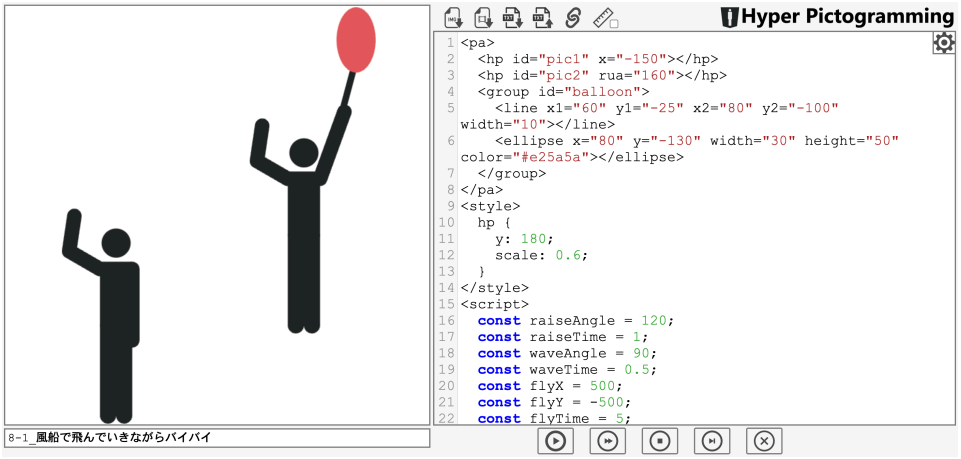
第 8 章　おわりに

ここまで，ピクトグラムを作成しながら，HPML，CSS，JavaScript の基礎について学んできました．次は，以下のようなオリジナルのピクトグラムを作成してみましょう．

コード例 8-1

```
<pa>
  <hp id="pic1" x="-150"></hp>
  <hp id="pic2" rua="160"></hp>
  <group id="balloon">
    <line x1="60" y1="-25" x2="80" y2="-100" width="10"></line>
    <ellipse x="80" y="-130" width="30" height="50" color="#e25a5a"></ellipse>
  </group>
</pa>
<style>
  hp {
    y: 180;
    scale: 0.6;
  }
</style>
<script>
  const raiseAngle = 120;
  const raiseTime = 1;
  const waveAngle = 90;
  const waveTime = 0.5;
  const flyX = 400;
  const flyY = -400;
  const flyTime = 5;

  pic1.byebye = byebye;
  pic1.byebye();
  pic2.fly = fly;
  pic2.fly();
  pic2.byebye = byebye;
  pic2.byebye();
  balloon.fly = fly;
  balloon.fly();

  function byebye() {
    this.RW("LUA", -raiseAngle, raiseTime);

    for (let i = 0; i < 5; i++) {
      this.RW("LLA", -waveAngle, waveTime);
      this.RW("LLA", waveAngle, waveTime);
    }

    this.RW("LUA", raiseAngle, raiseTime);
  }

  function fly() {
    this.M(flyX, flyY, flyTime);
```

```
  }
</script>
```



これは 2 人の人型ピクトグラムの内，1 人が風船で飛んでいきながら，1 人はその場で，バイバイをするアニメーションのピクトグラムです．

ピクトグラムを使うと上手く情報を伝えることができそうなものはありますか？ここまでに学習したことを使用して，オリジナルのピクトグラムを作成してみましょう．