書店で見かけたScratchの入門書が簡単すぎるので誰からも頼まれていませんが勝手にScratchについて解説します。

TurboWarp 狂の副部長

2023年9月10日

皆さんこんにちは。Scratchを(おそらく)極めた、やる気が出ない副部長です。タイトルが一部のああいうラノベのように長いですね。この記事では今後も長ったらしい文、また文法がおかしい文、読みにくい文がずっと登場しますので苦手な方は次の記事へ進んでくれてかまいません。

さて、本題に入ります。いつのことだか忘れましたが、私がUnityとCsharpについて学び始めたころ(たったのおよそ1、２ヶ月前ですね)、それに関する参考書がないかな〜と近くの書店の「プログラミング」というコーナーに寄りました。いい感じの解説書は無事手に入りましたが、とある参考書が私の目に留まったんですよね。そう、今回の本題のきっかけ(悪口・愚痴の対象でもある)となったScratchの参考書です。その場でパラパラとめくってみました。簡単。簡単すぎる。読み応え0。つまらないので別の参考書も手に取ってみました。説明は省略します。これでは実力のあるScratcher(Scratchをする人)は生まれません。~~だから今のScratchは民度が低いわけですね。~~
だからScratchが得意という人が少ないんですね。

というわけで前置きが馬鹿みたいに長くなりましたが、今回はScratchの概要、基本的な機能、参考書で語られていない部分、闇の部分を~~さらけ出して~~紹介して行きたいと思います。皆さんの待ちに待った目次をどうぞ。

目次

1 Scratchの基礎知識

1.1 概要

まずScratchとはいったい何なのか。まあこのような説明が必要な方は基礎を一から学ばれた方がよろしいと思うので、ぜひ書店に足を運ぶなり、ネットで記事を読むなりしたほうが分かりやすく学べると思いますが、この章ではそんな時間や余裕のない方のために、簡単に説明をしていこうと思います。

Scratchは一種のプログラミング言語であり、その非常にわかりやすいUI、手軽な操作により様々な教育現場で使われています。あいにく本郷では講座はありません。Scratchの仕組みとしては、「スプライト」と呼ばれる2Dのオブジェクトを編集しステージ上で動かすというのがメインです。エディター画面(図1)にて、左側のパレットからブロックをドラッグ&ドロップ

してスクリプトを組み立てていったり、「コスチューム」とよばれるスプライトのスキンを編集したりするといった感じです。

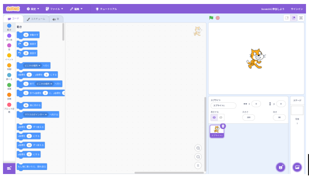

図 1

組み立てたスクリプトは、クリックするか、「旗が押されたとき」というイベントカテゴリ (後述) のブロックを先頭にくっつけてステージの上の緑色の旗を押すと実行されます。少し細かい話をしますと、Scratch は JavaScript で記述されており、30fps で動きます。また、拡張機能を使うと LEGO と連動したり、カメラで動きを検知したりすることもできます。

まあ、こんな感じで Scratch はキーボードがなくてもマウスだけでプログラミングができるという破格の性能 (無料だけど) をしています。

1.2 TurboWarp とは

ここからは布教活動となりますので、興味のない方は飛ばしてもらって結構です。

それではまず皆さん唱えましょう。Repeat after me. TurboWarp(ターボワープ) は神です。

TurboWarp は GarboMuffin 氏によって製作された Scrtach の Mod です。UI、操作方法、仕組みなどはほぼ同じですが、TurboWarp はプロジェクトの軽量化に加え、フリーズ・クラッシュ防止の Warp タイマー、FPS の変更、Scratch におけるさまざまな制限を解除できる設定など、Scratch と比べて機能が非常に充実しています。さらに、アドオンの設定では、一時停止ボタン、変数マネージャー、ブロックのドロップダウン検索などと地味ですがとてもうれしいっ機能も盛りだくさんです。また、最近知ったのが TurboWarp の独自の拡張機能 (紛らわしいですがアドオンとは別物です) ですが、異常なほど多く、Scratch に既にあるアナログな拡張機能に加え、Gamepad(ゲームコントローラを連携できる)、Files(ファイルのインポート・エクスポートができるブロックを追加)、Runtime Option(通常は自分でいじる必要のある環境設定をスクリプトから調整することが可能)、Sensing+ や Clone+ や Look+(「調べる」や「見た目」というカテゴリとクローン関連のブロックを追加)、Clipping & Blending(画像の切り抜きや描写に関する調整が可能)、Local Storage(ほぼ Cookie) などと言い足りませんが様々なものがあります。ちなみに執筆時点でおよそ 80 個で、まだまだ増えています。そして、それらのほとんどはまさに「痒い所に手が届く」な便利な機能です。まだ Scratch をお使いの方はぜひ TurboWarp への乗り換えをご検討ください。TurboWarp は Scratch と同じ形式でプロジェクトをインポート・エクスポートできるので、データの移動はとても簡単です。

2 Scratch のブロック

さて、お次は Scratch の機能とそれに関連するブロックについてみていきましょう。先ほどお伝えした通り、Scratch はブロックをドラッグ&ドロップしてスクリプトを組み立てていくのですが、書店の参考書にはそれぞれのブロックに対する詳細な説明がありませんでした。これは減点ですね。ということでここから Scratch のすべてのブロックとそのカテゴリに対して説明していこうと思うのですが、Scratch はブロックが多く~~面倒くさい~~スペースがあまりないので、以降、ブロック、使用頻度、説明の順番で簡単に説明していきます。それでは、

```
 ________________
／|     Hmcc     |
| |   (*’▽’)    |
| |    舞昆布     |
 ‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾
```

ゆっくりしていってね！！

↑モニターマン in LaTeX

2.1 動き

動きカテゴリは主にそのスプライトの座標、角度について制御するブロックの集まりです。基本的で直感的に理解しやすいブロックだらけです。

1

(n) 歩動かす

使用頻度×

そのスプライトが現在向いている方向に向かって座標を x だけずらす。たとえば、スプライトの向き=30°、n=2 のとき、このブロックを実行すると x 座標が $\sqrt{3}$、y 座標が 1 変わる。ぶっちゃけ使わない。

2

(n) 度回す

使用頻度○

そのスプライトの現在の向きを n だけ変える。デフォルトで 90°。また、時計回りとその逆で 2 つブロックがありるが、負の値もちゃんと読み取ってくれて動作するのでただただ無駄なだけ。なんで 2 個もあるんだ？

3

(どこかの場所▽) へ行く

使用頻度×

このブロックを実行することで、ランダムな場所に行ったりほかのスプライトやマウスに座標を合わせることができる。代用できるので使わない。

4

x 座標を (n1)、y 座標を (n2) にする

使用頻度△

x と y 座標を変更できるブロック。個別に変更できるほかのブロックがあるので正直いらない。あと個人的にキモい。

5

(n) 秒で (どこかの場所▽) へ行く

使用頻度×

「どこかの場所へ行く」の秒数がついたバージョン。座標で事足りるっつってんだろうが。

6

(n) 秒で x 座標を (n1) に、y 座標を (n2) に変える

使用頻度△

少しまともなやつ。FPS に影響されないので、ラグいプロジェクトや TurboWarp で FPS を変更したプロジェクトで使えるかも。基本代替可。あとキモい

7

(n) 度に向ける

使用頻度○

スプライトの向きを指定する。使うときはめちゃくちゃ重宝する。自分は最近使っていない。

8

(マウスのポインター▽) へ向ける

使用頻度×

座標＆三角関数で物足りる。はい。

9

x 座標を (n) ずつ変える
y 座標を (n) ずつ変える

使用頻度◎

プライトの xy 座標の移動は主にこれを使う。これらのブロックのせいでほかのいくつものブロックが死んだ。とても優秀

10

x 座標を (n) にする
y 座標を (n) にする

使用頻度◎

スプライトの座標の指定。位置の初期化、指定場所への移動などはこれを使う。めちゃくちゃ優秀。

11

もし端についたら、跳ね返る

使用頻度×

謎ブロック。実行された時点でステージの端にぶつかったとき、ぶつかった場所に応じて x 軸または y 軸で向きを反転する。ほんとに謎。

12

回転方法を [左右のみ▽] にする

使用頻度△

左右のみ、回転しない、自由に回転の 3 つから選べる。左右のみの場合、0°～179°はそのまま、180°(-180°)～359°(-1°) はコスチュームの左右が反転した状態解いて反映される。それ以外の 2 つは字面通り。

13

(x 座標)
(y 座標)

使用頻度◎
このスプライトの現在の座標を出力する引数。他スプライトの座標を取得するときは調べるカテゴリのブロックが使えるが、正直これで代用できる

14

(向き)

使用頻度△
このスプライトの向きを出力。正直変数を作ったほうが管理しやすい。

2.2 見た目

見た目カテゴリは、コスチュームの描写、レイヤー、大きさに関連したブロックと、一部の「なぜここに分類したのかブロック」を含みます。横・縦のみの伸縮ができるブロックをずっと求めていましたが、そんな拡張機能が最近 TurboWarp に追加されました。控えめにいって神ですね。

1

(こんにちは！) と (n) 秒言う

使用頻度×
こんなブロック、このカテゴリに入れちゃっていいんですか?

2

(こんにちは！) と言う

使用頻度×
本当にこのカテゴリに入れちゃっていいんですか??

3

(うーん...) と (n) 秒考える

使用頻度×
運営さん、このブロックはこのカテゴリに属していると思っているんですか???

4

(うーん...) と (n) 秒考える

使用頻度×
まじめな話をすると、一応リアルタイムで何かの値をモニターすることはできます。代替可なのでほぼ意味なし。枠のデザイン変更出来たらまだ使い道はあったんですけどね。ちなみに TurboWarp の拡張機能に、これのほぼ上位互換である Animated Text なるものがございまして...

5

コスチュームを (costume ▽) にする

使用頻度◎
スプライトのコスチュームを変更するときに欠かせない。

6

次のコスチュームにする

使用頻度○
「前のコスチュームにする」ブロックもください。そうしたらもっと使ってやりますよ。

7

背景を (背景 1 ▽) にする、次の背景にする

使用頻度×
背景はステージのスキンを指します。そもそもステージを背景目的で使うことなんてめったにないのでお役御免です。

8

大きさを (n) ずつ変える
大きさを (n) %にする

使用頻度◎
スプライトの大きさを制御できる唯一のブロック。10000 %とかにすると当たり判定がおかしくなることがあるので要注意。

9

[色▽] の効果を (n) ずつ変える
[色▽] の効果を (n) にする

使用頻度○
ドロップダウンメニューから選んだ画像効果をかける。色、明るさ、透明度などよく見る効果や渦巻、魚眼レンズなど使い道がよくわからない効果がたくさんある。

10

画像効果をなくす

使用頻度○
一個上のブロックでかけた画像効果をすべて解除。効果をかけまくって何が何だか分からなくなったときに使う。

11

表示する
隠す

使用頻度◎
スプライトを表示・非表示にしてくれる。見えてほしくないスプライトはこれで隠しまし

ょう。

12

[最前面▽] へ移動する
(n) 層 [手前に出す▽]

使用頻度○
Scratch にはレイヤーの概念が存在しており、レイヤー id が大きい方が前面に表示されます。また、1 レイヤーにつき 1 スプライトまでしか配置できません。レイヤーの調節に関しては、TurboWarp の拡張機能「Looks+」を使うことをお勧めします。

13

(コスチュームの [番号▽])

使用頻度○
ドロップダウンメニューから読み取るものを番号か名前か選択できます。このブロックのおかげでコスチューム名をデータを入れる場所として使うことができます。これは余談ですが、Scratch はコードでは a と A を区別できないのに、コスチューム名では区別できます。

14

(背景の [番号▽])

使用頻度×
いうまでもないですね。もはやスペースの無駄

15

(大きさ)

使用頻度△
(向き) 同様変数を作った方が管理しやすいのであまり使わない。

2.3 音

音カテゴリはそのまま音に関するブロックの集まりです。特にこれ以上言うことはないですね。

1

終わるまで (sound ▽) の音を鳴らす

使用頻度○
音が鳴り終わるまで処理が終わらないので、特に音楽 (bgm) を流すときは「ずっと」ブロックと相性がいいので重宝する。

2

(sound ▽) の音を鳴らす

使用頻度○
音を垂れ流したまま次のブロックが処理される。おもに短い効果音などそのまま垂れ流されても困らない場合、または次のブロックが重要で迅速に処理されるべき場合に使う。

3

すべての音を止める

使用頻度○
ほかのスプライトが鳴らしている音もすべて止めるので要注意。

4

[ピッチ▽] の効果を (n) ずつ変える
[ピッチ▽] の効果を (n) にする

使用頻度○
ドロップダウンメニューからピッチ、「左右にパン」を選択可能。0 でデフォルト。ちなみにピッチを 120 に設定することでおよそ 2 倍速、70 でおよそ 1.5 倍速。また-infinity に設定することで (TurboWarp の高度な設定でその他の制限を解除する必要があり) 音を一時停止することもできます。Scratch は無理なのでぜひ TurboWarp を。

5

音の効果をなくす

使用頻度△
「画像効果をなくす」の音版。ピッチのリセットは自分で 0 に設定したほうがなんとなくすっきりするのであまり使わない。

6

音量を (n) ずつ変える
音量を (n) %にする

使用頻度△
音量はスプライトごとに違うので注意。

7

(音量)

使用頻度△
(向き) と (大きさ) と同じなので割愛。

2.4 イベント

さあやってきました、Scratch にて最も重要といえるかもしれないイベンカテゴリです。イベントカテゴリでは、例えば「緑の旗が押された」とか、「メッセージを受け取った」などのイベントをトリガーに、その後ろにくっついているブロックを活

性化させます。このカテゴリがなきゃ Scratch は動きません。なぜなら先頭にイベントカテゴリンのブロックがついていないスクリプトは自動的に動作しないからですね。

1

旗が押されたとき

使用頻度 ほぼ必須◎◎
これがなきゃ Scratch は始まらない！ぐらいに重要で初心者のころからずっと使っているであろうブロック。今更解説する必要などないでしょう。

2

[スペース▽] キーが押されたとき

使用頻度○
おそらくあなたも初心者のころに使ったであろうブロック。ほかの条件と組み合わせる場合＜(スペース▽) キーが押された＞で事足りますが、イベントとして使うときはやはりこのブロックがしっくりきます。ちなみに TurboWarp のアドオンで「キー入力オプションの追加」を ON にすることで、ctrl や shift、Enter や記号キーを感知できるようになります。

3

このスプライトが押されたとき

使用頻度△
「押されたとき」という条件はめちゃくちゃ使うんですけどこれじゃあほかの条件 (例えば変数が特定の値の時のみ) とうまく組み合わせることができないんですよね。なので基本的には＜＜マウスが押された＞かつ＜(マウスのポインター▽) に触れた＞＞で代用することになります。

4

背景が [背景 1 ▽] になったとき

使用頻度×
背景は使わない。

5

[音量] ＞ (n) のとき

使用頻度○
ドロップダウンメニューから音量かタイマーか選べる。一部のイベントブロック (例えばこれ) は、コードペイン (コードを組む場所) に出しただけで常時プログラムが走っている状態になります。赤丸を押しても止まりません。しかし、条件を満たさない限りは実際にはただの見た目の問題なので気にしないでください。ちなみに小技として、タイマー＞ 0 を利用すると、実質的に「赤丸が押されたとき」というブロックを再現することができます。知らない人はぜひチャレンジしてみてください。

6

[メッセージ 1 ▽] を受け取ったとき

使用頻度◎
メッセージは主にスプライト間でのタイミングのやり取りに使われます。普通に便利

7

[メッセージ 1 ▽] を送る
[メッセージ 1 ▽] を送って待つ

使用頻度◎
便利。「送って待つ」のほうは、メッセージを受け取ったスプライトが処理を完了するまで次のブロックに進めないので、受け取るスプライトに先に必要な処理をさせる場合に使える。有能。

2.5 制御

やってまいりました。イベントカテゴリと並ぶほど重要なやつ、それが制御カテゴリです。これを自由自在に使いこなせれるのが中級者 (と勝手に思っています) ので、頑張ってマスターしましょう。

1

(n) 秒待つ

使用頻度◎
初心者の頃は特にお世話になったブロックの一つ。タイミングの調整にめちゃくちゃ使えます。豆知識ですが、正確な秒数は n=1 で約 1.020 秒 (パフォーマンスにもよりますが私の場合は 1.017〜1.023 秒でした)、n=0 で約 0.033 秒 (=30fps のときの１フレーム) ですちなみに何も入力しなくても n=0 として認識されます。

2

(n) 回繰り返す {}

使用頻度◎

もはや解説などいりませんね。こんな使いやすいブロック。ちなみに小技ですが、「このクローンを削除する」を「1 回繰り返す」で挟むとスクリプトの途中でクローンを全消ししてメインのスプライトだけ処理を継続することもできちゃいます。

3

ずっと {}

使用頻度◎
ずっと繰り返したい処理に対して使う。条件付きの場合は中にほかのブロックを挟むといい

4

もし<>なら {}

使用頻度◎
条件分岐第一弾。if-。特定の条件を満たしたときのみこのブロックで囲んだブロックを処理する。

5

もし<>なら{}でなければ {}

使用頻度◎条件分岐第二弾。if-else-。重ね掛け (else の部分にまた if-else-を入れる繰り返し) すると見た目が悪くなるが機能性は素晴らしい。

6

<>まで待つ

使用頻度◎
何かの条件がそろうまで次の処理に移らない。タイミング合わせや、「ずっと {}」とくみあわせて何かの条件があるときは実行しないようにするのが基本的な使い方かと。

7

<>まで繰り返す {}

使用頻度◎
上記のブロックが待っている間に何か処理をしたい場合の使う。便利。

8

[すべてを止める▽]

使用頻度○
ドロメニュから「すべてを止める」「このスクリプトを止める」「このスプライトのほかのスクリプトを止める」が選択可能。すべてを止めると文字通りプロジェクトのすべてが止まるのでほぼ使わない。このスクリプトを止めるは条件分岐にはめ込むのが基本の使い方。スプライトのほかのスクリプトは現在実行している塊以外のスプライト内のスクリプトをすべて止める。「ずっと {}」求められてしまうので要注意。

9

クローンされたとき

使用頻度◎
なんでイベントカテゴリじゃないんですか。メインスプライトでは絶対に動作することのないイベントブロックです。クローンされたときの初期化をさせたり、うしろに「ずっと {}」をつけて削除されるまで所定の処理を行わせるのが一般。

10

(自分自身▽) のクローンを作る

使用頻度◎
変数などのパラメーターや現在のコスチュームや座標などが全く同じクローンを作ります。一般的には識別子として使う変数を 1 ずつ変えながら「(n) 回繰り返す」にはめ込んで使う。実行者がクローンでもクローンを作れます。

11

このクローンを削除する

使用頻度◎
クローンは作ったら必ずいつかは消去しなければクローン上限数 (300 体) に達して死ぬので必須です (TurboWarp の高度な設定で上限をなくせますが重いです)。さっきも書きましたが「(1) 回繰り返す」で挟むとクローンを全消ししてメインスプライトだけ残せます。

定義ブロックは、何個かのブロックを圧縮したものとなります。基本的な使い方としては、何回も使うような同じブロックの集合体を定義ブロックとして圧縮して見た目をよくして総ブロック数を抑えます。引数も設定できるので、

3 制作

リスト 1 Sample

```
1 mes "hello, world!"
```