初版作成　2023.1.30

最終修正　2024.2.2

4.　Scratchによる物理シミュレーション

薮哲郎

# 目的

　ボールを投げるシミュレーションをする。

# 準備

　右下の「背景を選ぶ」をクリックし「Blue Sky」を選ぶ。ボールのスプライトを新規に追加し、大きさを60にする。。

# 理論

　位置を , 速度を 加速度を で表す。速度は1秒間の移動距離、加速度は1秒間の速度の変化量を表す。加速度は時刻にかかわらず、一定値をとることを仮定する。以下の微分方程式が成立する。

　微分方程式は傾きを表す式だから、現時刻の値が分かっているとき、時刻 後の値は次式で得られる。

　scratchでは時刻の正確な制御はできない（「例えば0.1秒待つ」のブロックが正確に0.1秒待つわけではない）ので、以後はScratchの世界での数式を考える。

　変数として速度 vx, vyを新規作成する。加速度 ax, ayは定数なので、変数にする必要はない。位置x, yはスプライト毎にあらかじめ設定されている。

　Scratchの世界では「速度はループ1回あたりの移動量 (単位はpx)」「加速度はループ1回あたりの速度の増加量 (単位はpx)」である。次式が成立する。

x = x + vx

y = y + vy

vx = vx + ax

vy = vy + ay

　ボールを投げるとき、水平方向（x方向）の加速度は0、垂直方向（y方向）の加速度は重力加速度である。scratchの世界の加速度はループ1回あたりの速度の増加量を表し、1以下が適切な値である。

# 軌跡の描き方

　左下の「拡張機能を追加」アイコンをクリックして「ペン」を追加する。

　「ペンを下ろす」「ペンを上げる」「全部消す」を使う。

# 作成するプログラム

(1) スタート時：ボールを左下位置に設置する。

(2) スペースキーが押された：ボールの移動をシミュレートする無限ループを実行する。

（ヒント）投げ上げる角度がθ、初速度がvのとき、速度の初期値は次式で得られる。

　　　vx = v \* cos(theta)

　　　vy = v \* sin(theta)

　　　角度はdegreeで指定する。

　重力加速度、初速度を適切な値に設定し、ボールが画面からはみ出ないようにする。

# 注意

　微分方程式を解く方法として、ここで示したように数値を計算して解くことを「数値計算（数値シミュレーション）」と言う。微分方程式を解析的に解いて

を導出する方法とは、**似て非なるものである！！！**解析的に解くのと、数値的に解くのは考え方も方法も全く異なる。数値シミュレーションのコードにおいて、 のような項が出てきたなら、その方法は誤っている。それは解析的に解いているのであり、数値シミュレーションではない。

　数値シミュレーションは芸術である（Numerical Recipes in Cからの受け売り。訳書では技芸という言葉を使っている）。ここで使用した定式化はオイラー法と呼ばれる方法であり、最も単純な方法である。精度が悪いので、周期現象を解析すると振幅が徐々に大きくなるなどの問題がある。微分方程式を安定かつ高速に解くために、ざまざまな方法が提案されている。詳しくは数値計算の教科書を参照してほしい。

# 課題

　ボールを投げるシミュレーションのコードを作りなさい。

（提出方法）

(1)「ファイル」→「コンピュータに保存する」でコードをローカルのPCに保存する。ファイルの拡張子はsb3である。

(2) Teamsにアップロードする

# 発展

　ボールがバウンドする現象もシミュレートする。やや不正確であるが、ボールが地面の中に入ることを防止するため、バウンドは以下のように処理する。y座標のみを考えている。

　y1は地面の境界のy座標である。時刻3のときのy座標は3' ではなく3に設定する。時刻2での速度をvyとするとき、時刻3での速度は完全弾性衝突のとき-vyである。反射係数が a のとき、-a \* vy である。

# バウンド処理のサンプル

グラフィカル ユーザー インターフェイス, グラフ

自動的に生成された説明y < -135なら、vyを反転させる。

-1 \* vyという式を使っているが、

0 - vyでもよい。

少し強引だが、地面にのめり込むのを防ぐために、強制的にy座標を

-133 にする。

「もし」の位置を速度更新の後に配置すると、振幅が徐々に増大するという現象が起こる。

# 応用編：ハードル走ゲームを作る

# ゲーム用画面



# 基本プログラム



# ジャンプ中はキー入力を受け付けない



# ハードルのコスチュームとコード



あとは、ネコがハードルに当たったときに、当たり回数をカウントする変数を追加し、制限時間を付ければ完成。