



Ruby/SDLで お気軽っぽい ゲームプログラミング

第6回関西Ruby勉強会

2005.10.08

鈴木 一範(京大マイコンクラブ)



今日の内容

- 今日の話し手はこんなですよ~
- SDL、Ruby/SDLとはなんぞや?
- まずはデモでも見てみましょう
- Ruby/SDLで何が出来るの?
- じゃあ、ゲームはどうやって作るの?
- これからのRuby/SDL



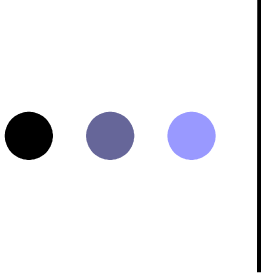
簡単に自己紹介 その1

- 本職は京都大学の学部生(情報学科)
 - ◆授業ではC言語とかやっています
- プログラムを始めたのは高校から
 - ◆高校の時はVB2.0&Win3.1でゲーム作ってた
- Rubyを始めたのは大学の2回生から
 - ◆ohaiさん(Ruby/SDLの作者)に勧められて
 - ◆その後学園祭などでゲームを展示したり
コミケに出展しています



簡単に自己紹介 その2

- Linux歴は3年ぐらい
 - ◆ せいぜい自宅サーバでちょこちょこ遊ぶ程度
 - ◆ Railsをやってみたいと思ってたりする
- チャリ(自転車)に乗るのが好き
 - ◆ 今日30分ぐらいチャリを飛ばして来ました



SDLとはなんぞや その1

- SDL・・・Simple DirectMedia Layerの略
様々なプラットフォームで動くマルチメディア用
ライブラリ(公式サイトより)
→要はDirectXみたいなもの
◆これ1つで音声や映像、キーボードやジョイスティックからの入力処理が可能
- 特徴・・・様々なプラットフォームで動く
Linux,Windows,BeOS,MacOS,Solaris,FreeBSD



SDLとはなんぞや その2

- SDL自体は必要最小限の機能を提供し
ライブラリで様々な機能や足りない機能を提供する

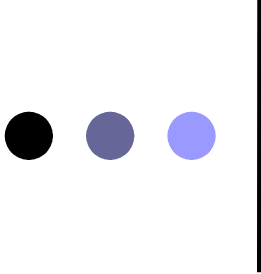
◆例えば

SDL_mixer・・・wavやmp3,oggファイルの演奏

SDL_ttf・・・TrueTypeFontを使って文字描画

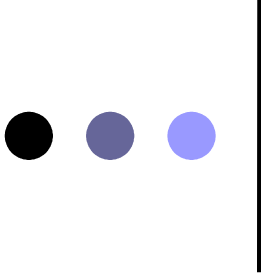
SGE・・・点や線、円などの描画

図形の拡大縮小回転の実現



Ruby/SDLとはなんぞや その1

- その名の通りRubyからSDLを利用する為のライブラリ、つまりRubyでゲームを作るためのライブラリである。
- 実装されている機能
 - ・2D高速描画
 - ・キーボード、マウス、ジョイスティックからの入力機能
 - ・SDL_mixer による音声の再生
 - ・CDの再生
 - ・SDL_TTF によるTrueTypeFontの描画
 - ・SDL_image による各種イメージファイルのロード (bmp,png,jpeg,gifなど)
 - ・SMPGEによるmpegの再生
 - ・SDL_SKKによる、行単位の日本語入力
 - ・OpenGL Interfaceによる、3D描画
 - ・SDL_kanji による、bdfフォントの描画



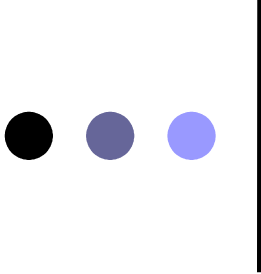
Ruby/SDLとはなんぞや その2

○ 特長

- なんと言ってもRubyが使える
- 様々なプラットフォームで動くゲームが作れる
windows上ならexerbでexeファイル化
それ以外なら直接ソースから実行可能

○ 問題点

- 遅い



Ruby/SDLとはなんぞや その3

○ インストール

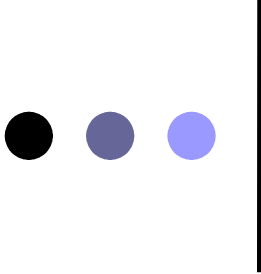
- Windowsならwin32bin-pack.lzhをダウンロードしてパスを通す、あるいは既存のバイナリに必要なファイルをコピー。

(opengl.soもコピーしておくと便利)

- LinuxなどはREADME.jaを読んでください。

SDLやらSDL_ttf、SDL_imageをインストールする必要があるので少し大変かも。

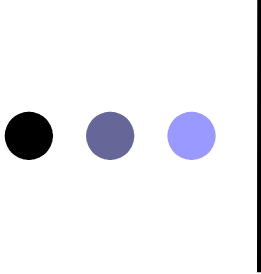
URI...<http://www.kmc.gr.jp/~ohai/rubysdl.html>



まずはデモでも見てみましょう

○ 注意事項

- ・うわ、これマジさむーいと思ってもスルーしましょう。
- ・実際のKMCの雰囲気はもっと健全です、たぶん。
- ・ソースの内容などは理解できなくてもいいので、
とにかくRuby/SDLでこんな物が作れるんだ、こういうことが
出来るんだということを分かっていたらと思います。

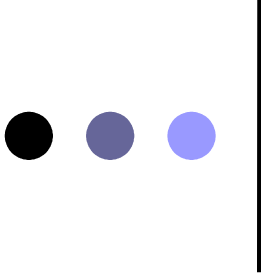


Ruby/SDLで何ができるの?

その1 画像描画

- 画像を描画する方法は2種類。
 - `SDL::Surface#put(image,x,y)`
これは画像を単に置くだけ。
サンプル `screen.put(image, 50,50)`
 - `SDL.blitSurface(src,srcX,srcY,srcW,srcH,dst,dstX,dstY)`
これは画像の一部を切り取って貼り付ける

サンプル
`SDL.blitSurface(image, 50,50,100,100,screen,100,100)`



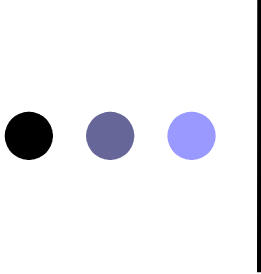
Ruby/SDLで何ができるの?

その1 画像描画

```
require 'sdl'
SDL.init(SDL::INIT_VIDEO)
screen = SDL::setVideoMode(640,480,16,SDL::SWSURFACE)
image = SDL::Surface.load('bmp/chara1.bmp')
image.setColorKey(SDL::SRCCOLORKEY|SDL::RLEACCEL,
                  image.getPixel(0,0))

screen.put(image,50,50)
#SDL.blitSurface(image,50,50,100,100,screen,100,100)
screen.updateRect(0,0,0,0)

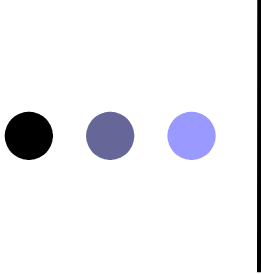
while true
  case event = SDL::Event2.poll
  when SDL::Event2::Quit
    exit!
  end
  SDL.delay(100)
end
```



Ruby/SDLで何ができるの？

その1 画像描画

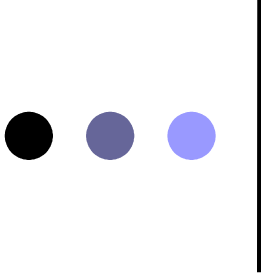
- 画像(サーフェス)に対するメソッド
 - `SDL::Surface#displayFormat`
画像を高速で描画出来るようにする。
 - `SDL::Surface#setAlpha(flag,alpha)`
画像に α 値(透明度、0で透明、255で通常と同等)を設定
 - `SDL::Surface#copyRect(x,y,width,height)`
(x,y,width,height)の長方形を複製したサーフェスを生成
 - `SDL::Surface#transformSurface(bgcolor,angle,xscale,yscale,flags)`
angle度回転し、X方向にxscale倍、Y方向にyscale倍、
生じたすきまをbgcolorで塗りつぶしたサーフェスを生成



Ruby/SDLで何ができるの？

その1 画像描画(図形編)

- 描画出来る図形は以下のようなものがある。
 - 長方形 `SDL::Surface#fillRect(x,y,w,h,color)`
(枠のみ、塗りつぶし、αブレンド)
 - 線(太さ1px) `SDL::Surface#drawLine(x1,y1,x2,y2,color)`
(普通、アンチエイリアス、αブレンド)
 - 円 `SDL::Surface#drawAACircle(x,y,r,color)`
(枠のみ、塗りつぶし、アンチエイリアス、αブレンド)
 - 楕円 `SDL::Surface#drawEllipse(x,y,r,color,alpha)`
(枠のみ、塗りつぶし、アンチエイリアス、αブレンド)

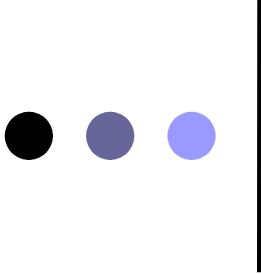


Ruby/SDLで何ができるの?

その1 画像描画(図形編)

○ 逆に以下のような図形は普通には描画出来ない。

- ・円弧
→ 中心、半径を決めて点を描画する
- ・回転した楕円、回転した長方形など
→ サーフエスを作成し、回転させる
- ・太い直線
→ 各点ごとに円を描画する
- ・多角形
→ 外枠だけなら可能、塗りつぶしは今のところ不明
- ・曲線(エルミート曲線とかベジェ曲線、Bスプライン曲線など)
→ ごめんなさい `m(_ _;)`、各自でがんばってください。



Ruby/SDLで何ができるの？

その2 イベント処理

- SDL::EventとSDL::Event2があるが主にSDL::Event2を使う

キーを押した、離れた

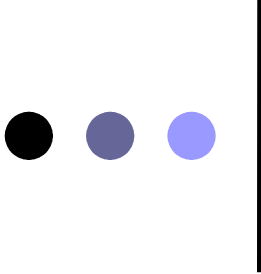
マウスを動かした、押した、離れた

ジョイスティックのスティックを動かした

ジョイスティックのボタンを押した、離れた

プログラムを終了しようとした(×ボタンを押した)

という入力を処理できる。

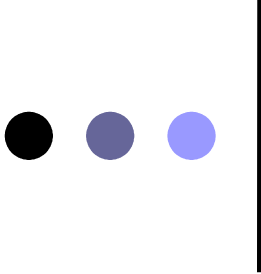


Ruby/SDLで何ができるの?

その2 イベント処理

```
require 'sdl'
SDL.init(SDL::INIT_VIDEO)
screen = SDL::setVideoMode(640,480,16,SDL::SWSURFACE)

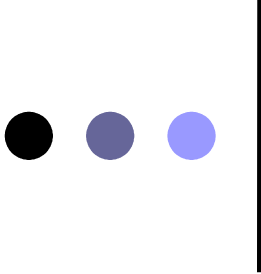
while true
  while event = SDL::Event2.poll
    case event
      when SDL::Event2::Quit
        #終了処理(break、exit!)などを書く
      when SDL::Event2::MouseButtonUp
        if event.button == SDL::Mouse::BUTTON_LEFT then
          #マウスの左ボタンがクリックされた時の処理
        end
      when SDL::Event2::MouseMotion
        #マウスが動いた時の処理
      when SDL::Event2::JoyAxis
        #ジョイスティックのスティックが押された時の処理
    end
  end
end
```



Ruby/SDLで何ができるの?

その2 イベント処理

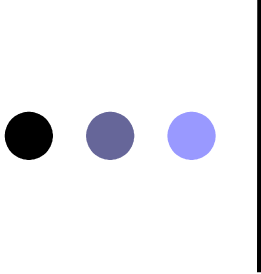
```
when SDL::Event2::KeyDown
  case event.sym #押されたボタンを返す
  when SDL::Key::ESCAPE
    exit!
  when SDL::Key::Z
    #zキーが押された時の処理
  when SDL::Key::UP
    #上のキーが押された時の処理
  when SDL::Key::DOWN
    #下のキーが押された時の処理
  end
end
end
end
end
```



Ruby/SDLで何ができるの？

その2 イベント処理(入力処理)

- しかしSDL::Event2では押した、離したは分かっても押しっぱなしや離しているは分らない。
- そこで直接的にキーやジョイスティックの状態を取得する必要がある
 - SDL::Key.scan
現在のキーの状態(どれが押されているか)を取得
 - SDL::Joystick.updateAll
現在のジョイスティックの状態を取得



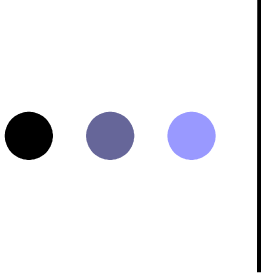
Ruby/SDLで何ができるの?

その2 イベント処理(入力処理)

```
require 'sdl'
SDL.init(SDL::INIT_VIDEO|SDL::INIT_JOYSTICK)
screen = SDL::setVideoMode(640,480,16,SDL::SWSURFACE)
Joystick = SDL::Joystick.open(0) if SDL::Joystick.num != 0

SDL::Joystick.updateAll
SDL::Key.scan

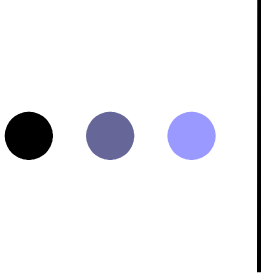
if (SDL::Key.press?( SDL::Key::RIGHT ) ||
    (SDL::Joystick.num != 0 && Joystick.axis(0) > 300 ))
    #右が押しっぱなしでの処理
end
If (SDL::Key.press?( SDL::Key::LEFT ) ||
    (SDL::Joystick.num != 0 && Joystick.axis(0) < -300 ))
    #左が押しっぱなしでの処理
end
```



Ruby/SDLで何ができるの?

その3 音楽再生

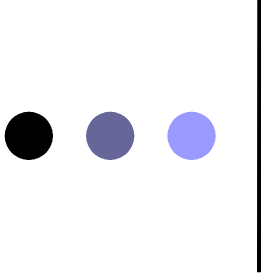
- WAVEファイルとそれ以外(.mp3 .ogg .mid .itなど)で少し扱い方が異なる。
 - ・WAVEファイルの場合
 - SDL::Mixer::Wave.load(*filename*) でファイルをロード
 - SDL::Mixer.playChannel(*channel*,*wave*,*loop*) で再生
 - 以下チャンネルごとに設定する(設定出来る)
 - ・それ以外の場合
 - SDL::Mixer::Music.load(*filename*)でロード
 - SDL::Mixer.playMusic(*music*,*loops*) で再生
 - 1つの音楽のみ処理可能(複数の曲を同時再生できない)



Ruby/SDLで何ができるの?

その3 音楽再生

- 音楽関係のメソッド。
 - `SDL::Mixer.halt(channel)`
 - `SDL::Mixer.pause(channel)`
 - `SDL::Mixer.resume(channel)`
channelのWAVEを止める、一時停止する、再開する
 - `SDL::Mixer.setVolume(channel, volume)`
channelのボリュームを設定(範囲は0から128)
 - `SDL::Mixer.fadeInMusic(music, loops, ms)`
 - `SDL::Mixer.fadeOutMusic(ms)`
musicで指定した音楽をmsで指定したミリ秒かけてフェードイン
同様にmsで指定したミリ秒でフェードアウトする



Ruby/SDLで何ができるの?

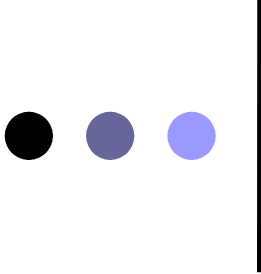
その3 音楽再生

```
require 'sdl'
SDL.init(SDL::INIT_VIDEO|SDL::INIT_AUDIO)
screen = SDL::setVideoMode(640,480,16,SDL::SWSURFACE)
SDL::Mixer.open(44100, SDL::Mixer::DEFAULT_FORMAT,
                SDL::Mixer::DEFAULT_CHANNELS, 4096)

music = SDL::Mixer::Music.load('sound/kitchen.mp3')
hit_wav = SDL::Mixer::Wave.load('sound/enemy_bullet_hit.wav')

SDL::Mixer.playMusic(music, -1) # -1でループ再生

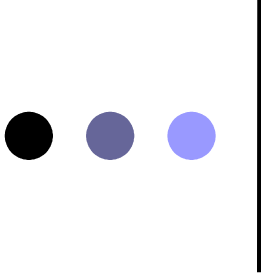
while true
  while event = SDL::Event2.poll
    case event
    when SDL::Event2::Quit
      exit!
```



Ruby/SDLで何ができるの?

その3 音楽再生

```
when SDL::Event2::KeyDown
  case event.sym
  when SDL::Key::UP
    SDL::Mixer.pauseMusic
  when SDL::Key::DOWN
    SDL::Mixer.resumeMusic
  when SDL::Key::LEFT
    SDL::Mixer.playChannel(0,hit_wav,-1)
  when SDL::Key::RIGHT
    SDL::Mixer.halt(0)
  end
end
end
end
end
```

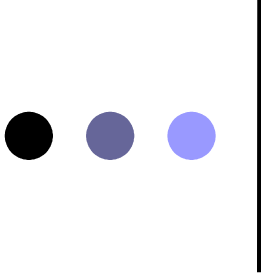



Ruby/SDLで何ができるの？

その4 文字描画

- Ruby/SDLで扱える文字は2種類
(ビットマップフォント、TrueTypeフォント)
→ビットマップフォントは使いづらいので
ここではTrueTypeフォントのみを説明
- ・注意点
英語や数字なら問題ないが日本語の場合だと
文字列をUTF-8にコンバートする必要がある。
(String#toutf8とする事で可能、ただしRuby1.8.2以上で
require 'kconv'しておく)

フォントは著作権を伴うので添付する場合は少し気をつけよう



Ruby/SDLで何ができるの？

その4 文字描画

○ 文字関係のメソッド

- `SDL::TTF#textSize(text)`
textを描画したときの大きさを[幅,高さ]という配列で返す。
- `SDL::TTF#drawBlendedUTF8(dest,text,x,y,r,g,b)`
textをdestの位置(x,y)の所に書きこむ。色はr,g,b。
- `SDL::TTF#renderBlendedUTF8(text,r,g,b)`
textが描画されたサーフェスを生成する

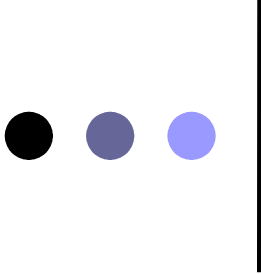


Ruby/SDLで何ができるの?

その4 文字描画

```
require 'sdl'
require 'kconv'
SDL.init(SDL::INIT_VIDEO)
screen = SDL::setVideoMode(640,480,16,SDL::SWSURFACE)
SDL::TTF.init
font = SDL::TTF.open("mikachan-P.ttf",50)
sample_str = "こんにちは、世界!!!"
sample_str = sample_str.toutf8
font.drawBlendedUTF8(screen, sample_str, 100,100, 255,255,255)
screen.updateRect(0,0,0,0)

while true
  case event = SDL::Event2.poll
  when SDL::Event2::Quit
    exit!
  end
  SDL.delay(100)
end
```



Ruby/SDLで何ができるの?

その5 その他の機能

○ SDL::CollisionMap

ふたつの画像が重なっているかどうかを簡単に判定

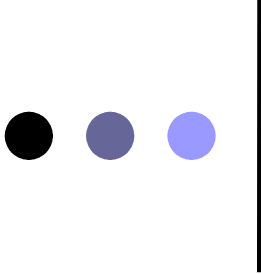
使い方

- ・SDL::Surface#makeCollisionMapでcollision map を作成。
(あらかじめカラーキーを設定出来る)
- ・SDL::CollisionMap#collisionCheck(*x1*, *y1*, *collisionMap*, *x2*, *y2*)
で判定

○ SDL::WM関連

ウィンドウのキャプション、アイコンの設定

- ・SDL::WM.setCaption(*title*,*icon*) など



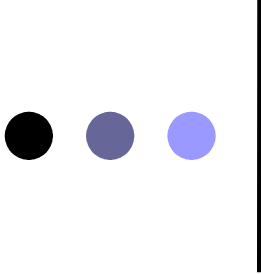
Ruby/SDLで何ができるの？

その5 その他の機能

- ほかにも
 - ・SDL::CD (音楽CDの再生など)
 - ・SDL::MPEG(MPEGファイルの再生など)
 - ・SDL::SKK (キーボードからの日本語入力、ただしSKK)

などがあるが、今回は説明しない。

理由・・・ゲームプログラミングにおいてはそれほど必要ではない
つか、俺も知らないの。

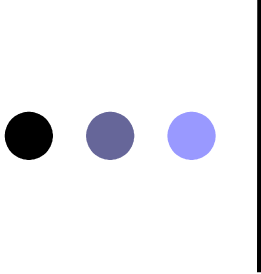


じゃあ、ゲームは どうやって作るの? その1

- 今まで説明してきたのはRuby/SDLの個々の機能。
- じゃあ具体的にはどうやってゲームを作るのか
→基本的にはループでぐるぐる回す

注意点として

- ・プログラムの進行速度(FPS)を保つ必要がある
- ・描画の順番に気をつける
- ・オブジェクト指向を上手に使ってみよう(せっかくRubyなんだし)



じゃあ、ゲームは どうやって作るの? その2

○ Ruby/SDLによるゲームプログラムの基本的な流れ

1,初期化

```
SDL.init(SDL::INIT_VIDEO|SDL::INIT_AUDIO|SDL::INIT_JOYSTICK)
```

```
SDL::TTF.init
```

```
SDL::Mixer.open
```

```
screen = SDL::setVideoMode(640,480,16,SDL::SWSURFACE)
```

```
SDL::WM.setCaptionなど
```

2,初期設定

```
SDL::Surface.load, SDL::Surface#setColorKeyなどで画像を読み込む
```

```
SDL::Mixer::Music.load, SDL::Mixer::Wave.loadなどで音楽を読み込む
```

```
SDL::TTF.openなどでフォントを読み込む
```

(複数の大きさの文字を使いたい場合は複数個設定する)

じゃあ、ゲームは どうやって作るの? その3

3,メインループ

```
while true
```

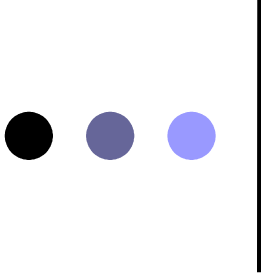
```
  while event = SDL::Event2.poll ... end
  SDL::Key.scan
  その他、ループ1回ごとの処理
```

イベント処理
入力処理

```
  timer.wait_frame do
    screen.fillRect(0, 0, 640, 480, [0, 0, 0])
    screen.put
    screen.updateRect(0,0,0,0)
  end
```

FPS調整
画面全体の消去
画面に描画(順番に注意)
画面全体の更新

```
end
```

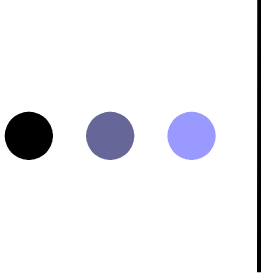



じゃあ、ゲームは どうやって作るの? その4

- プログラムの進行速度(FPS)とは?
 - 1秒間に何回画面を更新させるか
 - つまり、どれだけなめらかな動きをさせるか
 - だいたい60FPS
- 俺の場合はいつもohai先輩のサイトにあるFPSTimerを使用してます(第7章 時間の処理 その2 より http://www.kmc.gr.jp/~ohai/rubysdl_intro.html)。

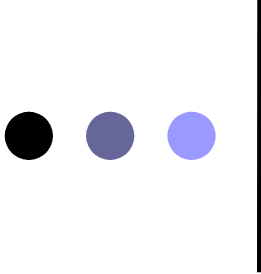
使い方

- ・クラス(FPSTimerSample)を保存
- ・requireで読み込む。
- ・`timer = FPSTimerSample.new` `timer.reset` で
インスタンス生成および動作準備
- ・画像の描画部分を`timer.wait_frame do ~ end` ではさむ



じゃあ、ゲームは どうやって作るの? その5

- 描画の順番に気をつける
 - 基本は
背景→プレイヤーの攻撃→敵→プレイヤー→敵の攻撃→その他
- では、どうやってこの順番で描画させるか?
→ここでオブジェクト指向を使ってみよう!!!



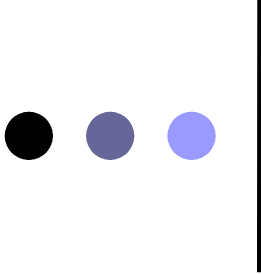
じゃあ、ゲームは どうやって作るの? その6

○ オブジェクト指向を上手に使ってみよう

- 1,画面上に存在する物を背景、プレイヤーの攻撃、敵、プレイヤー、敵の攻撃、その他に分類してクラス化する
- 2,上の分類(背景、プレイヤーの攻撃など)ごとに配列を作り、挿入する
- 3,上の分類を1つの配列に挿入する(順番は上の通りに)
- 4,

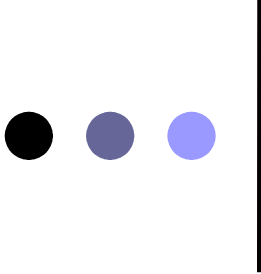
```
object.each do |sub_object|  
  sub_object.each do |contents|  
    クラス内のメソッド  
  end  
end
```


これを使ってすべてのクラスについて処理を行う。



ちょっとしたテクニックっぽい 何か その1

- なんだかんだ言ってもやっぱり遅い画像処理
(マシンパワーに物を言わせる手もあるが)
→やっぱある程度はプログラムを速くしておきたい
 - ・回転処理(SDL::Surface#transformSurface)は重い
→あらかじめ回転しておいた画像(1度ずつ)を用意しておく
 - ・大きい画像(1枚絵の背景など)はネックになる
→適度にごまかす(背景無し、線や円を使う等)
 - ・かといって大量に図形を描画するとそれはそれで遅い
→がんばれ!!!
- Windowsユーザはバイナリが欲しい
 - ・いちいちRuby/SDLをインストールする人はそうはいない
→exerbというツールを使ってexe化しておく



ちょっとしたテクニックっぽい 何か その2

- まあ、Ruby自体も遅い(C言語とかと比べると)
→マシンパワーに頼るとバランスがおかしくなる場合もある
 - ・そのあたりを考えると必ずしもRuby/SDLによる
ゲームプログラミングがお気軽なものとは言えないかもしれない。
 - ・そういうニュアンスを込めて、今回の講座のタイトルを
Ruby/SDLでお気軽っぽいゲームプログラミング
としてみました。
 - ・でもいろいろ工夫すれば普通に60FPSぐらい出る
かっちょええゲームも作れるんですけどね。



これからのRuby/SDL

- やっぱりRuby/SDLユーザを増やしたい!!!
 - ・そのためにはRuby/SDLのおもしろさを
もっと多くの人に伝える必要がある。
 - ・しかし、現状ではWeb上にサンプルプログラムが少ないので
おもしろさを知っている人がいろいろとがんばらないと。
(とは思っているがなかなか行動に移せない)

というわけで皆さん、
是非Ruby/SDLでいろいろ遊んでみましょう。