基礎プログラミングおよび演習＃06

1820004, 大竹　春生

2018/11/11

構想

課題 X

自分 (達) の「美しい」と思う絵を生成するプログラムを作成しなさい。「美しい」の定義は各自にお任せしますので、自分のレベルに合った内容で結構です。レポートを重視するので、きちんとどのように美しいか書いてください。レポートは次の順で記述してください

目標

オブジェクト指向的な設計指針を採用し、簡素で拡張性の高いコードを実装する。

計画

要件

要件 1

- 最も抽象度の高い図形を生成する処理はクラスとして切り出す。[1]
- 図形クラスのメソッドを直接呼出し画像を生成するクラスを実装する。

[1] 図形クラス

要件 2

- 最も抽象度の高い図形を生成する処理はクラスとして切り出す。[2]
- 図形クラスのメソッドを呼び出し風景画像へ配置されるオブジェクトを生成するクラスを実装する。[3]
- オブジェクトを生成するクラスのメソッドを呼び出し「風景画像」を生成するクラス。

設計

要件 1

ディレクトリ構成

- fp18/lec006/fp06_01.rb　(図形クラスのメソッドを呼び出し風景画像へ配置されるオブジェクトを生成するクラス)
- fp18/modules/fp_image.rb　(最も抽象度の高い図形を生成する処理)

クラス図

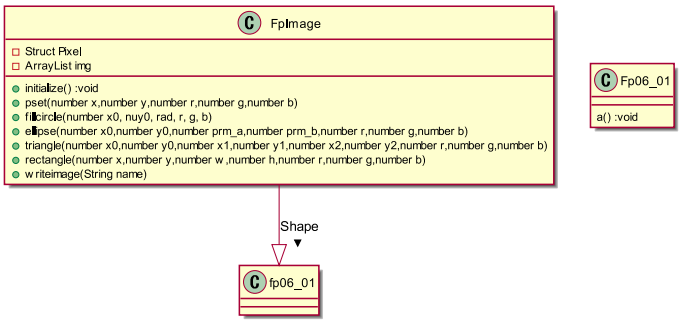

図 1: 要件 1 のクラス図

[2]要件 1 と共用の図形クラス
[3]図形を組み合わせてオブジェクトを生成するクラス

ユースケース

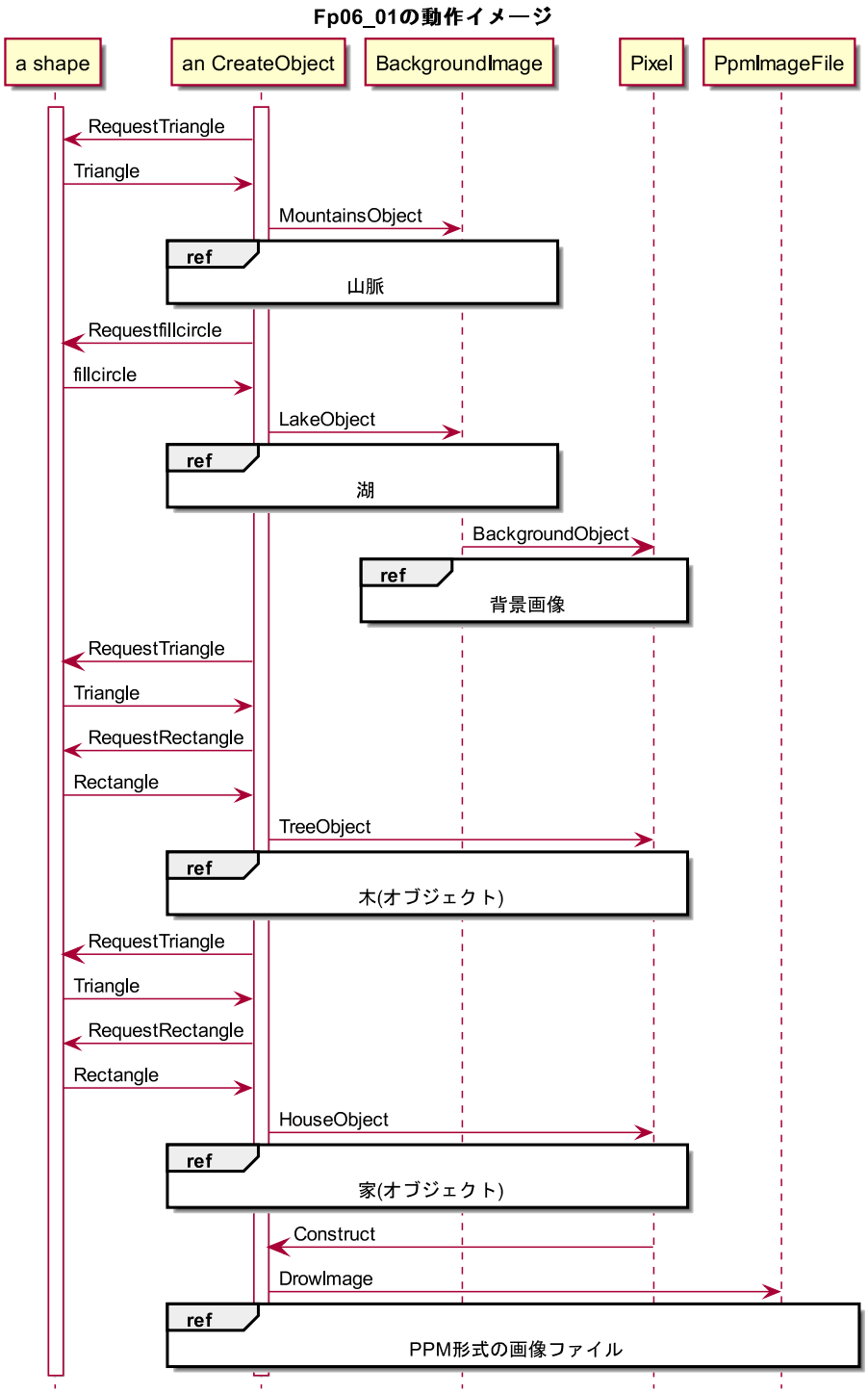

図 2: 要件 1 のユースケース

要件 2

ディレクトリ構成

- fp18/lec006/fp06_01_b.rb （「風景画像」を生成する処理)
- fp18/lec006/app/module/draw_image.rb （図形クラスのメソッドを呼び出し風景画像へ配置されるオブジェクトを生成するクラス)
- fp18/modules/fp_module.rb （最も抽象度の高い図形を生成するクラス)

クラス図

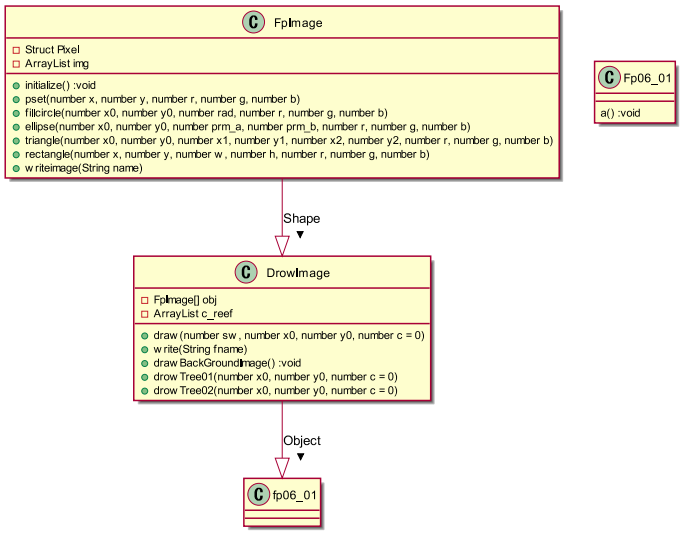

図 3: 要件 2 のクラス図

ユースケース

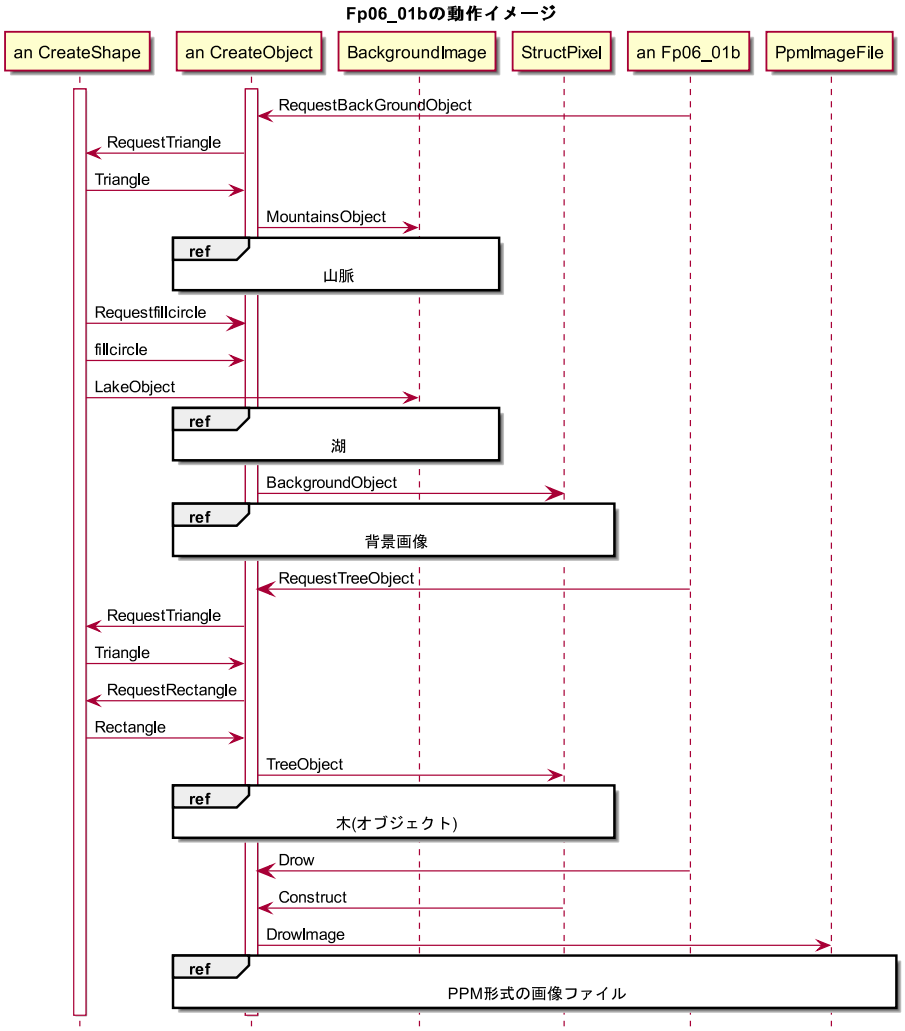

図 4: 要件 2 のユースケース

プログラムコード

共通処理

最も抽象度の高い図形を生成するクラス

- fp18/modules/fp_image.rb

```
class FpImage

  def initialize
    @Pixel = Struct.new(:r, :g, :b)
    @img = Array.new(200) do
    Array.new(300) do @Pixel.new(255,255,255) end
    end
    return true
  end

  def pset(x,y,r,g,b)
    if 0 <= x && x < 300 && 0 <= y && y < 200
    @img[y][x].r = r; @img[y][x].g = g; @img[y][x].b = b
    end
    return true
  end

  def writeimage(name)
    open(name, "wb") do |f|
      f.puts("P6\n300 200\n255")
      @img.each do |a|
        a.each do |p| f.write(p.to_a.pack("ccc")) end
      end
    end
    return true
  end

  def fillcircle(x0, y0, rad, r, g, b)
    begin
      200.times do |y|
        300.times do |x|
          if (x-x0)**2 + (y-y0)**2 <= rad**2 then pset(x, y, r
              , g, b) end
        end
      end
    rescue => e
      return e
    end
  end

  def ellipse(x0, y0, prm_a, prm_b, r, g, b)
    begin
      200.times do |y|
        300.times do |x|
          if (((x-x0).to_f**2)/(prm_a.to_f**2)) + (((y-y0).
             to_f**2)/(prm_b.to_f**2)) <= 1.0 then
            pset(x, y, r, g, b)
```

```
            end
          end
        end
      rescue => e
        return e
      end
    end

    def triangle(x0, y0, x1, y1, x2, y2, r, g, b)
      begin
        200.times do |y|
          300.times do |x|
            if (x1 - x0)*(y - y0) - (y1 - y0)*(x - x0) <= 0
            && (x2 - x0)*(y - y0) - (y2 - y0)*(x - x0) >= 0
            && (x2 - x1)*(y - y1) - (y2 - y1)*(x - x1) <= 0 then
              pset(x, y, r, g, b)
            end
          end
        end
        return true
      rescue => e
        return e
      end
    end

    def rectangle(x, y, w, h, r, g, b)
      begin
        j0 = (y - 0.5 * h).to_i
        j1 = (y + 0.5 * h).to_i
        i0 = (x - 0.5 * w).to_i
        i1 = (x + 0.5 * w).to_i
        j0.step(j1) do |j|
          i0.step(i1) do |i| pset(i, j, r, g, b) end
        end
        return true
      rescue => e
        return e
      end
    end

  end
```

要件1

- fp18/lec006/fp06_01.rb

```
require __dir__ + '/../modules/fp_image.rb'
#/=====================================================/#
# 処理の中でオブジェクトを生成する実装例
# ※実装は煩雑だが処理固有のオブジェクトを定義できる
#/=====================================================/#
def main
  begin
    obj = FpImage.new

    obj.rectangle(150,100,300,200,67,135,233)

    obj.rectangle(150,160,300,80,171,255,127)

    #/=====================================================/#
    # 背景の山
    #/=====================================================/#
    # triangle(x0, y0, x1, y1, x2, y2, r, g, b)
    obj.triangle(150,30,-80,120,200,120,255,255,255)
    # triangle(x0, y0, x1, y1, x2, y2, r, g, b)
    obj.triangle(200,30,0,120,300,120,86,99,143)
    # triangle(x0, y0, x1, y1, x2, y2, r, g, b)
    obj.triangle(40,20,-40,120,80,120,231,232,226)
    # triangle(x0, y0, x1, y1, x2, y2, r, g, b)
    obj.triangle(260,20,180,120,360,120,231,232,226)
    # ellipse(x0, y0, prm_a, prm_b, r, g, b)
    #/=====================================================/#
    #  湖
    #/=====================================================/#
    obj.ellipse(100,160,60,20,0,0,255)
    obj.ellipse(0,180,120,40,0,0,255)
    #/=====================================================/#
    #  木
    #/=====================================================/#
    # Tree Obj01
    # rectangle(x, y, w, h, r, g, b)
    obj.rectangle(40,110,4,40,93,93,33)
    # triangle(x0, y0, x1, y1, x2, y2, r, g, b)
    obj.triangle(40,30,30,70,50,70,47,79,54)
    # triangle(x0, y0, x1, y1, x2, y2, r, g, b)
    obj.triangle(40,50,35,90,55,90,47,79,54)
    # triangle(x0, y0, x1, y1, x2, y2, r, g, b)
    obj.triangle(40,80,40,110,60,110,47,79,54)
    # Tree Obj01
    # rectangle(x, y, w, h, r, g, b)
    obj.rectangle(10,110,4,40,93,93,33)
    # triangle(x0, y0, x1, y1, x2, y2, r, g, b)
    obj.triangle(10,30,0,70,20,70,47,79,54)
    # triangle(x0, y0, x1, y1, x2, y2, r, g, b)
    obj.triangle(10,50,-5,90,15,90,47,79,54)
    # triangle(x0, y0, x1, y1, x2, y2, r, g, b)
    obj.triangle(10,80,-10,110,20,110,47,79,54)
    # Tree Obj01
```

```
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(30,110,4,40,93,93,33)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(30,30,20,70,40,70,47,79,54)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(30,60,15,90,45,90,47,79,54)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(30,80,10,110,50,110,47,79,54)
# Tree Obj01
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(50,115,4,40,93,93,33)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(50,35,40,75,60,75,68,160,119)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(50,55,35,95,65,95,68,160,119)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(50,75,30,115,70,115,68,160,119)
# Tree Obj01
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(20,120,4,40,93,93,33)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(20,40,10,80,30,80,81,178,76)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(20,60,5,100,35,100,81,178,76)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(20,80,0,120,40,120,81,178,76)
# Tree Obj01
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(250,115,4,35,93,93,33)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(250,35,240,75,260,75,44,82,60)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(250,55,235,95,265,95,44,82,60)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(250,75,230,115,270,115,44,82,60)
# Tree Obj01
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(240,120,4,40,93,93,33)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(240,40,230,80,250,80,47,79,54)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(240,60,225,100,255,100,47,79,54)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(240,80,220,120,260,120,47,79,54)
# Tree Obj01
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(290,140,4,40,93,93,33)
# ellipse(x0, y0, prm_a, prm_b, r, g, b)
obj.ellipse(290,80,20,50,51,96,69)
# Tree Obj01
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(270,150,4,40,124,96,53)
# ellipse(x0, y0, prm_a, prm_b, r, g, b)
obj.ellipse(270,90,20,40,81,178,76)
# Tree Obj01
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(280,160,4,40,124,96,53)
```

```
      # ellipse(x0, y0, prm_a, prm_b, r, g, b)
      obj.ellipse(280,100,20,40,11,218,81)
      #/=================================================/#
      # 家の描画
      #/=================================================/#
      # rectangle(x, y, w, h, r, g, b)
      obj.rectangle(220,120,20,80,127,37,9)
      # rectangle(x, y, w, h, r, g, b)
      obj.rectangle(200,150,82,52,0,0,0)
      # rectangle(x, y, w, h, r, g, b)
      obj.rectangle(200,150,80,50,255,249,177)
      # triangle(x0, y0, x1, y1, x2, y2, r, g, b)
      obj.triangle(200,90,150,130,250,130,255,0,0)
      # triangle(x0, y0, x1, y1, x2, y2, r, g, b)
      obj.triangle(200,100,160,130,240,130,255,249,177)
      # rectangle(x, y, w, h, r, g, b)
      obj.rectangle(185,145,40,30,132,90,18)
      # rectangle(x, y, w, h, r, g, b)
      obj.rectangle(176,145,16,26,211,237,251)
      # rectangle(x, y, w, h, r, g, b)
      obj.rectangle(194,145,16,26,211,237,251)
      # rectangle(x, y, w, h, r, g, b)
      obj.rectangle(225,150,20,40,132,90,18)
      #/=================================================/#
      # ppm 生成
      #/=================================================/#
      obj.writeimage("img/6a.ppm")
      return true
    rescue => e
      return e
    end
  end
```

- 要件 1 の実行例

```
[o1820004@sol lec006]$ cd /home2/y2018/o1820004/fp18/lec006
[o1820004@sol lec006]$ pwd
/home2/y2018/o1820004/fp18/lec006
[o1820004@sol lec006]$ irb
irb(main):001:0>
irb(main):001:0> load 'fp06_01.rb'
=> true
irb(main):002:0> main
=> true
irb(main):006:0> quit
```

要件 2

風景画像へ配置されるオブジェクトを生成するクラス

- fp18/lec006/app/module/draw_image.rb

```
require __dir__ + '/../../../modules/fp_image.rb'
#/=====================================================/#
# 図形クラスを呼び出し風景オブジェクトを生成するクラス
#/=====================================================/#
class DrowImage
  def initialize
    @obj = FpImage.new
    # 葉の色相を配列として初期化
    @c_reef = {
    0 => [47,79,54],
    1 => [51,96,69],
    2 => [224,234,58],
    3 => [209,232,41],
    4 => [127,246,85],
    5 => [0,255,65],
    6 => [11,218,81],
    7 => [107,191,63],
    8 => [88,191,63],
    9 => [34,195,80]
    }
  end

  def draw(sw, x0, y0, c = 0)
    begin
      raise ArgumentError if sw.nil? || x0.nil? || y0.nil?
      if sw == 'tree1' then
      puts('tree1')
      self.drowTree01(x0, y0, c)
      elsif sw == 'tree2' then
        puts('tree2')
        self.drowTree02(x0, y0, c)
      else
        return false
      end
      return true
    rescue => e
      return e
    end
  end

  #/=====================================================/#
  # ディレクトリへ画像ファイルを出力するメソッド
  # param String file name
  # return bool
  #/=====================================================/#
  def write(fname = 'test')
    begin
      @obj.writeimage("img/" + fname + ".ppm")
      @onj = nil
      return true
```

```
    rescue => e
      return e
    end
  end

  #/=================================================/#
  # 背景画像生成メソッド
  # param void
  # return bool
  #/=================================================/#
  def drawBackGroundImage
    begin
      #/=============================================/
      #  背景
      #/=============================================/
      @obj.rectangle(150,100,300,200,67,135,233)
      @obj.rectangle(150,160,300,80,171,255,127)
      #/=============================================/
      #  背景の山
      #/=============================================/
      # triangle(x0, y0, x1, y1, x2, y2, r, g, b)
      @obj.triangle(150,30,-80,120,200,120,255,255,255)
      # triangle(x0, y0, x1, y1, x2, y2, r, g, b)
      @obj.triangle(200,30,0,120,300,120,86,99,143)
      # triangle(x0, y0, x1, y1, x2, y2, r, g, b)
      @obj.triangle(40,20,-40,120,80,120,231,232,226)
      # triangle(x0, y0, x1, y1, x2, y2, r, g, b)
      @obj.triangle(260,20,180,120,360,120,231,232,226)
      # ellipse(x0, y0, prm_a, prm_b, r, g, b)
      #/=============================================/
      # 湖
      #/=============================================/
      @obj.ellipse(100,160,60,20,0,0,255)
      @obj.ellipse(0,180,120,40,0,0,255)
      return true
    rescue => e
      return e
    end
  end

  #/=================================================/#
  # 三角形と長方形を組み合わせて「木」を生成するメソッド
  # param int x座標
  # param int y座標
  # param int 葉の色
  # return bool
  #/=================================================/#
  def drowTree01(x0, y0, c = 0)
    begin
      raise ArgumentError if x0.nil? || y0.nil?
      # rectangle(x, y, w, h, r, g, b)
      @obj.rectangle(x0, y0 + 80, 4, 40, 93, 93, 33)
      # triangle(x0, y0, x1, y1, x2, y2, r, g, b)
      @obj.triangle(x0, y0, x0 - 10, y0 + 40, x0 + 10, y0 + 40,
         @c_reef[c][0],@c_reef[c][1],@c_reef[c][2])
      # triangle(x0, y0, x1, y1, x2, y2, r, g, b)
      @obj.triangle(x0, y0 + 20, x0 - 15,y0 + 60, x0 + 15,y0 +
```

```
            60, @c_reef[c][0],@c_reef[c][1],@c_reef[c][2])
        # triangle(x0, y0, x1, y1, x2, y2, r, g, b)
        @obj.triangle(x0, y0 + 50 ,x0 - 20 , y0 + 80, x0 + 20, y0
            + 80, @c_reef[c][0],@c_reef[c][1],@c_reef[c][2])
        return true
      rescue => e
        return e
      end
    end

    #/=====================================================/#
    # 楕円と長方形を組み合わせて「木」を生成するメソッド
    # param int x座標
    # param int y座標
    # param int 葉の色
    # return bool
    #/=====================================================/#
    def drowTree02(x0, y0, c = 0)
      begin
        raise ArgumentError if x0.nil? || y0.nil?
        # Tree Obj01
        # rectangle(x, y, w, h, r, g, b)
        @obj.rectangle(x0, y0 + 60, 4, 40, 93, 93, 33)
        # ellipse(x0, y0, prm_a, prm_b, r, g, b)
        @obj.ellipse(x0, y0, 20, 50, @c_reef[c][0], @c_reef[c][1],
            @c_reef[c][2])
      rescue => e
      return e
        end
    end
  end
```

風景画像」を生成する処理

- fp18/lec006/fp06_01_b.rb

```
require __dir__ + '/app/module/draw_image.rb'
#/=====================================================/#
# オブジェクトの生成をクラスへ切り分けた実装例
# ※処理は簡素だが定型的なオブジェクトしか生成できない。
#/=====================================================/#
def main
  obj = DrowImage.new
  obj.drawBackGroundImage
  obj.draw('tree1',10,30,0)
  obj.draw('tree1',30,30,1)
  obj.draw('tree1',50,30,1)
  obj.draw('tree1',70,30,1)
  obj.draw('tree1',90,30,0)
  obj.draw('tree1',110,30,1)
  obj.draw('tree1',130,30,0)
  obj.draw('tree1',150,30,1)
  obj.draw('tree1',170,30,0)
  obj.draw('tree1',190,30,1)
  obj.draw('tree1',170,30,0)
  obj.draw('tree1',190,30,1)
  obj.draw('tree1',210,30,0)
  obj.draw('tree1',230,30,1)
  obj.draw('tree1',250,30,0)
  obj.draw('tree1',270,30,1)
  obj.draw('tree1',290,30,0)
  obj.draw('tree1',310,30,1)
  obj.draw('tree1',20,40,9)
  obj.draw('tree1',40,40,8)
  obj.draw('tree1',60,40,7)
  obj.draw('tree1',80,40,1)
  obj.draw('tree1',100,40,9)
  obj.draw('tree1',120,40,8)
  obj.draw('tree1',140,40,7)
  obj.draw('tree1',160,40,1)
  obj.draw('tree1',180,40,9)
  obj.draw('tree1',200,40,8)
  obj.draw('tree2',240,70,1)
  obj.draw('tree2',250,80,2)
  obj.draw('tree2',260,70,3)
  obj.draw('tree2',270,80,4)
  obj.draw('tree2',280,70,5)
  obj.write
  return true
end
```

- 要件 2 の実行例

```
[o1820004@sol lec006]$ cd /home2/y2018/o1820004/fp18/lec006
[o1820004@sol lec006]$ pwd
/home2/y2018/o1820004/fp18/lec006
[o1820004@sol lec006]$ irb
irb(main):001:0>
irb(main):004:0> load 'fp06_01_b.rb'
=> true
irb(main):002:0> main
=> true
irb(main):006:0> quit
```

プログラムの説明

共通処理

fp18/modules/fp_image.rb

「10/29〜:#05 2 次元配列; レコード; 画像」で作成した処理をクラス化。

- インスタンスフィールドの初期化を行うメソッド「initialize」

```
def initialize
  @Pixel = Struct.new(:r, :g, :b)
  @img = Array.new(200) do
  Array.new(300) do @Pixel.new(255,255,255) end
  end
  return true
end
```

- 引数として渡された座標 (x,y) へ RGB カラーコードをセットするメソッド「pset」

```
def pset(x,y,r,g,b)
  if 0 <= x && x < 300 && 0 <= y && y < 200
    @img[y][x].r = r; @img[y][x].g = g; @img[y][x].b = b
  end
  return true
end
```

- メ引数として渡された「ファイルパス/ファイル名」へ PPM ファイルを書き出すソッド「writeimage」

```
def writeimage(name)
  open(name, "wb") do |f|
    f.puts("P6\n300 200\n255")
    @img.each do |a|
    a.each do |p| f.write(p.to_a.pack("ccc")) end
    end
  end
  return true
end
```

- メソッド「fillcircle」※ #6 課題では使用しない処理

```
def fillcircle(x0, y0, rad, r, g, b)
  begin
    200.times do |y|
      300.times do |x|
        if (x-x0)**2 + (y-y0)**2 <= rad**2 then pset(x, y, r, g,
            b) end
      end
    end
  rescue => e
    return e
  end
end
```

- 楕円形を描画するメソッド「ellipse」

```
def ellipse(x0, y0, prm_a, prm_b, r, g, b)
  begin
    200.times do |y|
      300.times do |x|
        if (((x-x0).to_f**2)/(prm_a.to_f**2)) + (((y-y0).to_f
           **2)/(prm_b.to_f**2)) <= 1.0 then
          pset(x, y, r, g, b)
        end
      end
    end
  rescue => e
    return e
  end
end
```

- 三角形を描画するメソッド「triangle」

```
def triangle(x0, y0, x1, y1, x2, y2, r, g, b)
  begin
    200.times do |y|
      300.times do |x|
        if (x1 - x0)*(y - y0) - (y1 - y0)*(x - x0) <= 0
        && (x2 - x0)*(y - y0) - (y2 - y0)*(x - x0) >= 0
```

```
          && (x2 - x1)*(y - y1) - (y2 - y1)*(x - x1) <= 0 then
            pset(x, y, r, g, b)
          end
        end
      end
      return true
    rescue => e
      return e
    end
  end
```

- 長方形を描画するメソッド「rectangle」

```
  def rectangle(x, y, w, h, r, g, b)
    begin
      j0 = (y - 0.5 * h).to_i
      j1 = (y + 0.5 * h).to_i
      i0 = (x - 0.5 * w).to_i
      i1 = (x + 0.5 * w).to_i
      j0.step(j1) do |j|
        i0.step(i1) do |i| pset(i, j, r, g, b) end
      end
      return true
    rescue => e
      return e
    end
  end
```

要件 1

fp18/lec006/fp06_01.rb

以下、main メソッドの各処理の詳細。

- 背景の山を描画する処理。座標並びに RGB コードを直接指定しクラス「FpImage」の「triangle」メソッドを直接呼出す。

```
#/=====================================================/#
# 背 景 の 山
#/=====================================================/#
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(150,30,-80,120,200,120,255,255,255)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(200,30,0,120,300,120,86,99,143)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(40,20,-40,120,80,120,231,232,226)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(260,20,180,120,360,120,231,232,226)
```

- 湖を描画する処理。座標並びに RGB コードを直接指定しクラス「FpImage」の「ellipse」メソッドを直接呼出す。

```
#/=====================================================/#
#  湖
#/=====================================================/#
obj.ellipse(100,160,60,20,0,0,255)
obj.ellipse(0,180,120,40,0,0,255)
```

- 木を描画する処理。座標並びに RGB コードを直接指定しクラス「FpImage」の「rectangle」及び「rectangle」メソッドを直接呼出す。

```
#/=====================================================/#
#  木
#/=====================================================/#
# Tree Obj01
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(40,110,4,40,93,93,33)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(40,30,30,70,50,70,47,79,54)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(40,50,35,90,55,90,47,79,54)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(40,80,40,110,60,110,47,79,54)
```

- 家を描画する処理。座標並びに RGB コードを直接指定しクラス「FpImage」の「rectangle」及び「triangle」メソッドを直接呼出す。

```
#/=================================================/#
# 家の描画
#/=================================================/#
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(220,120,20,80,127,37,9)
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(200,150,82,52,0,0,0)
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(200,150,80,50,255,249,177)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(200,90,150,130,250,130,255,0,0)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(200,100,160,130,240,130,255,249,177)
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(185,145,40,30,132,90,18)
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(176,145,16,26,211,237,251)
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(194,145,16,26,211,237,251)
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(225,150,20,40,132,90,18)
```

要件 1 の説明

図形を生成するクラス「FpImage」の各図形生成処理メソッドを直接呼び出し任意のオブジェクトを描画できる実装とした。

木、家等のオブジェクトを直接描画する為、実装は煩雑となるがオブジェクトのサイズ、色、配置等自由度は高い。

fp18/lec006/app/module/draw_image.rb

- 「FpImage」クラスのインスタンスを生成しフィールドの初期化を行うメソッド「initialize」

```
def initialize
  @obj = FpImage.new
  # 葉の色相を配列として初期化
  @c_reef = {
  0 => [47,79,54],
  1 => [51,96,69],
  2 => [224,234,58],
  3 => [209,232,41],
  4 => [127,246,85],
  5 => [0,255,65],
  6 => [11,218,81],
  7 => [107,191,63],
  8 => [88,191,63],
  9 => [34,195,80]
  }
end
```

- 第一引数に指定されたメソッドを呼び出すメソッド「draw」

```
def draw(sw, x0, y0, c = 0)
  begin
    raise ArgumentError if sw.nil? || x0.nil? || y0.nil?
    if sw == 'tree1' then
      puts('tree1')
      self.drowTree01(x0, y0, c)
    elsif sw == 'tree2' then
      puts('tree2')
      self.drowTree02(x0, y0, c)
    else
      return false
    end
    return true
  rescue => e
    return e
  end
end
```

- 第一引数に指定されたパスへPPMファイルを出力するメソッド

```
#/=====================================================/#
# ディレクトリへ画像ファイルを出力するメソッド
# param String file name
# return bool
#/=====================================================/#
def write(fname = 'test')
  begin
    @obj.writeimage("img/" + fname + ".ppm")
```

```
      @onj = nil
      return true
    rescue => e
      return e
    end
  end
```

- 背景画像を描画するメソッド

```
    #/================================================/#
  # 背景画像生成メソッド
  # param void
  # return bool
  #/================================================/#
  def drawBackGroundImage
    begin
      #/============================================/
      #  背景
      #/============================================/
      @obj.rectangle(150,100,300,200,67,135,233)
      @obj.rectangle(150,160,300,80,171,255,127)
      #/============================================/
      #  背景の山
      #/============================================/
      # triangle(x0, y0, x1, y1, x2, y2, r, g, b)
      @obj.triangle(150,30,-80,120,200,120,255,255,255)
      # triangle(x0, y0, x1, y1, x2, y2, r, g, b)
      @obj.triangle(200,30,0,120,300,120,86,99,143)
      # triangle(x0, y0, x1, y1, x2, y2, r, g, b)
      @obj.triangle(40,20,-40,120,80,120,231,232,226)
      # triangle(x0, y0, x1, y1, x2, y2, r, g, b)
      @obj.triangle(260,20,180,120,360,120,231,232,226)
      # ellipse(x0, y0, prm_a, prm_b, r, g, b)
      #/============================================/
      # 湖
      #/============================================/
      @obj.ellipse(100,160,60,20,0,0,255)
      @obj.ellipse(0,180,120,40,0,0,255)
      return true
    rescue => e
      return e
    end
  end
```

- クラス「FpImage」の「rectangle」、「triangle」メソッドを呼び出し「木」オブジェクトを生成するメソッド

```
  #/================================================/#
  # 三角形と長方形を組み合わせて「木」を生成するメソッド
  # param int x座標
  # param int y座標
  # param int 葉の色
```

```
  # return bool
  #/=================================================/#
  def drowTree01(x0, y0, c = 0)
    begin
      raise ArgumentError if x0.nil? || y0.nil?
      # rectangle(x, y, w, h, r, g, b)
      @obj.rectangle(x0, y0 + 80, 4, 40, 93, 93, 33)
      # triangle(x0, y0, x1, y1, x2, y2, r, g, b)
      @obj.triangle(x0, y0, x0 - 10, y0 + 40, x0 + 10, y0 + 40,
         @c_reef[c][0],@c_reef[c][1],@c_reef[c][2])
      # triangle(x0, y0, x1, y1, x2, y2, r, g, b)
      @obj.triangle(x0, y0 + 20, x0 - 15,y0 + 60, x0 + 15,y0 + 60,
          @c_reef[c][0],@c_reef[c][1],@c_reef[c][2])
      # triangle(x0, y0, x1, y1, x2, y2, r, g, b)
      @obj.triangle(x0, y0 + 50 ,x0 - 20 , y0 + 80, x0 + 20, y0 +
         80, @c_reef[c][0],@c_reef[c][1],@c_reef[c][2])
      return true
    rescue => e
      return e
    end
  end
```

- クラス「FpImage」の「rectangle」、「ellipse」メソッドを呼び出し「木」オブジェクトを生成するメソッド

```
  #/=================================================/#
  # 楕円と長方形を組み合わせて「木」を生成するメソッド
  # param int x座標
  # param int y座標
  # param int 葉の色
  # return bool
  #/=================================================/#
  def drowTree02(x0, y0, c = 0)
    begin
      raise ArgumentError if x0.nil? || y0.nil?
      # Tree Obj01
      # rectangle(x, y, w, h, r, g, b)
      @obj.rectangle(x0, y0 + 60, 4, 40, 93, 93, 33)
      # ellipse(x0, y0, prm_a, prm_b, r, g, b)
      @obj.ellipse(x0, y0, 20, 50, @c_reef[c][0], @c_reef[c][1],
         @c_reef[c][2])
    rescue => e
      return e
    end
  end
```

要件 2

fp18/lec006/fp06_01 b.rb

以下、main メソッドの各処理の詳細。

- クラス「DrowImage」のメソッド「drawBackGroundImage」を呼び出し「背景画像」を描画する処理

```
obj.drawBackGroundImage
```

- クラス「DrowImage」のメソッド「tree1」を呼び出し「木」を描画する処理

```
obj.draw('tree1',180,40,9)
```

- クラス「DrowImage」のメソッド「tree2」を呼び出し「木」を描画する処理

```
obj.draw('tree2',240,70,1)
```

要件 2 の説明

図形を生成するクラス「FpImage」をオブジェクトを生成するクラス「DrowImage」で呼び出し、呼び出し元の処理からはオブジェクト物理名、座標 (x,y)、色コードを指定するのみで、任意の箇所に任意のオブジェクトを配置できる実装とした。

上記により、定型的なオブジェクト生成にかかる実装コストを低減し、再利用による簡易な実装を可能とした。

生成された絵

要件 1 のソースで生成した画像

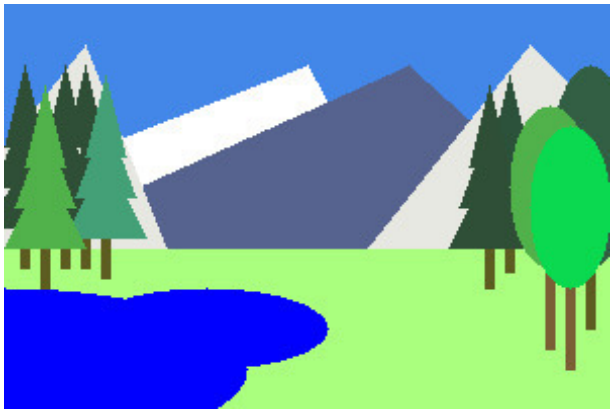

図 5: 要件 1:風景のみ

定型的な「木」オブジェクトをすべてフルスクラッチで手書きした作例。

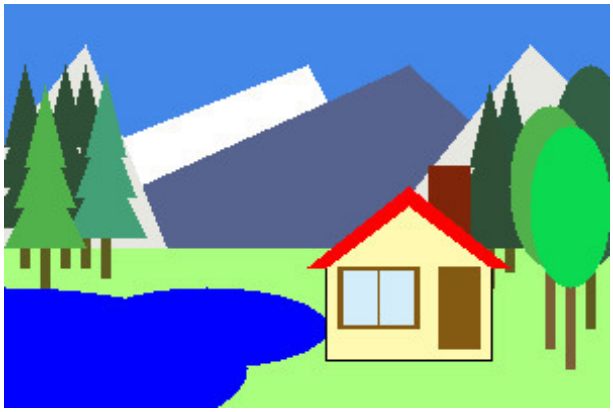

図 6: 要件 1:家オブジェクトを配置

上記に加えて「家」オブジェクトを配置した作例。
任意の定型的ではないオブジェクトを配置できる為、自由度は高いものの実装コストも高まる。

要件 2 のソースで生成した画像

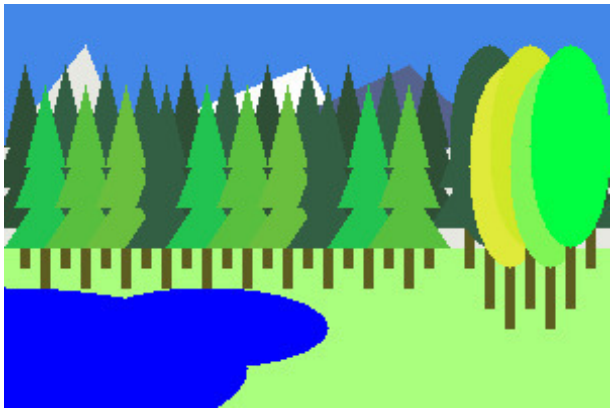

図 7: 要件 2:風景のみ

定型的な「木」オブジェクトの生成をメソッド化した作例。定型化してしまったオブジェクトの生成コストは低いが自由度も低い。

考察

本課題は基本的な図形の生成処理についてはナレッジがテキスト上に明示されており、テキストの方針に従う限りにおいて、図形生成については既知の内容となる。上記を踏まえ、テキストのナレッジを再利用する形でオブジェクト指向的な設計方針の試行の機会と捉え、前述の方針で成果物を実装した。

ある程度複雑な画像は図形「オブジェクト」の組み合わせで表現できる為、効率性の観点から要件 2 の実装が最終的な成果物となるが、画像生成に関しては非定型的な実装についても考慮すべき場合もあり得ると考え、プロトタイプ作成の段階で作成した成果物を「要件 1」として追記した。

また当初「目標」に明示した「拡張性」についても要件 1、要件 2 のケースで明記した通り、画像生成、オブジェクト生成処理をクラスとしての実装とした事で再利用と拡張を実現できたと考える。

ソースコード一覧

- /home2/y2018/o1820004/fp18/modules/fp_image.rb
- /home2/y2018/o1820004/fp18/lec006/fp06_01.rb
- /home2/y2018/o1820004/fp18/lec006/fp06_01_b.rb
- /home2/y2018/o1820004/fp18/lec006/app/module/draw_image.rb

アンケートの解答

Q1. 画像が自由に生成できるようになりましたか。ある程度。Q2. 画像をうまく生成する「コツ」は何だと思いましたか。定型的な図形の抽象化、オブイジェクトの汎化。Q3. リフレクション (今回の課題で分かったこと)・感想・要望をどうぞ。特になし。

提出用ソースコード

/home2/y2018/o1820004/fp18/modules/fp_image.rb

```
class FpImage

def initialize
@Pixel = Struct.new(:r, :g, :b)
@img = Array.new(200) do
Array.new(300) do @Pixel.new(255,255,255) end
end
return true
end

def pset(x,y,r,g,b)
if 0 <= x && x < 300 && 0 <= y && y < 200
@img[y][x].r = r; @img[y][x].g = g; @img[y][x].b = b
end
return true
end

def writeimage(name)
open(name, "wb") do |f|
f.puts("P6\n300 200\n255")
@img.each do |a|
a.each do |p| f.write(p.to_a.pack("ccc")) end
end
end
return true
end

def fillcircle(x0, y0, rad, r, g, b)
```

```
begin
200.times do |y|
300.times do |x|
if (x-x0)**2 + (y-y0)**2 <= rad**2 then pset(x, y, r, g, b) end
end
end
rescue => e
return e
end
end

def ellipse(x0, y0, prm_a, prm_b, r, g, b)
begin
200.times do |y|
300.times do |x|
if (((x-x0).to_f**2)/(prm_a.to_f**2)) + (((y-y0).to_f**2)/(prm_b.to_f**2)) <= 1.0 t
pset(x, y, r, g, b)
end
end
end
rescue => e
return e
end
end

def triangle(x0, y0, x1, y1, x2, y2, r, g, b)
begin
200.times do |y|
300.times do |x|
if (x1 - x0)*(y - y0) - (y1 - y0)*(x - x0) <= 0
&& (x2 - x0)*(y - y0) - (y2 - y0)*(x - x0) >= 0
&& (x2 - x1)*(y - y1) - (y2 - y1)*(x - x1) <= 0 then
pset(x, y, r, g, b)
end
end
end
return true
rescue => e
return e
```

```
end
end

def rectangle(x, y, w, h, r, g, b)
begin
j0 = (y - 0.5 * h).to_i
j1 = (y + 0.5 * h).to_i
i0 = (x - 0.5 * w).to_i
i1 = (x + 0.5 * w).to_i
j0.step(j1) do |j|
i0.step(i1) do |i| pset(i, j, r, g, b) end
end
return true
rescue => e
return e
end
end

end
```

/home2/y2018/o1820004/fp18/lec006/ fp06_01.rb

```
require __dir__ + '/../modules/fp_image.rb'
#/=================================================/#
# 処理の中でオブジェクトを生成する実装例
# ※実装は煩雑だが処理固有のオブジェクトを定義できる
#/=================================================/#
def main
begin
obj = FpImage.new

obj.rectangle(150,100,300,200,67,135,233)

obj.rectangle(150,160,300,80,171,255,127)

#/=================================================/#
# 背景の山
#/=================================================/#
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
```

```
obj.triangle(150,30,-80,120,200,120,255,255,255)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(200,30,0,120,300,120,86,99,143)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(40,20,-40,120,80,120,231,232,226)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(260,20,180,120,360,120,231,232,226)
# ellipse(x0, y0, prm_a, prm_b, r, g, b)
#/================================================/#
#  湖
#/================================================/#
obj.ellipse(100,160,60,20,0,0,255)
obj.ellipse(0,180,120,40,0,0,255)
#/================================================/#
#  木
#/================================================/#
# Tree Obj01
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(40,110,4,40,93,93,33)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(40,30,30,70,50,70,47,79,54)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(40,50,35,90,55,90,47,79,54)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(40,80,40,110,60,110,47,79,54)
# Tree Obj01
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(10,110,4,40,93,93,33)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(10,30,0,70,20,70,47,79,54)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(10,50,-5,90,15,90,47,79,54)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(10,80,-10,110,20,110,47,79,54)
# Tree Obj01
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(30,110,4,40,93,93,33)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(30,30,20,70,40,70,47,79,54)
```

```
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(30,60,15,90,45,90,47,79,54)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(30,80,10,110,50,110,47,79,54)
# Tree Obj01
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(50,115,4,40,93,93,33)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(50,35,40,75,60,75,68,160,119)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(50,55,35,95,65,95,68,160,119)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(50,75,30,115,70,115,68,160,119)
# Tree Obj01
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(20,120,4,40,93,93,33)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(20,40,10,80,30,80,81,178,76)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(20,60,5,100,35,100,81,178,76)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(20,80,0,120,40,120,81,178,76)
# Tree Obj01
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(250,115,4,35,93,93,33)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(250,35,240,75,260,75,44,82,60)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(250,55,235,95,265,95,44,82,60)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(250,75,230,115,270,115,44,82,60)
# Tree Obj01
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(240,120,4,40,93,93,33)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(240,40,230,80,250,80,47,79,54)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(240,60,225,100,255,100,47,79,54)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
```

```
obj.triangle(240,80,220,120,260,120,47,79,54)
# Tree Obj01
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(290,140,4,40,93,93,33)
# ellipse(x0, y0, prm_a, prm_b, r, g, b)
obj.ellipse(290,80,20,50,51,96,69)
# Tree Obj01
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(270,150,4,40,124,96,53)
# ellipse(x0, y0, prm_a, prm_b, r, g, b)
obj.ellipse(270,90,20,40,81,178,76)
# Tree Obj01
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(280,160,4,40,124,96,53)
# ellipse(x0, y0, prm_a, prm_b, r, g, b)
obj.ellipse(280,100,20,40,11,218,81)
#/================================================/#
# 家の描画
#/================================================/#
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(220,120,20,80,127,37,9)
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(200,150,82,52,0,0,0)
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(200,150,80,50,255,249,177)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(200,90,150,130,250,130,255,0,0)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
obj.triangle(200,100,160,130,240,130,255,249,177)
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(185,145,40,30,132,90,18)
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(176,145,16,26,211,237,251)
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(194,145,16,26,211,237,251)
# rectangle(x, y, w, h, r, g, b)
obj.rectangle(225,150,20,40,132,90,18)
#/================================================/#
# ppm 生成
```

```
#/================================================/#
obj.writeimage("img/6a.ppm")
return true
rescue => e
return e
end
end
```

/home2/y2018/o1820004/fp18/lec006/app/module/draw_image.rb

```
require __dir__ + '/../../../modules/fp_image.rb'
#/================================================/#
# 図形クラスを呼び出し風景オブジェクトを生成するクラス
#/================================================/#
class DrowImage
def initialize
@obj = FpImage.new
# 葉の色相を配列として初期化
@c_reef = {
0 => [47,79,54],
1 => [51,96,69],
2 => [224,234,58],
3 => [209,232,41],
4 => [127,246,85],
5 => [0,255,65],
6 => [11,218,81],
7 => [107,191,63],
8 => [88,191,63],
9 => [34,195,80]
}
end

def draw(sw, x0, y0, c = 0)
begin
raise ArgumentError if sw.nil? || x0.nil? || y0.nil?
if sw == 'tree1' then
puts('tree1')
self.drowTree01(x0, y0, c)
elsif sw == 'tree2' then
```

```
puts('tree2')
self.drowTree02(x0, y0, c)
else
return false
end
return true
rescue => e
return e
end
end

#/=================================================/#
# ディレクトリへ画像ファイルを出力するメソッド
# param String file name
# return bool
#/=================================================/#
def write(fname = 'test')
begin
@obj.writeimage("img/" + fname + ".ppm")
@onj = nil
return true
rescue => e
return e
end
end

#/=================================================/#
# 背景画像生成メソッド
# param void
# return bool
#/=================================================/#
def drawBackGroundImage
begin
#/=============================================/
#  背景
#/=============================================/
@obj.rectangle(150,100,300,200,67,135,233)
@obj.rectangle(150,160,300,80,171,255,127)
#/=============================================/
```

```
#  背景の山
#/=============================================/
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
@obj.triangle(150,30,-80,120,200,120,255,255,255)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
@obj.triangle(200,30,0,120,300,120,86,99,143)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
@obj.triangle(40,20,-40,120,80,120,231,232,226)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
@obj.triangle(260,20,180,120,360,120,231,232,226)
# ellipse(x0, y0, prm_a, prm_b, r, g, b)
#/=============================================/
# 湖
#/=============================================/
@obj.ellipse(100,160,60,20,0,0,255)
@obj.ellipse(0,180,120,40,0,0,255)
return true
rescue => e
return e
end
end

#/=================================================/#
# 三角形と長方形を組み合わせて「木」を生成するメソッド
# param int x 座標
# param int y 座標
# param int 葉の色
# return bool
#/=================================================/#
def drowTree01(x0, y0, c = 0)
begin
raise ArgumentError if x0.nil? || y0.nil?
# rectangle(x, y, w, h, r, g, b)
@obj.rectangle(x0, y0 + 80, 4, 40, 93, 93, 33)
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
@obj.triangle(x0, y0, x0 - 10, y0 + 40, x0 + 10, y0 + 40, @c_reef[c][0],@c_reef[c]
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
@obj.triangle(x0, y0 + 20, x0 - 15,y0 + 60, x0 + 15,y0 + 60, @c_reef[c][0],@c_reef
# triangle(x0, y0, x1, y1, x2, y2, r, g, b)
```

```
@obj.triangle(x0, y0 + 50 ,x0 - 20 , y0 + 80, x0 + 20, y0 + 80, @c_reef[c][0],@c_re
return true
rescue => e
return e
end
end

#/=================================================/#
# 楕円と長方形を組み合わせて「木」を生成するメソッド
# param int x座標
# param int y座標
# param int 葉の色
# return bool
#/=================================================/#
def drowTree02(x0, y0, c = 0)
begin
raise ArgumentError if x0.nil? || y0.nil?
# Tree Obj01
# rectangle(x, y, w, h, r, g, b)
@obj.rectangle(x0, y0 + 60, 4, 40, 93, 93, 33)
# ellipse(x0, y0, prm_a, prm_b, r, g, b)
@obj.ellipse(x0, y0, 20, 50, @c_reef[c][0], @c_reef[c][1], @c_reef[c][2])
rescue => e
return e
end
end
end
```

/home2/y2018/o1820004/fp18/lec006/fp06_01_b.rb

```
require __dir__ + '/app/module/draw_image.rb'
#/=================================================/#
# オブジェクトの生成をクラスへ切り分けた実装例
# ※処理は簡素だが定型的なオブジェクトしか生成できない。
#/=================================================/#
def main
obj = DrowImage.new
obj.drawBackGroundImage
obj.draw('tree1',10,30,0)
```

```
obj.draw('tree1',30,30,1)
obj.draw('tree1',50,30,1)
obj.draw('tree1',70,30,1)
obj.draw('tree1',90,30,0)
obj.draw('tree1',110,30,1)
obj.draw('tree1',130,30,0)
obj.draw('tree1',150,30,1)
obj.draw('tree1',170,30,0)
obj.draw('tree1',190,30,1)
obj.draw('tree1',170,30,0)
obj.draw('tree1',190,30,1)
obj.draw('tree1',210,30,0)
obj.draw('tree1',230,30,1)
obj.draw('tree1',250,30,0)
obj.draw('tree1',270,30,1)
obj.draw('tree1',290,30,0)
obj.draw('tree1',310,30,1)
obj.draw('tree1',20,40,9)
obj.draw('tree1',40,40,8)
obj.draw('tree1',60,40,7)
obj.draw('tree1',80,40,1)
obj.draw('tree1',100,40,9)
obj.draw('tree1',120,40,8)
obj.draw('tree1',140,40,7)
obj.draw('tree1',160,40,1)
obj.draw('tree1',180,40,9)
obj.draw('tree1',200,40,8)
obj.draw('tree2',240,70,1)
obj.draw('tree2',250,80,2)
obj.draw('tree2',260,70,3)
obj.draw('tree2',270,80,4)
obj.draw('tree2',280,70,5)
obj.write
return true
end
```