Homework 1

UFO Animation

正文

我们先使用老师提供的UFO模型

```
(define UFO
      (overlay
      (circle 10 "solid" "green")
      (rectangle 40 4 "solid" "green")))
```

```
UFO
```

之后新建一个画布，可以让UFO有空间在画布上移动:

```
(define WIDTH 100)
(define HEIGHT 600)
(define MTSCN (empty-scene WIDTH HEIGHT))
```

在画布中放置降落的比场景底部高10个像素的平坦岩床

```
(define picture-of-bg
     (place-image (rectangle 100 20 "solid" "grey") 50 HEIGHT MTSCN))
```

之后我们便可以定义UFO/火箭的运动方程，为了使UFO/火箭的下落更加真实，我们定义其运动方程:

```
(define (distance t)
  (cond
    [(< t 108)
  (- (* 10.8 t) (* 0.05 (* t t)) )]
    [(>= t 108)
   HEIGHT]))
```

尽管这个加速度的大小不太正常但是模拟降落的运动效果还是很不错的，当$t > 108$的时候我们将距离固定(让其降落到平坦岩床)

接下来我们将UFO/火箭依照Animate函数放置到画布(picture-of-bg)中完成效果

```
(define (picture-of-UFO t)
 (cond
  [(<= (distance t) (- UFO-CENTER-TO-TOP 10))
   (place-image UFO 50 (distance t) picture-of-bg)]
  [(> (distance t) (- UFO-CENTER-TO-TOP 10))
   (place-image UFO 50 (- UFO-CENTER-TO-TOP 10) picture-of-bg)]))
```

当距离到达距离底部10个像素的时候固定，当UFO/火箭距离底部大于10个像素的时候距离随(distanct t)函数变化，其中UFO-CENTER-TO-TOP:

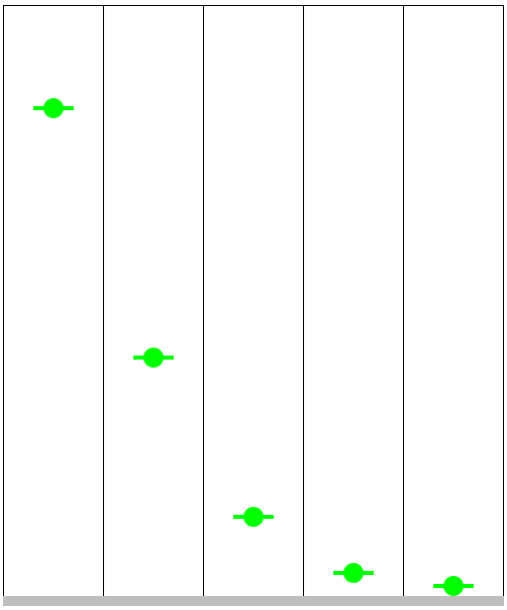
```
(define UFO-CENTER-TO-TOP
  (- HEIGHT (/ (image-height UFO) 2)))
```

执行

```
  (animate picture-of-UFO)
```

即可观察到UFO的降落:

我们依次观察$t = 10, 40, 70, 90, 100$的情况:



同样的对于火箭的降落:

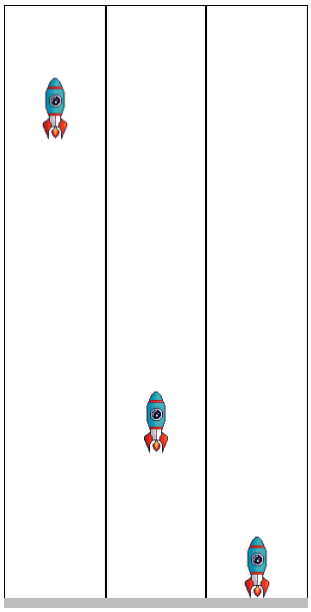
全部代码为:

```
; Parameter
(define WIDTH 100)
(define HEIGHT 600)
(define MTSCN (empty-scene WIDTH HEIGHT))
(define ROCKET ::rocket)
(define ROCKET-CENTER-TO-TOP
  (- HEIGHT (/ (image-height ROCKET) 2)))
; Function
(define (distance t)
```

```
  (cond
    [(< t 108)
  (- (* 10.8 t) (* 0.05 (* t t)) )]
    [(>= t 108)
   HEIGHT]))
(define picture-of-bg
     (place-image (rectangle 100 20 "solid" "grey") 50 HEIGHT MTSCN))
(define (picture-of-rocket.v5 (distance t))
 (cond
  [(<= (distance t) (- ROCKET-CENTER-TO-TOP 10))
   (place-image ROCKET 50 (distance t) picture-of-bg)]
  [(> (distance t) (- ROCKET-CENTER-TO-TOP 10))
   (place-image ROCKET 50 (- ROCKET-CENTER-TO-TOP 10) picture-of-bg)]))
```

执行

```
(animate picture-of-ROCKET)
```



附录

同样的对于任何的图片均可以完成降落的效果如我们自己绘制一个UFO:

```
(require 2htdp/image)
(define body (ellipse 60 30 "solid" "grey" ))
(define ufowin  (overlay/xy (rectangle 40 10 "solid" "grey")
          0 -10
          (ellipse 40 20 "solid" (color 0 225 255))))
(define ufobody (overlay/xy   ufowin
                -10 5
                body))
(define ufo (overlay/xy   (rectangle 60 5 "solid" "blue")
                0 -18
                ufobody))
ufo
```

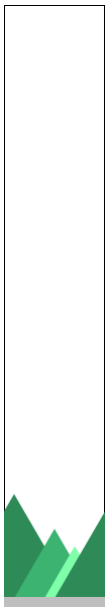
我们可以得到:

并且我们也可以对画布进行更改:

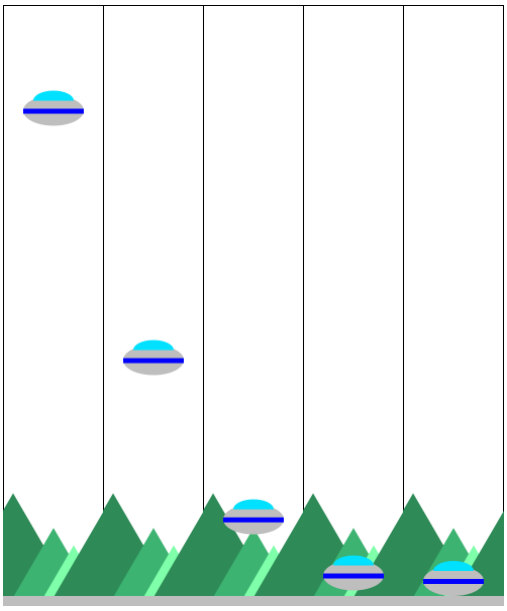
```
(define (mountain n c) (triangle n "solid" c))
(define (picture-of-bg n)
    (define scene1 (place-image (mountain (* n 1.3) (color 46 139 87)) 10 HEIGHT MTSCN))
    (define scene2 (place-image (mountain (* n 0.9) (color 60 179 113)) 50 HEIGHT scene1))
    (define scene3 (place-image (mountain (* n 0.7) (color 127 255 170)) 70 HEIGHT scene2))
    (define scene4 (place-image (mountain (* n 1.5) (color 46 139 87)) 120 HEIGHT scene3))
    (place-image (rectangle 100 20 "solid" "grey") 50 HEIGHT scene4))
```

执行

```
(picture-of-bg 200)
```



置入相关函数完成降落:



Ship

我们将船分为三个部分:

船身

```
(define (shipbody n c)
 (overlay/xy
  (overlay/xy (overlay/xy
  (flip-horizontal (scale/xy 1 1.5
     (Parallelogram n c)))
  (* -7 n) (* 1.5 n)
  (rectangle (* 10 n) (* 3 n) "solid" c))
  (* -8 n) (* -1.5 n)
  (scale/xy 1.5 2
  (Parallelogram n c)))
 (* 6 n) 0
 (rotate -10 (triangle/sss (* 12 n) (* 5 n) (* 15 n) "solid" c))))
```

其中(Parallelogram n c)定义为:

```
(define (Parallelogram n c)
 (overlay/xy (flip-vertical (flip-horizontal (right-triangle (* 4 n) (* 3 n) "solid" c)))
 (* 4 n) 0
 (right-triangle (* 4 n) (* 3 n) "solid" c)))
```

船舱

```
(define (Cabin n c l)
 (overlay/xy (rectangle (* 2 n) (* 1 n) "solid" (color 0 0 255))
(* -6 n) (* -0.5 n)
 (overlay/xy
 (flip-horizontal (right-triangle (* 2 n) (* 3 n) "solid" c))
 (* 2 n) 0
 (rectangle (* (* 8 n) l) (* 3 n) "solid" c))))
```

旗帜

```
(define (flag n) (scale (* 0.5 n) ::flag)))
```

利用Overlay/xy函数将其组合在一起完成船的绘制:

```
(define (ship n)
 (overlay/xy (shipbody n (color 139 136 120))
  (* 9 n) (* -2.5 n)
 (overlay/xy
  (shipabove n (color 238 232 205) 1)
  (* 3 n) (* -2 n)
  (shipabove (* 0.8 n) (color 205 201 165) 0.8))))
(define (ship-final n)
 (overlay/xy
  (flag n)
  (* -15 n) (* 2.1 n)
  (ship n)))
;位置的倍数属实不好调整。
```

完整代码:

```
#lang slideshow
(require 2htdp/image)
(define (Parallelogram n c)
 (overlay/xy (flip-vertical (flip-horizontal (right-triangle (* 4 n) (* 3 n) "solid" c)))
 (* 4 n) 0
 (right-triangle (* 4 n) (* 3 n) "solid" c)))
(define (shipbody n c)
 (overlay/xy
   (overlay/xy (overlay/xy
   (flip-horizontal (scale/xy 1 1.5
      (Parallelogram n c)))
   (* -7 n) (* 1.5 n)
   (rectangle (* 10 n) (* 3 n) "solid" c))
   (* -8 n) (* -1.5 n)
   (scale/xy 1.5 2
   (Parallelogram n c)))
  (* 6 n) 0
  (rotate -10 (triangle/sss (* 12 n) (* 5 n) (* 15 n) "solid" c))))
(define (shipabove n c l)
 (overlay/xy (rectangle (* 2 n) (* 1 n) "solid" (color 0 0 255))
(* -6 n) (* -0.5 n)
 (overlay/xy
 (flip-horizontal (right-triangle (* 2 n) (* 3 n) "solid" c))
 (* 2 n) 0
 (rectangle (* (* 8 n) l) (* 3 n) "solid" c))))
(define (ship n)
 (overlay/xy (shipbody n (color 139 136 120))
  (* 9 n) (* -2.5 n)
 (overlay/xy
  (shipabove n (color 238 232 205) 1)
  (* 3 n) (* -2 n)
  (shipabove (* 0.8 n) (color 205 201 165) 0.8))))
(define (flag n) (scale (* 0.005 n) .))
(define (ship-final n)
 (overlay/xy
  (flag n)
  (* -15 n) (* 2.1 n)
  (ship n)))

(ship-final n)
```

```
;n可以调整船的大小
```

执行

```
(ship-final 50)
```

完成效果