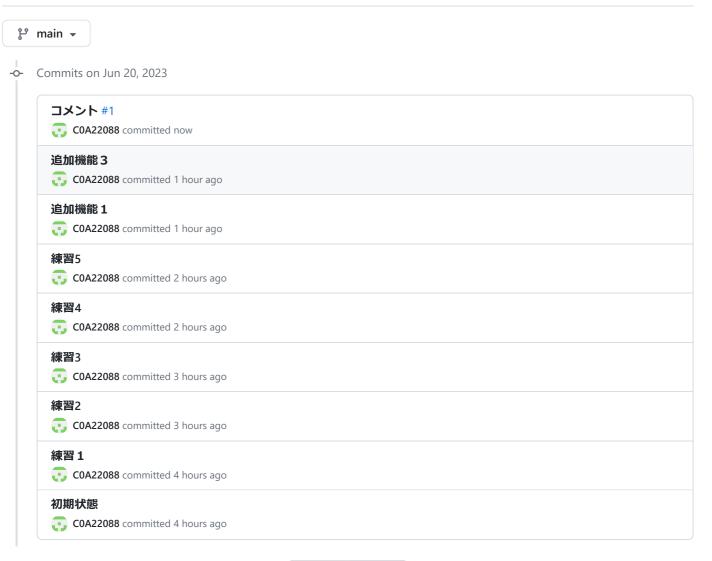
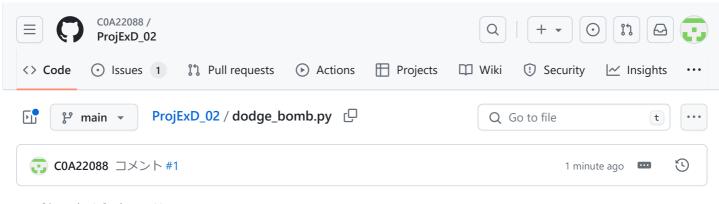


Commits



Newer

Older



```
141 lines (119 loc) · 4.89 KB
                                               면뽀
                                                                 \langle \rangle
  Code
           Blame
     1
           import random
     2
           import sys
     3
           import pygame as pg
     4
           import time
     6
     7
    8
           WIDTH, HEIGHT = 1600, 900
    9 🗸
           delta = {
    10
               pg.K_UP: (0, -5),
    11
               pg.K_DOWN: (0, +5),
    12
               pg.K_LEFT: (-5, 0),
    13
               pg.K_RIGHT: (+5, 0),
   14
           }
    15
   16
           def check_bound(rect: pg.Rect) -> tuple[bool, bool]:
   17 🗸
   18
               こうかとんRect、爆弾Rectが画面外 or 画面内かを判定する関数
    19
               引数:こうかとんRect or 爆弾Rect
    20
               戻り値:横方向、縦方向の判定結果タプル(True:画面内/False: E
    21
    22
    23
               yoko, tate = True, True
               if rect.left < 0 or WIDTH < rect.right: # 横方向判定
    24
    25
    26
               if rect.top < 0 or HEIGHT < rect.bottom: # 縱方向判定
    27
                   tate = False
    28
               return yoko, tate
    29
    30
    31 🗸
           def main():
               pg.display.set_caption("逃げろ!こうかとん")
    32
    33
               screen = pg.display.set_mode((WIDTH, HEIGHT))
    34
               bg_img = pg.image.load("ex02/fig/pg_bg.jpg")
    35
               kk_img = pg.image.load("ex02/fig/3.png")
               kk_img = pg.transform.rotozoom(kk_img, 0, 2.0)
    36
    37
               kk_img_load = pg.image.load("ex02/fig/3.png")
   38
               kk_img_1 = pg.transform.rotozoom(kk_img_load, 0, 2.0) #
    39
               kk_img_2 = pg.transform.flip(kk_img_1, True, False) # ro
               # こうかとんの画像方向の辞書
   40
   41
               kk_imgs = {
                   (+5, 0): kk_img_2, # 右方向こうかとんの画像
   42
    43
                   (+5, -5): pg.transform.rotozoom(kk_img_2, 45, 1.0),
    44
                   (0, -5): pg.transform.rotozoom(kk_img_2, 90, 1.0), #
    45
                   (-5, -5): pg.transform.rotozoom(kk_img_1, -45, 1.0),
```

```
Symbols X

Find definitions and references for functions and other symbols in this file by clicking a symbol below or in the code.

Filter symbols

const delta

func check_bound

func main
```

```
(-5, +5): pg.transform.rotozoom(kk_img_1, 45, 1.0),
 47
 48
                (0, +5): pg.transform.rotozoom(kk_img_2, -90, 1.0),
                (+5, +5): pg.transform.rotozoom(kk_img_2, -45, 1.0),
 49
 50
                }
            kk_img = kk_imgs[+5, 0]
 51
            # こうかとんSurface (kk_img) からこうかとんRect (kk_rct) を抽
 52
            kk_rct = kk_img.get_rect()
 53
            kk_rct.center = 900, 400
 54
 55
 56
            bb_imgs = []
 57
            for r in range(1, 11):
                bb_img = pg.Surface((20*r, 20*r))
 58
 59
                pg.draw.circle(bb_img, (255, 0, 0), (10*r, 10*r), 10*
                bb_img.set_colorkey((0, 0, 0))
 61
                bb_imgs.append(bb_img)
 62
            x, y = random.randint(0, 1600), random.randint(0, 900)
 63
            bb_rct = bb_imgs[0].get_rect()
            bb_rct.center = (x, y)
            vx, vy = +1, +1
 65
            accs = [a for a in range(1, 11)] # 加速度のリスト b
 67
 68
            bd_img = pg.Surface((20, 20)) # 練習 1
            bd_img.set_colorkey((0, 0, 0)) # 黒い部分を透明にする
 69
 70
            pg.draw.circle(bd_img, (255, 0, 0), (10, 10), 10)
            x = random.randint(0, WIDTH)
 71
 72
            y = random.randint(0, HEIGHT)
 73
            # 爆弾Surface (bd_img) から爆弾Rect (bd_rct) を抽出する
 74
            bd_rct = bd_img.get_rect()
            # 爆弾Rectの中心座標を乱数で指定する
 75
 76
            bd_rct.center = x, y
 77
            vx, vy = +5, +5 # 練習2
 78
 79
            clock = pg.time.Clock()
 80
            tmr = 0
            while True:
 81
 82
                for event in pg.event.get():
                    if event.type == pg.QUIT:
 83
 84
                        return
 85
 86
 87
                if kk rct.colliderect(bd rct): # 着弾するとこうかとん画
 88
                    kk_img_lose_load = pg.image.load("ex02/fig/9.png"
                    kk img_lose = pg.transform.rotozoom(kk_img_lose_l
 89
 90
                    kk_img = kk_img_lose
 91
                    screen.blit(kk_img, kk_rct)
 92
                    pg.display.update()
 93
                    time.sleep(2)
 94
                    return
 95
 96
                key_lst = pg.key.get_pressed()
 97
                sum mv = [0, 0] # 合計移動量
 98
                for k, mv in delta.items():
 99
                    if key_lst[k]:
100
                        sum_mv[0] += mv[0]
                        sum_mv[1] += mv[1]
101
102
                kk_rct.move_ip(sum_mv)
103
                if check bound(kk rct) != (True, True):
104
                    kk_rct.move_ip(-sum_mv[0], -sum_mv[1])
105
106
                key_lst = pg.key.get_pressed()
107
                # こうかとんの画像方向を選ぶための変数
108
                kk_0 = 0
```

```
kk_1 = 0
109
110
               for k, mv in delta.items():
                  if key_lst[k]:
111
112
                       kk_0 = kk_0 + mv[0]
113
114
                      kk_1 = kk_1 + mv[1]
115
116
              if kk_0 != 0 or kk_1 != 0: # 飛ぶ方向に従ってこうかとん
117
                   kk_img = kk_imgs[kk_0, kk_1]
118
               screen.blit(kk_img, kk_rct)
119
121
122
               screen.blit(bg_img, [0, 0])
123
               screen.blit(kk_img, kk_rct)
124
               bd_rct.move_ip(vx, vy) # 練習2
               yoko, tate = check_bound(bd_rct)
125
126
               if not yoko: # 横方向に画面外だったら
127
128
                  vx *= -1
               if not tate: # 縦方向に範囲外だったら
129
130
                   vy *= -1
131
               screen.blit(bd_img, bd_rct)
132
               pg.display.update()
               tmr += 1
133
134
               clock.tick(50)
135
136
       if __name__ == "__main__":
137
138
           pg.init()
139
           main()
140
           pg.quit()
           sys.exit()
141
```

