

## ProjExD\_3 / fight\_kokaton.py ☐

8055167 · 37 minutes ago

(I)

쟅 c0c24020ae multibeamブランチ

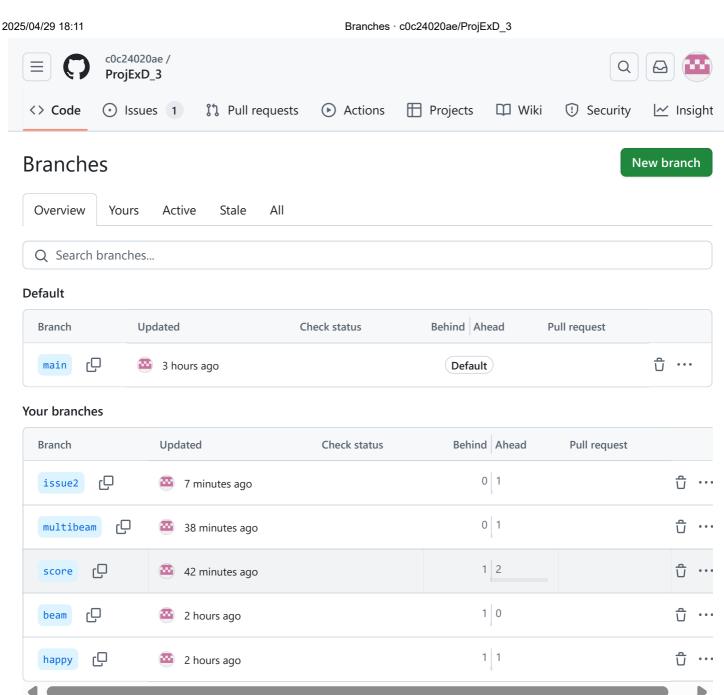
197 lines (174 loc) · 7 KB

```
<>
Code
        Blame
         import os
   1
   2
         import random
         import sys
    3
   4
         import time
   5
         import pygame as pg
   6
   7
         WIDTH = 1100 # ゲームウィンドウの幅
   8
   9
         HEIGHT = 650 # ゲームウィンドウの高さ
         NUM OF BOMBS = 5 # 爆弾の個数
   10
         os.chdir(os.path.dirname(os.path.abspath(__file__)))
   11
   12
   13
   14
         def check_bound(obj_rct: pg.Rect) -> tuple[bool, bool]:
   15
              オブジェクトが画面内or画面外を判定し、真理値タプルを返す関数
   16
              引数:こうかとんや爆弾, ビームなどのRect
   17
              戻り値:横方向,縦方向のはみ出し判定結果(画面内:True/画面外:False)
   18
   19
   20
              yoko, tate = True, True
   21
              if obj_rct.left < 0 or WIDTH < obj_rct.right:</pre>
   22
                 yoko = False
   23
              if obj_rct.top < 0 or HEIGHT < obj_rct.bottom:</pre>
                 tate = False
   24
   25
              return yoko, tate
   26
   27
   28
        class Bird:
   29
              ゲームキャラクター (こうかとん) に関するクラス
   30
   31
              delta = { # 押下キーと移動量の辞書
   32
   33
                 pg.K_UP: (0, -5),
                 pg.K_DOWN: (0, +5),
   34
   35
                 pg.K_LEFT: (-5, 0),
                 pg.K_RIGHT: (+5, 0),
   36
   37
              img0 = pg.transform.rotozoom(pg.image.load("fig/3.png"), 0, 0.9)
   38
              img = pg.transform.flip(img0, True, False) # デフォルトのこうかとん (右向き)
   39
              imgs = { # 0度から反時計回りに定義
   40
   41
                 (+5, 0): img, #右
                  (+5, -5): pg.transform.rotozoom(img, 45, 0.9), # 右上
```

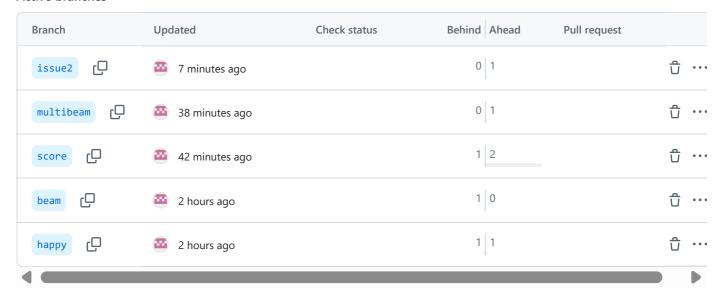
```
43
               (0, -5): pg.transform.rotozoom(img, 90, 0.9), #上
44
               (-5, -5): pg.transform.rotozoom(img0, -45, 0.9), # 左上
45
               (-5, 0): img0, #左
               (-5, +5): pg.transform.rotozoom(img0, 45, 0.9), # 左下
46
47
               (0, +5): pg.transform.rotozoom(img, -90, 0.9), #下
               (+5, +5): pg.transform.rotozoom(img, -45, 0.9), # 右下
48
           }
49
50
51
           def __init__(self, xy: tuple[int, int]):
52
               こうかとん画像Surfaceを生成する
53
               引数 xy:こうかとん画像の初期位置座標タプル
54
55
56
               self.img = __class__.imgs[(+5, 0)]
57
               self.rct: pg.Rect = self.img.get_rect()
58
               self.rct.center = xy
59
60
           def change_img(self, num: int, screen: pg.Surface):
61
62
               こうかとん画像を切り替え, 画面に転送する
               引数1 num:こうかとん画像ファイル名の番号
63
               引数2 screen:画面Surface
64
65
               self.img = pg.transform.rotozoom(pg.image.load(f"fig/{num}.png"), 0, 0.9)
66
67
               screen.blit(self.img, self.rct)
68
69
           def update(self, key_lst: list[bool], screen: pg.Surface):
70
               押下キーに応じてこうかとんを移動させる
71
72
               引数1 key lst:押下キーの真理値リスト
               引数2 screen:画面Surface
73
74
75
               sum_mv = [0, 0]
               for k, mv in __class__.delta.items():
76
                   if key_lst[k]:
77
                      sum_mv[0] += mv[0]
78
79
                      sum mv[1] += mv[1]
20
               self.rct.move_ip(sum_mv)
               if check_bound(self.rct) != (True, True):
81
82
                   self.rct.move_ip(-sum_mv[0], -sum_mv[1])
               if not (sum mv[0] == 0 and sum mv[1] == 0):
83
84
                   self.img = __class__.imgs[tuple(sum_mv)]
85
               screen.blit(self.img, self.rct)
86
87
88
     class Beam:
89
90
            こうかとんが放つビームに関するクラス
91
92
           def __init__(self, bird:"Bird"):
93
               ビーム画像Surfaceを生成する
94
               引数 bird:ビームを放つこうかとん(Birdインスタンス)
95
96
97
               self.img = pg.image.load(f"fig/beam.png") # こうかとんSurface
               self.rct = self.img.get rect() # こうかとんRect
98
99
               self.rct.centery = bird.rct.centery # こうかとんの中心縦座標
               self.rct.left = bird.rct.right # ビームの左座標 = こうかとんの右座標
```

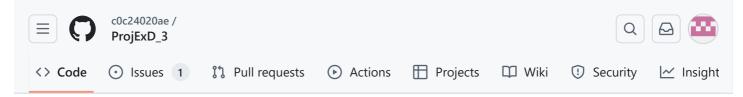
```
101
                 self.vx, self.vy = +5, 0
102
103 🗸
            def update(self, screen: pg.Surface):
104
                 ビームを速度ベクトルself.vx, self.vyに基づき移動させる
105
                 引数 screen:画面Surface
106
107
108
                 if check_bound(self.rct) == (True, True):
                     self.rct.move_ip(self.vx, self.vy)
109
110
                     screen.blit(self.img, self.rct)
111
112
113 ∨ class Bomb:
114
             爆弾に関するクラス
115
116
117 🗸
             def __init__(self, color: tuple[int, int, int], rad: int):
118
                 引数に基づき爆弾円Surfaceを生成する
119
120
                 引数1 color:爆弾円の色タプル
                 引数2 rad:爆弾円の半径
121
122
                 self.img = pg.Surface((2*rad, 2*rad))
123
124
                 pg.draw.circle(self.img, color, (rad, rad), rad)
125
                 self.img.set colorkey((0, 0, 0))
126
                 self.rct = self.img.get_rect()
127
                 self.rct.center = random.randint(0, WIDTH), random.randint(0, HEIGHT)
128
                 self.vx, self.vy = +5, +5
129
130 🗸
            def update(self, screen: pg.Surface):
131
                 爆弾を速度ベクトルself.vx, self.vyに基づき移動させる
132
                 引数 screen:画面Surface
133
134
135
                yoko, tate = check_bound(self.rct)
                 if not yoko:
136
                     self.vx *= -1
137
                 if not tate:
138
                    self.vy *= -1
139
140
                 self.rct.move_ip(self.vx, self.vy)
                 screen.blit(self.img, self.rct)
141
142
143
144 ∨ def main():
145
             pg.display.set_caption("たたかえ!こうかとん")
             screen = pg.display.set mode((WIDTH, HEIGHT))
146
147
             bg_img = pg.image.load("fig/pg_bg.jpg")
148
            bird = Bird((300, 200))
            beam = None
149
150
             \# bomb = Bomb((255, 0, 0), 10)
             bombs = [Bomb((255, 0, 0), 10) for _ in range(NUM_OF_BOMBS)]
151
             clock = pg.time.Clock()
            tmr = 0
153
             while True:
154
                for event in pg.event.get():
155
156
                     if event.type == pg.QUIT:
157
158
                     if event.type == pg.KEYDOWN and event.key == pg.K_SPACE:
```

```
# スペースキー押下でBeamクラスのインスタンス生成
159
160
                       beam = Beam(bird)
                screen.blit(bg_img, [0, 0])
161
162
163
                # if bomb is not None:
164
165
                for bomb in bombs:
                    if bird.rct.colliderect(bomb.rct):
166
                       # ゲームオーバー時に,こうかとん画像を切り替え,1秒間表示させる
167
168
                       bird.change_img(8, screen)
                       pg.display.update()
169
170
                       time.sleep(1)
171
                       return
172
173
               #if beam is not None:
                for j, bomb in enumerate(bombs):
174
175
                    if beam is not None:
                        if beam.rct.colliderect(bomb.rct): # ビームと爆弾の衝突判定
176
                           beam = None # ビームを消す
177
                           bombs[j] = None # 爆弾を消す
178
                           bird.change_img(6, screen) # よろこびエフェクト
179
                    bombs = [bomb for bomb in bombs if bomb is not None] # 撃ち落とされてない爆弾だけのリストに
180
181
182
                key lst = pg.key.get pressed()
183
                bird.update(key lst, screen)
184
                if beam is not None:
185
                    beam.update(screen)
                for bomb in bombs:
186
187
                    bomb.update(screen)
188
                pg.display.update()
189
                tmr += 1
                clock.tick(50)
190
191
192
193
       if __name__ == "__main__":
194
            pg.init()
            main()
195
196
            pg.quit()
            sys.exit()
197
```



## **Active branches**





## Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks or learn more about diff comparisons.

