












 c0a22100 / ProjExD_05

 Code  Issues  Pull requests  Actions  Projects  Security  Insights

  main  ProjExD_05 / musou_kokaton.py  Go to file

t



c0a22100 Update musou_kokaton.py

20 minutes ago



893 lines (768 loc) · 33.7 KB

Code

Blame

Raw



```
1  import math
2  import random
3  import sys
4  import time
5  from typing import Any
6
7  from pygame.locals import *
8  import pygame as pg
9  from pygame.sprite import AbstractGroup
10
11
12  WIDTH = 1600 # ゲームウィンドウの幅
13  HEIGHT = 900 # ゲームウィンドウの高さ
14
15  def start_screen(screen):
16      """
17      スタート画面を表示する
18      引数1 screen: 画面Surface
19      """
20      bg_img = pg.image.load("ex05/fig/pg_bg.jpg")
21      font_title = pg.font.Font(None, 150)
22      text_title = font_title.render("SHOOTING GAME", True, (0, 0, 0))
23      text_title_rect = text_title.get_rect(center=(WIDTH // 2, HEIGHT // 2 - 40))
24
25      font = pg.font.Font(None, 80)
26      text = font.render("Press SPACE BAR to start.", True, (0, 0, 0))
27      text_rect = text.get_rect(center=(WIDTH // 2, HEIGHT // 2 + 90))
28
29      screen.blit(bg_img, (0, 0))
30      screen.blit(text_title, text_title_rect) # タイトルを表示する
31      screen.blit(text, text_rect) # 操作方法を表示する
32      pg.display.flip()
33
34      while True:
35          for event in pg.event.get():
36              if event.type == pg.QUIT:
37                  pg.quit()
38                  sys.exit()
39              elif event.type == pg.KEYDOWN and event.key == pg.K_SPACE:
40                  return
41
42  def check_bound(obj: pg.Rect) -> tuple[bool, bool]:
```

```

43     """
44     オブジェクトが画面内か画面外かを判定し、真理値タプルを返す
45     引数 obj: オブジェクト (爆弾, こうかとん, ビーム) SurfaceのRect
46     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)
47     """
48     yoko, tate = True, True
49     if obj.left < 0 or WIDTH < obj.right: # 横方向のはみ出し判定
50         yoko = False
51     if obj.top < 0 or HEIGHT < obj.bottom: # 縦方向のはみ出し判定
52         tate = False
53     return yoko, tate
54
55
56 ✓ def calc_orientation(org: pg.Rect, dst: pg.Rect) -> tuple[float, float]:
57     """
58     orgから見て, dstがどこにあるかを計算し, 方向ベクトルをタプルで返す
59     引数1 org: 爆弾SurfaceのRect
60     引数2 dst: こうかとんSurfaceのRect
61     戻り値: orgから見たdstの方向ベクトルを表すタプル
62     """
63     x_diff, y_diff = dst.centerx-org.centerx, dst.centery-org.centery
64     norm = math.sqrt(x_diff**2+y_diff**2)
65     return x_diff/norm, y_diff/norm
66
67
68 ✓ class Bird(pg.sprite.Sprite):
69     """
70     ゲームキャラクター (こうかとん) に関するクラス
71     """
72     delta = { # 押下キーと移動量の辞書
73         pg.K_UP: (0, -1),
74         pg.K_DOWN: (0, +1),
75         pg.K_LEFT: (-1, 0),
76         pg.K_RIGHT: (+1, 0),
77     }
78
79 ✓ def __init__(self, num: int, xy: tuple[int, int]):
80     """
81     こうかとん画像Surfaceを生成する
82     引数1 num: こうかとん画像ファイル名の番号
83     引数2 xy: こうかとん画像の位置座標タプル
84     """
85     super().__init__()
86     img0 = pg.transform.rotozoom(pg.image.load(f"ex05/fig/{num}.png"), 0, 2.0)
87     img = pg.transform.flip(img0, True, False) # デフォルトのこうかとん
88     self.imgs = {
89         (+1, 0): img, # 右
90         (+1, -1): pg.transform.rotozoom(img, 45, 1.0), # 右上
91         (0, -1): pg.transform.rotozoom(img, 90, 1.0), # 上
92         (-1, -1): pg.transform.rotozoom(img0, -45, 1.0), # 左上
93         (-1, 0): img0, # 左
94         (-1, +1): pg.transform.rotozoom(img0, 45, 1.0), # 左下
95         (0, +1): pg.transform.rotozoom(img, -90, 1.0), # 下
96         (+1, +1): pg.transform.rotozoom(img, -45, 1.0), # 右下
97     }
98     self.dire = (+1, 0)
99     self.image = self.imgs[self.dire]
100     self.rect = self.image.get_rect()

```

```
101         self.rect.center = xy
102         self.speed = 10
103         self.state = "normal"
104
105 ✓ def change_img(self, num: int, screen: pg.Surface):
106     """
107     こうかとん画像を切り替え、画面に転送する
108     引数1 num: こうかとん画像ファイル名の番号
109     引数2 screen: 画面Surface
110     """
111
112     self.image = pg.transform.rotozoom(pg.image.load(f"ex05/fig/{num}.png"), 10, 2.0)
113
114
115     screen.blit(self.image, self.rect)
116
117 ✓ def update(self, key_lst: list[bool], screen: pg.Surface):
118     """
119     押下キーに応じてこうかとんを移動させる
120     引数1 key_lst: 押下キーの真理値リスト
121     引数2 screen: 画面Surface
122     """
123     sum_mv = [0, 0]
124     for k, mv in __class__.delta.items():
125         if key_lst[k]:
126             self.rect.move_ip(+self.speed*mv[0], +self.speed*mv[1])
127             sum_mv[0] += mv[0]
128             sum_mv[1] += mv[1]
129     if check_bound(self.rect) != (True, True):
130         for k, mv in __class__.delta.items():
131             if key_lst[k]:
132                 self.rect.move_ip(-self.speed*mv[0], -self.speed*mv[1])
133     if not (sum_mv[0] == 0 and sum_mv[1] == 0):
134         self.dire = tuple(sum_mv)
135         self.image = self.imgs[self.dire]
136
137         if self.state == "normal" or self.hyper_life < 0:
138             self.state = "normal"
139             self.image = self.imgs[self.dire]
140         elif self.state == "hyper":
141             self.image = pg.transform.laplacian(self.imgs[self.dire]) # 画像imageを変換
142             self.hyper_life -= 1 # 発動時間hyper_lifeを1減らす
143
144     screen.blit(self.image, self.rect)
145
146     def get_direction(self) -> tuple[int, int]:
147         return self.dire
148
149 ✓ def change_state(self, state: str, hyper_life: int):
150     """
151     追加機能3
152     引数1 state: 状態 ("hyper"と"normal")
153     引数2 hyper_life: 発動時間
154     """
155     self.state = state
156     self.hyper_life = hyper_life
157
158     #追加機能(残像 こうかとんjr)
```

```

159 ✓ class Small_Bird(pg.sprite.Sprite):
160     """
161     ゲームキャラクター（こうかとん）に関するクラス
162     """
163     delta2 = { # 押下キーと移動量の辞書
164         pg.K_UP: (0, -1),
165         pg.K_DOWN: (0, +1),
166         pg.K_LEFT: (-1, 0),
167         pg.K_RIGHT: (+1, 0),
168     }
169
170 ✓ def __init__(self, num: int, xy: tuple[int, int]):
171     """
172     こうかとん画像Surfaceを生成する
173     引数1 num: こうかとん画像ファイル名の番号
174     引数2 xy: こうかとん画像の位置座標タプル
175     """
176     super().__init__()
177     img10 = pg.transform.rotozoom(pg.image.load(f"ex05/fig/{num}.png"), 0, 2.0)
178     img10 = pg.transform.scale(img10, (70, 70))
179     img = pg.transform.flip(img10, True, False) # デフォルトのこうかとん
180     self.imgs = {
181         (+1, 0): img, # 右
182         (+1, -1): pg.transform.rotozoom(img, 45, 1.0), # 右上
183         (0, -1): pg.transform.rotozoom(img, 90, 1.0), # 上
184         (-1, -1): pg.transform.rotozoom(img10, -45, 1.0), # 左上
185         (-1, 0): img10, # 左
186         (-1, +1): pg.transform.rotozoom(img10, 45, 1.0), # 左下
187         (0, +1): pg.transform.rotozoom(img, -90, 1.0), # 下
188         (+1, +1): pg.transform.rotozoom(img, -45, 1.0), # 右下
189     }
190     self.dire = (+1, 0)
191     self.image2 = self.imgs[self.dire]
192     self.rect = self.image2.get_rect()
193     self.rect.center = xy
194     self.speed = 10
195     self.state2 = "small"
196
197 ✓ def change_img(self, num: int, screen: pg.Surface):
198     """
199     こうかとん画像を切り替え、画面に転送する
200     引数1 num: こうかとん画像ファイル名の番号
201     引数2 screen: 画面Surface
202     """
203     self.image2 = pg.transform.rotozoom(pg.image.load(f"fig/{num}.png"), 10, 2.0)
204     self.image2 = pg.transform.scale(self.image2, (50, 50))
205     screen.blit(self.image2, self.rect)
206
207 ✓ def update(self, key_lst: list[bool], screen: pg.Surface):
208     """
209     押下キーに応じてこうかとんを移動させる
210     引数1 key_lst: 押下キーの真理値リスト
211     引数2 screen: 画面Surface
212     """
213     sum_mv = [0, 0]
214     for k, mv in __class__.delta2.items():
215         if key_lst[k]:
216             self.rect.move_ip(+self.speed*mv[0], +self.speed*mv[1])

```

```

217         sum_mv[0] += mv[0]
218         sum_mv[1] += mv[1]
219     if check_bound(self.rect) != (True, True):
220         for k, mv in __class__.delta2.items():
221             if key_lst[k]:
222                 self.rect.move_ip(-self.speed*mv[0], -self.speed*mv[1])
223     if not (sum_mv[0] == 0 and sum_mv[1] == 0):
224         self.dire = tuple(sum_mv)
225         self.image2 = self.imgs[self.dire]
226
227         if self.state2 == "small" or self.hyper_life < 0:
228             self.state2 = "small"
229             self.image2 = self.imgs[self.dire]
230         elif self.state2 == "hyper":
231             self.image2 = pg.transform.laplacian(self.imgs[self.dire]) # 画像imageを変換
232             self.hyper_life -= 1 # 発動時間hyper_lifeを1減らす
233
234     screen.blit(self.image2, self.rect)
235
236     def get_direction(self) -> tuple[int, int]:
237         return self.dire
238
239     def change_state(self, state2: str, hyper_life: int):
240         """
241         追加機能3
242         引数1 state: 状態 ("hyper"と"normal")
243         引数2 hyper_life: 発動時間
244         """
245         self.state2 = state2
246         self.hyper_life = hyper_life
247
248
249     class Bomb(pg.sprite.Sprite):
250         """
251         爆弾に関するクラス
252         """
253         colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 255, 0), (255, 0, 255), (0, 255, 255)]
254
255     def __init__(self, emy: "Enemy", bird: Bird):
256         """
257         爆弾円Surfaceを生成する
258         引数1 emy: 爆弾を投下する敵機
259         引数2 bird: 攻撃対象のこうかとん
260         """
261         super().__init__()
262         rad = random.randint(10, 50) # 爆弾円の半径: 10以上50以下の乱数
263         color = random.choice(__class__.colors) # 爆弾円の色: クラス変数からランダム選択
264         self.image = pg.Surface((2*rad, 2*rad))
265         pg.draw.circle(self.image, color, (rad, rad), rad)
266         self.image.set_colorkey((0, 0, 0))
267         self.rect = self.image.get_rect()
268         # 爆弾を投下するemyから見た攻撃対象のbirdの方向を計算
269         self.vx, self.vy = calc_orientation(emy.rect, bird.rect)
270         self.rect.centerx = emy.rect.centerx
271         self.rect.centery = emy.rect.centery + emy.rect.height/2
272         self.speed = 6
273
274     def update(self):
275         """

```

```

275         """
276         爆弾を速度ベクトルself.vx, self.vyに基づき移動させる
277         引数 screen : 画面Surface
278         """
279         self.rect.move_ip(+self.speed*self.vx, +self.speed*self.vy)
280         if check_bound(self.rect) != (True, True):
281             self.kill()
282
283
284 ✓ class Beam(pg.sprite.Sprite):
285     """
286     ビームに関するクラス
287     """
288 ✓ def __init__(self, bird: Bird, angle_a: float=0):
289     """
290     ビーム画像Surfaceを生成する
291     引数 bird : ビームを放つこうかとん
292     """
293     super().__init__()
294     self.vx, self.vy = bird.get_direction()
295     angle = math.degrees(math.atan2(-self.vy, self.vx))+angle_a
296     self.size = random.uniform(1.5, 3.0)
297     self.image = pg.transform.rotozoom(pg.image.load(f"ex05/fig/beam.png"), angle, self.size)
298     self.vx = math.cos(math.radians(angle))
299     self.vy = -math.sin(math.radians(angle))
300     self.rect = self.image.get_rect()
301     self.rect.centery = bird.rect.centery+bird.rect.height*self.vy
302     self.rect.centerx = bird.rect.centerx+bird.rect.width*self.vx
303     self.speed = random.uniform(5, 20) #ビームのスピードをランダムに変更
304
305 ✓ def update(self):
306     """
307     ビームを速度ベクトルself.vx, self.vyに基づき移動させる
308     引数 screen : 画面Surface
309     """
310     self.rect.move_ip(+self.speed*self.vx, +self.speed*self.vy)
311     if check_bound(self.rect) != (True, True):
312         self.kill()
313
314
315 ✓ class Explosion(pg.sprite.Sprite):
316     """
317     爆発に関するクラス
318     """
319 ✓ def __init__(self, obj: "Bomb|Enemy", life: int):
320     """
321     爆弾が爆発するエフェクトを生成する
322     引数1 obj : 爆発するBombまたは敵機インスタンス
323     引数2 life : 爆発時間
324     """
325     super().__init__()
326     img = pg.image.load("ex05/fig/explosion.gif")
327     self.imgs = [img, pg.transform.flip(img, 1, 1)]
328     self.image = self.imgs[0]
329     self.rect = self.image.get_rect(center=obj.rect.center)
330     self.life = life
331
332 ✓ def update(self):
333     """

```

```

333
334         爆発時間を1減算した爆発経過時間_lifeに応じて爆発画像を切り替えることで
335         爆発エフェクトを表現する
336         """
337         self.life -= 1
338         self.image = self.imgs[self.life//10%2]
339         if self.life < 0:
340             self.kill()
341
342
343 ✓ class Enemy(pg.sprite.Sprite):
344     """
345     敵機に関するクラス
346     """
347     imgs = [pg.image.load(f"ex05/fig/alien{i}.png") for i in range(1, 4)]
348
349 ✓ def __init__(self):
350     super().__init__()
351     self.image = random.choice(__class__.imgs)
352     self.rect = self.image.get_rect()
353     self.rect.center = random.randint(0, WIDTH), 0
354     self.vy = +6
355     self.bound = random.randint(50, HEIGHT/2) # 停止位置
356     self.state = "down" # 降下状態or停止状態
357     self.interval = random.randint(50, 300) # 爆弾投下インターバル
358
359 ✓ def update(self):
360     """
361     敵機を速度ベクトルself.vyに基づき移動（降下）させる
362     ランダムに決めた停止位置_boundまで降下したら、_stateを停止状態に変更する
363     引数 screen: 画面Surface
364     """
365     if self.rect.centery > self.bound:
366         self.vy = 0
367         self.state = "stop"
368         self.rect.centery += self.vy
369
370
371 ✓ class Shield(pg.sprite.Sprite):
372     """
373     防御壁に関するクラス
374     引数1 bird:防御壁
375     引数2 life:防御壁の発動時間
376     """
377 ✓ def __init__(self, bird: Bird, life: int):
378     super().__init__()
379     self.vx, self.vy = bird.get_direction()
380     theta = math.atan2(-self.vy, self.vx) #こうかとんの向き（弧度法）
381     angle = math.degrees(theta) #こうかとんの向き（度数法）
382     self.image = pg.Surface((20, bird.rect.height*2))
383     self.image = pg.transform.rotozoom(self.image, angle, 1.0)
384     pg.draw.rect(self.image, (0, 0, 0), pg.Rect(0, 0, 20, bird.rect.height*2))
385     self.rect = self.image.get_rect()
386     self.rect.centerx = bird.rect.centerx+bird.rect.width*self.vx
387     self.rect.centery = bird.rect.centery+bird.rect.height*self.vy
388     self.life = life
389     # 爆弾を投下するemyから見た攻撃対象のbirdの方向を計算
390
391 ✓ def update(self):

```

```

391         self.update(screen)
392     """
393     爆発時間を1減算し、発動時間中は防御壁矩形を有効化
394     """
395     self.life -= 1
396     if self.life < 0:
397         self.kill()
398
399
400 ✓ class Score:
401     """
402     打ち落とした爆弾、敵機の数スコアとして表示するクラス
403     爆弾: 1点
404     敵機: 10点
405     """
406 ✓ def __init__(self):
407     self.font = pg.font.Font(None, 50)
408     self.color = (0, 0, 255)
409     self.score = 0
410     self.image = self.font.render(f"Score: {self.score}", 0, self.color)
411     self.rect = self.image.get_rect()
412     self.rect.center = 100, HEIGHT-50
413
414     def score_up(self, add):
415         self.score += add
416
417     def update(self, screen: pg.Surface):
418         self.image = self.font.render(f"Score: {self.score}", 0, self.color)
419         screen.blit(self.image, self.rect)
420
421
422 ✓ class NeoGravity(pg.sprite.Sprite):
423 ✓ def __init__(self, life: int):
424     super().__init__()
425     self.image = pg.Surface((WIDTH, HEIGHT))
426     pg.draw.rect(self.image, (10, 10, 10), pg.Rect(0, 0, WIDTH, HEIGHT))
427     self.image.set_colorkey((0, 0, 0))
428     self.image.set_alpha(200)
429     self.rect = self.image.get_rect()
430     self.rect.center = WIDTH/2, HEIGHT/2
431     self.life = life
432
433     def update(self):
434         self.life -= 1
435         if self.life < 0:
436             self.kill()
437
438
439 ✓ class Gravity(pg.sprite.Sprite):
440 ✓ def __init__(self, bird, life):
441     super().__init__()
442     rad = 200
443     self.life = life
444     self.image = pg.Surface((2*rad, 2*rad))
445     pg.draw.circle(self.image, (1, 1, 1), (rad, rad), rad)
446     self.rect = self.image.get_rect()
447     self.image.set_colorkey("black")
448     self.image.set_alpha(127) #黒を透明化
449     self.rect.center = bird.rect.center #self.rectがこうかとんを追う

```



```

450
451 ✓ def update(self, bird):
452     self.rect.center = bird.rect.center
453     self.life = self.life - 1
454     if self.life <= 0:
455         self.kill()
456
457
458 ✓ class Aura(pg.sprite.Sprite):
459     """
460     こうかとんにオーラを纏わせる
461     """
462 ✓ def __init__(self, bird):
463     super().__init__()
464     bird_rect = bird.rect
465     self.image = pg.Surface((10, 10))
466     pg.draw.rect(self.image, "purple", (0, 0, 10, 10))
467     self.image.set_alpha(91) #purpleを透明化
468     self.rect = self.image.get_rect()
469     self.life = 35 #オーラブロックの生成個数
470     self.rect[:-2] = \
471         random.randint(bird_rect[0], bird_rect[0]+bird_rect[2]), \
472         random.randint(bird_rect[1], bird_rect[1]+bird_rect[3])
473     #ブロックをこうかとの周りにランダムに生成
474
475     def update(self):
476         self.life -= 1
477         if self.life < 0:
478             self.kill()
479
480 ✓ class BeamPlus(pg.sprite.Sprite):
481     """
482     2発のビームの発射を可能にするクラス
483     """
484 ✓ def __init__(self, bird: Bird):
485     """
486     ビーム画像Surfaceを生成する
487     引数 bird: ビームを放つこうかとん
488     """
489     super().__init__()
490     self.vx, self.vy = bird.get_direction()
491     angle = math.degrees(math.atan2(-self.vy, self.vx))
492     self.image = pg.Surface((bird.rect.height/2, 20))
493     self.size = random.uniform(0.1, 1.0) #ビームの区別をつけるため小さくしている
494     self.image = pg.transform.rotozoom(self.image, angle, self.size)
495     pg.draw.rect(self.image, (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))
496     self.vx = math.cos(math.radians(angle))
497     self.vy = -math.sin(math.radians(angle))
498     self.rect = self.image.get_rect()
499     self.rect.centery = bird.rect.centery+bird.rect.height*self.vy
500     self.rect.centerx = bird.rect.centerx+bird.rect.width*self.vx
501     self.speed = 30 #大きさを小さくした分性能の差を無くすためにスピードを上げる
502
503 ✓ def update(self):
504     """
505     ビームを速度ベクトルself.vx, self.vyに基づき移動させる
506     引数 screen : 画面Surface
507     """

```

```

508         self.rect.move_ip(+self.speed*self.vx, +self.speed*self.vy)
509         if check_bound(self.rect) != (True, True):
510             self.kill()
511
512 ✓ class FrontKoukaShield(pg.sprite.Sprite):
513     """
514     こうかとんの前に防御壁を作るクラス
515     引数1 bird 防御壁
516     引数2 life 防御壁の発動秒数
517     """
518 ✓ def __init__(self, bird: Bird, life: int):
519     super().__init__()
520     self.vx, self.vy = bird.get_direction()
521     self.life = life
522     theta = math.atan2(-self.vy, self.vx) # こうかとんの向き(弧度法)
523     angle = math.degrees(theta) # こうかとんの向き(度数法)
524     self.image = pg.Surface((20, bird.rect.height * 2))
525     self.image = pg.transform.rotozoom(self.image, angle, 1.0)
526     pg.draw.rect(self.image, (255, 0, 0), pg.Rect(0, 0, 20, bird.rect.height * 2))
527     self.rect = self.image.get_rect()
528     self.rect.centerx = bird.rect.centerx + bird.rect.width*self.vx # self.rect.centerxがこうかとん
529     self.rect.centery = bird.rect.centery + bird.rect.height*self.vy # self.rect.centeryがこうかとん
530     # 爆弾を落下するemyから見た攻撃対象のbirdの方向を計算
531
532 ✓ def update(self, bird: Bird):
533     """
534     防御壁の発動秒数を1減算し、発動中はこうかとんの前に防御壁を有効化
535     """
536     self.rect.centerx = bird.rect.centerx + bird.rect.width*self.vx
537     self.rect.centery = bird.rect.centery + bird.rect.height*self.vy
538     self.life -= 1
539     if self.life < 0:
540         self.kill()
541
542
543 ✓ class BackKoukaShield(pg.sprite.Sprite):
544     """
545     こうかとんの後ろに防御壁を作るクラス
546     """
547
548 ✓ def __init__(self, bird: Bird, life: int):
549     """
550     引数1 bird 防御壁
551     引数2 life 防御壁の発動秒数
552     """
553     super().__init__()
554     self.vx, self.vy = bird.get_direction()
555     self.life = life
556     rev_theta = math.atan2(-self.vy, -self.vx)
557     angle2 = math.degrees(rev_theta)
558     self.image = pg.Surface((20, bird.rect.height*2))
559     self.image = pg.transform.rotozoom(self.image, angle2, 1.0)
560     pg.draw.rect(self.image, (255, 255, 0), pg.Rect(0, 0, 20, bird.rect.height * 2))
561     self.rect = self.image.get_rect()
562     self.rect.centerx = bird.rect.centerx + bird.rect.width*(-self.vx) # self.rect.centerxがこうかと
563     self.rect.centery = bird.rect.centery + bird.rect.height*(-self.vy) # self.rect.centeryがこうか
564
565 ✓ def update(self, bird: Bird):

```

```

566         """
567         防御壁の発動時間を1減算、発動中はこうかとんの後ろに防御壁を展開
568         """
569         self.rect.centerx = bird.rect.centerx + bird.rect.width*(-self.vx)
570         self.rect.centery = bird.rect.centery + bird.rect.height*(-self.vy)
571         self.life -= 1
572         if self.life < 0:
573             self.kill()
574
575
576 ✓ class KoukaBall(pg.sprite.Sprite):
577     """
578     こうかとんがこうかボールを放てるようにするクラス
579     """
580 ✓ def __init__(self, bird: Bird):
581     """
582     こうかボールを描画する
583     引数1 bird こうかボールを放つこうかとん
584     """
585     super().__init__()
586     self.vx, self.vy = bird.get_direction()
587     angle = math.degrees(math.atan2(-self.vy, self.vx))
588     rad = 100
589     self.image = pg.Surface((2*rad, 2*rad))
590     color = self.image.fill("white")
591
592     # 白を消す処理を入れる
593     self.image.set_alpha(200)
594     pg.draw.circle(self.image, "mediumorchid", (rad, rad), rad)
595     self.image.set_colorkey("white")
596     self.vx = math.cos(math.radians(angle))
597     self.vy = -math.sin(math.radians(angle))
598     self.rect = self.image.get_rect()
599     self.rect.centery = bird.rect.centery+bird.rect.height*self.vy
600     self.rect.centerx = bird.rect.centerx+bird.rect.width*self.vx
601     self.speed = 10
602
603 ✓ def update(self):
604     """
605     ビームを速度ベクトルself.vx, self.vyに基づき移動させる
606     引数 screen : 画面Surface
607     """
608     self.rect.move_ip(+self.speed*self.vx, +self.speed*self.vy)
609     if check_bound(self.rect) != (True, True):
610         self.kill()
611
612
613 ✓ class Beamplusalpha:
614     """
615     全方向に速度が不特定のビームを放つ処理
616     """
617 ✓ def __init__(self, bird: Bird, num: int):
618     """
619     ビーム画像Surfaceを生成する
620     引数 bird : ビームを放つこうかとん
621     引数 num : 発射するビームの数
622     bird, numの初期化を行う
623     """

```

```

624         self.beam_list = [] #リストの生成
625         self.bird = bird
626         self.num = num
627     def gen_beams(self):
628         """
629         角度をつけてビームを出す処理
630         """
631         vx, vy = self.bird.get_direction()
632         for i in range(-180, 181, int(100/(self.num-1))): #-180度から180度の間でint(100/(self.num-1))おきに
633             self.beam_list.append(Beam(self.bird, i)) #ビームの値をリストに代入
634         return self.beam_list
635
636
637     class Levelup:
638     def __init__(self):
639         """
640         ビームの結果に応じてレベルの上がる処理
641         """
642         self.font = pg.font.Font(None, 50)
643         self.color = (247, 146, 19)
644         self.level = 0
645         self.image = self.font.render(f"LEVEL: {self.level}", 0, self.color) #現在のレベル表示
646         self.rect = self.image.get_rect()
647         self.rect.center = WIDTH-80, 30 #self.imageの内容の表示位置
648
649     def levelup(self, add):
650         """
651         スコア増加の処理
652         """
653         self.level += add
654
655     def update(self, screen: pg.Surface):
656         self.image = self.font.render(f"LEVEL: {self.level}", 0, self.color)
657         screen.blit(self.image, self.rect)
658
659     def main():
660         pg.display.set_caption("真！ どうかとん無双")
661         screen = pg.display.set_mode((WIDTH, HEIGHT))
662         bg_img = pg.image.load("ex05/fig/pg_bg.jpg")
663         score = Score()
664
665         start_screen(screen)
666
667         bird = Bird(3, (900, 400))
668         s_bird = Small_Bird(3, (800, 300))
669         bombs = pg.sprite.Group()
670         beams = pg.sprite.Group()
671         exps = pg.sprite.Group()
672         emys = pg.sprite.Group()
673         neogrs = pg.sprite.Group()
674         gravities = pg.sprite.Group()
675         pluses = pg.sprite.Group()
676         levels = Levelup()
677         levels.level = 1
678         auras = pg.sprite.Group()
679
680         score.score = 0
681

```

```
682     shields = pg.sprite.Group()
683
684     FrontKS = pg.sprite.Group()
685     BackKS = pg.sprite.Group()
686     Kkball = pg.sprite.Group()
687     tmr = 0
688     clock = pg.time.Clock()
689     while True:
690         key_lst = pg.key.get_pressed()
691         for event in pg.event.get():
692             if event.type == pg.QUIT:
693                 return 0
694             if event.type == pg.KEYDOWN and event.key == pg.K_SPACE:
695                 beams.add(Beam(bird))
696                 for i in range(1,100):
697                     if score.score >= i*10:
698                         pluses.add(BeamPlus(bird))
699
700
701             if event.type == pg.KEYDOWN and event.key == pg.K_SPACE:
702                 beams.add(Beam(s_bird))
703             if event.type == pg.KEYDOWN and event.key == pg.K_LSHIFT: # 左シフトが押されているか判定
704                 bird.speed = 20 # スピードアップ
705             if event.type == pg.KEYUP and event.key == pg.K_LSHIFT: # 左シフトが押された状態から離れたら
706                 bird.speed = 10 # もとのスピードに戻る
707             if event.type == pg.KEYDOWN and event.key == pg.K_LSHIFT: # 左シフトが押されているか判定
708                 s_bird.speed = 20 # スピードアップ
709             if event.type == pg.KEYUP and event.key == pg.K_LSHIFT: # 左シフトが押された状態から離れたら
710                 s_bird.speed = 10 # もとのスピードに戻る
711             if event.type == pg.KEYDOWN and event.key == pg.K_RETURN:
712                 if score.score > 200:
713                     neogrs.add(NeoGravity(400))
714                     score.score_up(-200)
715
716
717             # 追加機能3
718             if event.type == pg.KEYDOWN and event.key == pg.K_RSHIFT and score.score > 100: #→Shiftキー
719                 score.score -= 100
720                 bird.change_state("hyper", 500)
721
722             if event.type == pg.KEYDOWN and event.key == pg.K_TAB and score.score > 50:
723                 #矢印キーとtabキーが押されて、スコアが50以上ならスコアを-50する。
724                 gravities.add(Gravity(bird, 500))
725                 score.score -= 50
726
727             if event.type == pg.KEYDOWN and event.key == pg.K_CAPSLOCK and len(shields) == 0 :
728                 if score.score > 50:
729                     score.score_up(-50)
730                     shields.add(Shield(bird, 400))
731
732             if event.type == pg.KEYDOWN and event.key == pg.K_F1 and score.score >40:
733                 levels.levelup(3) #レベル3アップ
734                 beams.add(Beamplusalpha(bird, 6).gen_beams())
735                 score.score_up(-40)
736
737
738             if event.type == pg.KEYDOWN and event.key == pg.K_RSHIFT and score.score > 100: #→Shiftキー
739                 score.score -= 100 #
```

```
740         s_bird.change_state("hyper", 500)
741
742     if event.type == pg.KEYDOWN and event.key == pg.K_x and len(FrontKS) == 0 :
743         if score.score > 50:
744             score.score_up(-50)
745             FrontKS.add(FrontKoukaShield(bird, 400))
746             BackKS.add(BackKoukaShield(bird, 400))
747
748     if event.type == pg.KEYDOWN and event.key == pg.K_d and score.score > 70:
749         Kkball.add(KoukaBall(bird))
750         score.score -= 70
751
752     screen.blit(bg_img, [0, 0])
753
754
755     if tmr%200 == 0: # 200フレームに1回, 敵機を出現させる
756         emys.add(Enemy())
757
758     for emy in emys:
759         if emy.state == "stop" and tmr%emy.interval == 0:
760             # 敵機が停止状態に入ったら, intervalに応じて爆弾投下
761             bombs.add(Bomb(emy, bird))
762
763     for emy in pg.sprite.groupcollide(emys, beams, True, True).keys():
764         exps.add(Explosion(emy, 100)) # 爆発エフェクト
765         score.score_up(10) # 10点アップ
766         levels.levelup(1) # レベル1アップ
767         bird.change_img(6, screen) # こうかたん喜びエフェクト
768         bird.change_img(6, screen) # こうかたん喜びエフェクト
769
770     for emy in pg.sprite.groupcollide(emys, beams, True, True).keys():
771         exps.add(Explosion(emy, 100)) # 爆発エフェクト
772         score.score_up(10) # 10点アップ
773         bird.change_img(6, screen) # こうかたん喜びエフェクト
774
775     for bomb in pg.sprite.groupcollide(bombs, beams, True, True).keys():
776         exps.add(Explosion(bomb, 50)) # 爆発エフェクト
777         score.score_up(1) # 1点アップ
778
779
780     for bomb in pg.sprite.spritecollide(bird, bombs, True):
781         if bird.state == "normal":
782             bird.change_img(8, screen) # こうかたん悲しみエフェクト
783             score.update(screen)
784             pg.display.update()
785             time.sleep(2)
786             return
787         elif bird.state == "hyper":
788             exps.add(Explosion(bomb, 50)) # 爆発エフェクト
789             score.score_up(1) # 1点アップ
790
791
792     for emy in pg.sprite.groupcollide(emys, gravities, True, False).keys():
793         exps.add(Explosion(emy, 100)) # 爆発エフェクト
794         score.score_up(10) # 10点アップ
795         levels.levelup(1) # レベルが1上がる
796         bird.change_img(6, screen) # こうかたん喜びエフェクト
797         s_bird.change_img(6, screen) # こうかたん喜びエフェクト
```

```
798
799     for bomb in pg.sprite.groupcollide(bombs, gravities, True, False).keys():
800         exps.add(Explosion(bomb, 50)) # 爆発エフェクト
801         score.score_up(1) # 1点アップ
802
803     for bomb in pg.sprite.groupcollide(bombs, shields, True, False).keys():
804         exps.add(Explosion(bomb, 50)) # 爆発エフェクト
805         score.score_up(1) # 1点アップ
806
807     for bomb in pg.sprite.groupcollide(bombs, FrontKS, True, False).keys():
808
809         exps.add(Explosion(bomb, 50)) # 爆発エフェクト
810         score.score_up(1) # 1点アップ
811
812     for bomb in pg.sprite.groupcollide(bombs, BackKS, True, False).keys():
813
814         exps.add(Explosion(bomb, 50)) # 爆発エフェクト
815         score.score_up(1) # 1点アップ
816
817     for emy in pg.sprite.groupcollide(emy, Kkball, True, False).keys():
818         exps.add(Explosion(emy, 50))
819         score.score_up(10)
820     for bomb in pg.sprite.groupcollide(bombs, Kkball, True, False).keys():
821         exps.add(Explosion(bomb, 50))
822         score.score_up(1)
823
824     if len(pg.sprite.spritecollide(bird, bombs, True)) != 0:
825         bird.change_img(8, screen) # こうかとおん悲しみエフェクト
826         score.update(screen)
827         pg.display.update()
828         time.sleep(2)
829     return
830
831     for bomb in pg.sprite.groupcollide(bombs, neogrs, True, False).keys():
832         exps.add(Explosion(bomb, 50)) # 爆発エフェクト
833         score.score_up(1) # 1点アップ
834
835     for emy in pg.sprite.groupcollide(emy, neogrs, True, False).keys():
836         exps.add(Explosion(emy, 100)) # 爆発エフェクト
837         score.score_up(10) # 10点アップ
838         bird.change_img(6, screen) # こうかとおん喜びエフェクト
839
840     for emy in pg.sprite.groupcollide(emy, pluses, True, True).keys():
841         exps.add(Explosion(emy, 100)) # 爆発エフェクト
842         levels.levelup(1) # レベルが1上がる
843         score.score_up(10) # 10点アップ
844         bird.change_img(6, screen) # こうかとおん喜びエフェクト
845
846     for bomb in pg.sprite.groupcollide(bombs, pluses, True, True).keys():
847         exps.add(Explosion(bomb, 50)) # 爆発エフェクト
848         score.score_up(1) # 1点アップ
849
850     bird.update(key_lst, screen)
851     s_bird.update(key_lst, screen)
852     beams.update()
853     beams.draw(screen)
854     emys.update()
855     emys.draw(screen)
```

```
856         bombs.update()
857         bombs.draw(screen)
858         exps.update()
859         exps.draw(screen)
860         neogrs.update()
861         neogrs.draw(screen)
862         score.update(screen)
863         gravities.update(bird)
864         gravities.draw(screen)
865         auras.update()
866         auras.add(Aura(bird))
867         auras.draw(screen)
868
869
870         shields.update()
871         shields.draw(screen)
872         pluses.update()
873         pluses.draw(screen)
874         levels.update(screen)
875
876         FrontKS.update(bird)
877         FrontKS.draw(screen)
878         BackKS.update(bird)
879         BackKS.draw(screen)
880         Kkball.update()
881         Kkball.draw(screen)
882         pg.display.update()
883         tmr += 1
884         clock.tick(50)
885
886
887 if __name__ == "__main__":
888     pg.init()
889     main()
890     pg.quit()
891     sys.exit
892     sys.exit()
```