```python
import os
import pygame as pg
import random
import sys
import time


WIDTH, HEIGHT = 1100, 650
start_time = None
time_limit = 10
DELTA = {pg.K_UP:   ( 0,-5),
         pg.K_DOWN: ( 0,+5),
         pg.K_LEFT: (-5, 0),
         pg.K_RIGHT:(+5, 0),
         }

os.chdir(os.path.dirname(os.path.abspath(__file__)))

def check_bound(obj_rct: pg.rect) -> tuple[bool, bool]:
    """
    引数　：こうかとん または 爆弾のRect
    戻り値：真理値タプル（横判定結果、縦判定結果）
    画面内ならTrue 画面外ならFalse
    """
    yoko,tate = True,True
    if obj_rct.left < WIDTH*2/5 or WIDTH*3/5 < obj_rct.right:
        yoko = False
```



```python
def check_bound(obj_rct: pg.rect) -> tuple[bool, bool]:
class Bird:
    """
    ゲームキャラクター（こうかとん）に関するクラス
    """
    delta = {  # 押下キーと移動量の辞書
        pg.K_UP: (0, -5),
        pg.K_DOWN: (0, +5),
        pg.K_LEFT: (-5, 0),
        pg.K_RIGHT: (+5, 0),
```

```python
    }
    img0 = pg.transform.rotozoom(pg.image.load("fig/3.png"), 0, 0.9)
    img = pg.transform.flip(img0, True, False)  # デフォルトのこうかとん（右向き）
    imgs = {  # 0度から反時計回りに定義
        (+5, 0): img,  # 右
        (+5, -5): pg.transform.rotozoom(img, 45, 0.9),  # 右上
        (0, -5): pg.transform.rotozoom(img, 90, 0.9),  # 上
        (-5, -5): pg.transform.rotozoom(img0, -45, 0.9),  # 左上
        (-5, 0): img0,  # 左
        (-5, +5): pg.transform.rotozoom(img0, 45, 0.9),  # 左下
        (0, +5): pg.transform.rotozoom(img, -90, 0.9),  # 下
        (+5, +5): pg.transform.rotozoom(img, -45, 0.9),  # 右下
    }

    def __init__(self, xy: tuple[int, int]):
        """
        こうかとん画像Surfaceを生成する
        引数 xy：こうかとん画像の初期位置座標タプル
        """
        self.img = __class__.imgs[(+5, 0)]
        self.rct: pg.Rect = self.img.get_rect()
        self.rct.center = xy

    def change_img(self, num: int, screen: pg.Surface):
        """
        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
        引数2 screen：画面Surface
        """
        self.img = pg.transform.rotozoom(pg.image.load(f"fig/{num}.png"), 0, 0.9)
        screen.blit(self.img, self.rct)

    def update(self, key_lst: list[bool], screen: pg.Surface):
        """
        押下キーに応じてこうかとんを移動させる
        引数1 key_lst：押下キーの真理値リスト
        引数2 screen：画面Surface
        """
        sum_mv = [0, 0]
        for k, mv in __class__.delta.items():
            if key_lst[k]:
                sum_mv[0] += mv[0]
                sum_mv[1] += mv[1]
        self.rct.move_ip(sum_mv)
        if check_bound(self.rct) != (True, True):
            self.rct.move_ip(-sum_mv[0], -sum_mv[1])
        if not (sum_mv[0] == 0 and sum_mv[1] == 0):
            self.img = __class__.imgs[tuple(sum_mv)]
        screen.blit(self.img, self.rct)

def gameover(screen):
    """
    ゲームオーバー時の処理
    引数:screen
    戻り値:無し
    背景をブラックアウト、GameOverの文字を中心に、その上に泣いているこうかとんの画像を配置
    """
    black= pg.Surface((WIDTH,HEIGHT))
```

```python
        pg.draw.rect(black,(0,0,0),(0,0,WIDTH,HEIGHT))
        black.set_alpha(100)
        screen.blit(black,(0,0))
        cry_img = pg.transform.rotozoom(pg.image.load("fig/8.png"),0,1.2)
        cry_rct = cry_img.get_rect(center=(WIDTH//2,HEIGHT//2-50))
        screen.blit(cry_img,cry_rct)
        fonto = pg.font.Font(None,80)
        txt = fonto.render("GameOver",True,
                           (0,150,255))
        txt_rct = txt.get_rect()
        txt_rct.center = WIDTH//2,HEIGHT//2
        screen.blit(txt,txt_rct)
        pg.display.update()
        time.sleep(5)

    def enemy(num,screen: pg.surface):
        if num == 1:
            en_img1 = pg.transform.rotozoom(pg.image.load("fig/DQMJ2.webp"),0,0.6)
            en_rct1 = en_img1.get_rect()
            en_rct1.centerx = 150
            en_rct1.centery = HEIGHT/2
            screen.blit(en_img1,en_rct1)
            stage1()

        elif num == 2:
            en_img2 = pg.transform.rotozoom(pg.image.load("fig/en1.png"),0,0.4)
            en_rct2 = en_img2.get_rect()
            en_rct2.centerx = WIDTH-150
            en_rct2.centery = HEIGHT/2
            screen.blit(en_img2,en_rct2)
            stage2()

        elif num == 3:
            en_img3 = pg.transform.rotozoom(pg.image.load("fig/en5.png"),0,1)
            en_img4 = pg.transform.rotozoom(pg.image.load("fig/en6.png"),0,1)
            en_rct3 = en_img3.get_rect()
            en_rct3.centerx = 150
            en_rct3.centery = HEIGHT/2
            en_rct4 = en_img4.get_rect()
            en_rct4.centerx = WIDTH-150
            en_rct4.centery = HEIGHT/2
            screen.blit(en_img3,en_rct3)
            screen.blit(en_img4,en_rct4)
            stage3()

        elif num == 4:
            en_img5 = pg.transform.rotozoom(pg.image.load("fig/en7.png"),0,0.6)
            en_rct5 = en_img5.get_rect()
            en_rct5.centerx = 150
            en_rct5.centery = HEIGHT/2
            screen.blit(en_img5,en_rct5)
            stageEX()


    def stage1():
        return 0

    def stage2():
```

```python
        return 0

    def stage3():
        return 0

    def stageEX():
        return 0

    def timescore(screen, stage):
        global start_time
        if start_time is None:
            start_time = time.time()  # タイマー開始

        spent_time = time.time() - start_time
        end_time = max(0, time_limit - spent_time)

        font = pg.font.Font(None, 36)
        time_text = font.render(f"Time: {int(end_time)}s", True, (255, 255, 255))
        screen.blit(time_text, (10, 10))

        # EXステージでのクリア表示
        if stage == 4 and end_time <= 0:
            black_scr = pg.Surface((WIDTH, HEIGHT))
            pg.draw.rect(black_scr, (0, 0, WIDTH, HEIGHT))
            black_scr.set_alpha(180)
            screen.blit(black_scr, (0, 0))

            kk_img = pg.transform.rotozoom(pg.image.load("fig/8.png"), 0, 0.9)
            kk_rct = kk_img.get_rect()
            kk_rct.center = WIDTH / 2 + 180, HEIGHT / 2
            screen.blit(kk_img, kk_rct)
            kk_rct.center = WIDTH / 2 - 180, HEIGHT / 2
            screen.blit(kk_img, kk_rct)

            clear_font = pg.font.Font(None, 80)
            clear_text = clear_font.render("クリア", True, (255, 255, 255))
            clear_text_rect = clear_text.get_rect()
            clear_text_rect.center = WIDTH / 2, HEIGHT / 2
            screen.blit(clear_text, clear_text_rect)
            pg.display.update()

            # 'n' キー待機
            waiting = True
            while waiting:
                for event in pg.event.get():
                    if event.type == pg.QUIT:
                        pg.quit()
                        sys.exit()
                    elif event.type == pg.KEYDOWN and event.key == pg.K_n:
                        waiting = False
                        start_time = None

            return stage + 1  # ここでステージ更新を停止または調整

        elif end_time <= 0:
            # 次のステージへの移行とタイマーリセット
            black_scr = pg.Surface((WIDTH, HEIGHT))
            black_scr.fill((0, 0, 0))
            black_scr.set_alpha(180)
```

```
            black_scr.set_alpha(180)
            screen.blit(black_scr, (0, 0))

            msg = font.render("Please Press 'N'", True, (255, 255, 255))
            screen.blit(msg, (WIDTH // 2 - msg.get_width() // 2, HEIGHT // 2))
            pg.display.update()

            # 'n' キー待機
            waiting = True
            while waiting:
                for event in pg.event.get():
                    if event.type == pg.QUIT:
                        pg.quit()
                        sys.exit()
                    elif event.type == pg.KEYDOWN and event.key == pg.K_n:
                        waiting = False
                        start_time = None

            return stage + 1

        return stage


    def skill():
        return 0

    def main():
        pg.display.set_caption("避けろ！こうかとん")
        screen = pg.display.set_mode((WIDTH, HEIGHT))
        bg_img = pg.transform.rotozoom(pg.image.load("fig/bg.png"),0,1.9)
        bird = Bird([WIDTH/2, HEIGHT/2])
        stage = 1

        clock = pg.time.Clock()
        tmr = 0
        while True:
            for event in pg.event.get():
                if event.type == pg.QUIT:
                    return
            screen.blit(bg_img, [0, 0])


            key_lst = pg.key.get_pressed()
            sum_mv = [0, 0]


            for key, tpl in DELTA.items():
                if key_lst[key]:
                    sum_mv[0] += tpl[0] #横方向
                    sum_mv[1] += tpl[1] #縦方向


            bird.update(key_lst, screen)
            enemy(stage,screen)
            stage = timescore(screen, stage)
            skill()
            pg.display.update()
            tmr += 1
            clock.tick(50)
```

```python


if __name__ == "__main__":
    pg.init()
    main()
    pg.quit()
    sys.exit()
```