```
import asyncio
import math
import random
import pygame

W, H = 360, 640
GRAVITY = 900
R = 24               # fruit size

pygame.init()
screen = pygame.display.set_mode((W, H))
pygame.display.set_caption('Fruit Slicer')
clock = pygame.time.Clock()
font = pygame.font.Font(None, 32)
big = pygame.font.Font(None, 56)

FRUITS = {
    'apple': ((220, 40, 40), (255, 225, 225)),
    'orange': ((255, 150, 0), (255, 215, 130)),
    'melon': ((40, 160, 60), (250, 70, 90)),
}

state = {}


def reset():
    state.update(items=[], halves=[], trail=[], finger=None, prev=None,
                 spawn=0.5, score=0, lives=3, over=False, flash=0.0)
    state.setdefault('best', 0)


def say(text, f, color, x, y):
    img = f.render(text, True, color)
    screen.blit(img, img.get_rect(center=(x, y)))


def seg_dist(px, py, ax, ay, bx, by):
    # distance from point (px,py) to the line segment a-b (the swipe)
    dx, dy = bx - ax, by - ay
    if dx == 0 and dy == 0:
        return math.hypot(px - ax, py - ay)
    t = ((px - ax) * dx + (py - ay) * dy) / (dx * dx + dy * dy)
    t = max(0, min(1, t))
    return math.hypot(px - (ax + t * dx), py - (ay + t * dy))


def throw_items():
    score = state['score']
    for _ in range(random.choice([1, 1, 2, 3])):
        bomb = random.random() < min(0.15 + 0.005 * score, 0.3)
        kind = 'bomb' if bomb else random.choice(list(FRUITS))
```

```
        x = random.randint(60, W - 60)
        state['items'].append({
            'x': x, 'y': H + 30, 'kind': kind,
            'vx': (W / 2 - x) * 0.25 + random.uniform(-40, 40),
            'vy': -random.uniform(850, 1000),
        })


def slice_item(item):
    state['items'].remove(item)
    if item['kind'] == 'bomb':
        state['lives'] -= 1
        state['flash'] = 0.3
    else:
        state['score'] += 1
        outer, inner = FRUITS[item['kind']]
        for side in (-1, 1):                    # two halves fly apart
            state['halves'].append({'x': item['x'], 'y': item['y'], 'color': inner, 'rim': outer,
                                    'vx': side * 120, 'vy': item['vy'] * 0.3, 't': 0.0})


def update(dt):
    s = state

    s['spawn'] -= dt
    if s['spawn'] <= 0:
        throw_items()
        s['spawn'] = max(0.55, 1.3 - 0.02 * s['score'])

    # swipe: check the line between last finger position and the new one
    if s['finger'] and s['prev']:
        for item in s['items'][:]:
            if seg_dist(item['x'], item['y'], *s['prev'], *s['finger']) < R:
                slice_item(item)

    for item in s['items'][:]:
        item['vy'] += GRAVITY * dt
        item['x'] += item['vx'] * dt
        item['y'] += item['vy'] * dt
        if item['y'] > H + R and item['vy'] > 0:       # fell down
            if item['kind'] != 'bomb':
                s['lives'] -= 1                        # missed a fruit
            s['items'].remove(item)

    for h in s['halves'][:]:
        h['vy'] += GRAVITY * dt
        h['x'] += h['vx'] * dt
        h['y'] += h['vy'] * dt
        h['t'] += dt
        if h['t'] > 0.8:
            s['halves'].remove(h)

    if s['lives'] <= 0:
        s['over'] = True
        s['best'] = max(s['best'], s['score'])
```

```
        s['items'] = []


def draw_item(item):
    x, y = int(item['x']), int(item['y'])
    if item['kind'] == 'bomb':
        pygame.draw.circle(screen, (30, 30, 30), (x, y), R)
        pygame.draw.circle(screen, (90, 90, 90), (x - 7, y - 7), 5)
        pygame.draw.line(screen, (150, 110, 60), (x, y - R), (x + 8, y - R - 10), 3)
        pygame.draw.circle(screen, (255, 120, 0), (x + 8, y - R - 10), 4)
    else:
        outer = FRUITS[item['kind']][0]
        pygame.draw.circle(screen, outer, (x, y), R)
        pygame.draw.circle(screen, (255, 255, 255), (x - 8, y - 8), 5)


def draw():
    s = state
    screen.fill((120, 20, 20) if s['flash'] > 0 else (30, 35, 60))

    for h in s['halves']:
        pygame.draw.circle(screen, h['rim'], (int(h['x']), int(h['y'])), int(R * 0.8))
        pygame.draw.circle(screen, h['color'], (int(h['x']), int(h['y'])), int(R * 0.55))
    for item in s['items']:
        draw_item(item)

    pts = s['trail']                                    # the white swipe trail
    for i in range(1, len(pts)):
        pygame.draw.line(screen, (255, 255, 255), pts[i - 1][:2], pts[i][:2], 4)

    say(f"Score: {s['score']}   Lives: {s['lives']}", font, (255, 255, 255), W // 2, 24)
    if s['score'] == 0 and not s['over']:
        say('Swipe the fruits! Avoid bombs!', font, (255, 255, 255), W // 2, 60)
    if s['over']:
        say('GAME OVER', big, (255, 90, 90), W // 2, H // 2 - 40)
        say(f"Score: {s['score']}   Best: {s['best']}", font, (255, 255, 255), W // 2, H // 2 + 5)
        say('Tap to play again', font, (255, 255, 255), W // 2, H // 2 + 45)


async def main():
    reset()
    running = True
    while running:
        dt = clock.tick(60) / 1000
        s = state

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False
            elif event.type == pygame.MOUSEBUTTONDOWN:
                if s['over']:
                    reset()
                    s = state
                s['finger'] = s['prev'] = event.pos
            elif event.type == pygame.MOUSEMOTION and event.buttons[0]:
```

```
                s['finger'] = event.pos
            elif event.type in (pygame.FINGERDOWN, pygame.FINGERMOTION):
                s['finger'] = (event.x * W, event.y * H)
                if event.type == pygame.FINGERDOWN:
                    s['prev'] = s['finger']
            elif event.type in (pygame.MOUSEBUTTONUP, pygame.FINGERUP):
                s['finger'] = s['prev'] = None

        if not s['over']:
            update(dt)
        s['prev'] = s['finger']

        # trail
        if s['finger']:
            s['trail'].append((s['finger'][0], s['finger'][1], 0.0))
        s['trail'] = [(x, y, t + dt) for x, y, t in s['trail'] if t < 0.15]
        s['flash'] = max(0, s['flash'] - dt)

        draw()
        pygame.display.flip()
        await asyncio.sleep(0)  # needed so the game also runs in the browser

    pygame.quit()


asyncio.run(main())
```