Tag 03 - Quellcode DFJK Rhythm Game

```
WIDTH = 640
HEIGHT = 720
TITLE = 'DFJK'
BPM = 120
BEAT = 60 / BPM
TRAVEL_TIME = 2.0
HIT_WINDOW = 0.16
HIT_Y = 600
NOTE_SPEED = (HIT_Y - 115) / TRAVEL_TIME
LANE_X = [140, 260, 380, 500]
LANE_KEYS = [keys.D, keys.F, keys.J, keys.K]
COLORS = ['lightgreen', 'lightblue', 'plum', 'khaki']

MELODY = [
    ('E5', 1), ('B4', .5), ('C5', .5), ('D5', 1), ('C5', .5), ('B4', .5),
    ('A4', 1), ('A4', .5), ('C5', .5), ('E5', 1), ('D5', .5), ('C5', .5),
    ('B4', 1), (None, .5), ('C5', .5), ('D5', 1), ('E5', 1),
    ('C5', 1), ('A4', 1), ('A4', 1), (None, 1),
    (None, .5), ('D5', .5), (None, .5), ('F5', .5), ('A5', 1), ('G5', .5), ('F5', .5),
    ('E5', 1), (None, .5), ('C5', .5), ('E5', 1), ('D5', .5), ('C5', .5),
    ('B4', 1), ('B4', .5), ('C5', .5), ('D5', 1), ('E5', 1),
    ('C5', 1), ('A4', 1), ('A4', 1), (None, 1),
]

BASS = (
    [('E2', .5), ('E3', .5)] * 4
    + [('A2', .5), ('A3', .5)] * 4
    + [('G#2', .5), ('G#3', .5)] * 2 + [('E2', .5), ('E3', .5)] * 2
    + [('A2', .5), ('A3', .5)] * 3 + [('B1', .5), ('C2', .5)]
    + [('D2', .5), ('D3', .5)] * 4
    + [('C2', .5), ('C3', .5)] * 4
    + [('B1', .5), ('B2', .5)] * 2 + [('E2', .5), ('E3', .5)] * 2
    + [('A1', .5), ('A2', .5)] * 4
)

PITCH_LANE = {'A4': 0, 'B4': 0, 'C5': 1, 'D5': 1,
              'E5': 2, 'F5': 2, 'G5': 3, 'A5': 3}
SONG_LENGTH = sum(duration for pitch, duration in MELODY) * BEAT


def prepare_sounds(song, volume):
    sounds = {}
    for pitch, duration in song:
        if pitch is not None and (pitch, duration) not in sounds:
            sound = tone.create(pitch, duration * BEAT * .9)
            sound.set_volume(volume)
            sounds[pitch, duration] = sound
    return sounds


melody_sounds = prepare_sounds(MELODY, .65)
bass_sounds = prepare_sounds(BASS, .3)

elapsed = -TRAVEL_TIME
notes = []
bass_index, bass_time = (0, 0.0)
hits, misses = (0, 0)
flash_until = [-TRAVEL_TIME] * 4
held = set()

note_time = 0.0
for pitch, duration in MELODY:
    if pitch is not None:
        notes.append({'time': note_time, 'lane': PITCH_LANE[pitch],
                      'sound': melody_sounds[pitch, duration]})
    note_time += duration * BEAT
```

```
def update(dt):
    global elapsed, bass_index, bass_time, misses
    elapsed += dt
    while bass_time <= elapsed:
        pitch, duration = BASS[bass_index]
        if elapsed < bass_time + duration * BEAT:
            bass_sounds[pitch, duration].play()
        bass_time += duration * BEAT
        bass_index = (bass_index + 1) % len(BASS)
    for note in notes:
        while elapsed > note['time'] + HIT_WINDOW:
            note['time'] += SONG_LENGTH
            misses += 1
            if misses == 3:
                print(f'Punktestand: {hits}')
                # raise SystemExit


def on_key_down(key):
    global hits
    if key in held:
        return
    held.add(key)
    if key not in LANE_KEYS:
        return
    lane = LANE_KEYS.index(key)
    flash_until[lane] = elapsed + .1
    for note in notes:
        if (note['lane'] == lane
                and abs(note['time'] - elapsed) <= HIT_WINDOW):
            note['time'] += SONG_LENGTH
            note['sound'].play()
            hits += 1
            return


def on_key_up(key):
    held.discard(key)


def draw():
    screen.fill('gray10')
    screen.draw.text('DFJK', center=(WIDTH // 2, 35), fontsize=44, color='white')
    screen.draw.text(f'Treffer: {hits}   Verpasst: {misses}',
                     center=(WIDTH // 2, 75), fontsize=24, color='white')
    for lane, x in enumerate(LANE_X):
        screen.draw.filled_rect(Rect(x - 50, 105, 100, 535), 'gray20')
        target = Rect(x - 43, HIT_Y - 12, 86, 24)
        if elapsed < flash_until[lane]:
            screen.draw.filled_rect(target, COLORS[lane])
        else:
            screen.draw.rect(target, COLORS[lane])
        screen.draw.text('DFJK'[lane], center=(x, 665), fontsize=36, color=COLORS[lane])
    # screen.draw.line((80, HIT_Y), (560, HIT_Y), 'white')
    for note in notes:
        y = HIT_Y - (note['time'] - elapsed) * NOTE_SPEED
        if 105 <= y <= 635:
            lane = note['lane']
            screen.draw.filled_rect(Rect(LANE_X[lane] - 38, y - 9, 76, 18), COLORS[lane])
```