Chart Builder Structure

このドキュメントは、Kairoscopeの `chart_builder.py` における全体構造を設計し、HDチャート解析の各処理モジュールと連携する中核エンジンを定義するものです。

📌主な役割

- 惑星位置データの取得（astro_position.py）
- 黄経からゲート・ラインの算出（gate_mapper.py）
- ゲート情報の参照（gate-definitions.json）
- アクティブチャネルとセンターの解析（channel_center.py）
- プロファイルおよび内的権威の導出（profile_logic.py / authority_logic.py）
- 最終的なチャート構造の出力（辞書/JSON形式）

使用モジュール：修正済コード集

🔗core/gate_mapper.py

```
def longitude_to_gate_line(longitude):
    longitude = longitude % 360
    gate_size = 360 / 64
    line_size = gate_size / 6
    gate_number = int(longitude / gate_size) + 1
    within_gate_deg = longitude % gate_size
    line_number = int(within_gate_deg / line_size) + 1
    return gate_number, line_number

def get_gate_and_line(ecliptic_longitude):
    gate, line = longitude_to_gate_line(ecliptic_longitude)
    return {"gate": gate, "line": line}
```

🔗core/channel_center.py

```
def get_active_channels_and_centers(gates, channel_defs):
    active_gate_nums = {g["gate"] for g in gates.values()}
    active_channels = []
    defined_centers = set()

    for ch in channel_defs:
        gate_a, gate_b = [int(g) for g in ch["gates"]]
        if gate_a in active_gate_nums and gate_b in active_gate_nums:
            active_channels.append(ch["channel"])
```

```
            defined_centers.update(ch["centers"])

    return active_channels, list(defined_centers)
```

🔗**core/profile_logic.py**

```
def calculate_profile(planet_gates):
    sun = planet_gates.get("Sun", {})
    earth = planet_gates.get("Earth", {})
    personality_line = sun.get("line")
    design_line = earth.get("line")

    if personality_line and design_line:
        return f"{personality_line}/{design_line}"
    return "Unknown"
```

🔗**core/authority_logic.py**

```
def determine_authority(gates, defined_centers):
    if "Solar Plexus" in defined_centers:
        return "Emotional"
    elif "Sacral" in defined_centers:
        return "Sacral"
    elif "Spleen" in defined_centers:
        return "Splenic"
    elif "Ego" in defined_centers:
        return "Ego Projected"
    elif "G" in defined_centers:
        return "Self Projected"
    elif type_hint == "Projector":
        return "Mental Projected"
    elif type_hint == "Reflector":
        return "Lunar"
    else:
        return "Unknown"
```

🔗**core/astro/astro_position.py**

```
from skyfield.api import load, Topos
from datetime import datetime
from pytz import timezone

def get_planet_positions(birth_data):
    ts = load.timescale()
```

```
    dt = datetime.strptime(f"{birth_data['date']} {birth_data['time']}", "%Y-%m-
%d %H:%M")
    dt = timezone("Asia/Tokyo").localize(dt)
    t = ts.from_datetime(dt)

    planets = load('de421.bsp')
    earth = planets['earth']
    loc = earth + Topos(latitude_degrees=40.8246, longitude_degrees=140.7400)

    planet_names = ["Sun", "Moon", "Mercury", "Venus", "Mars", "Jupiter",
"Saturn", "Uranus", "Neptune", "Pluto"]
    planet_ids = {
        "Sun": "sun",
        "Moon": "moon",
        "Mercury": "mercury",
        "Venus": "venus",
        "Mars": "mars",
        "Jupiter": "jupiter barycenter",
        "Saturn": "saturn barycenter",
        "Uranus": "uranus barycenter",
        "Neptune": "neptune barycenter",
        "Pluto": "pluto barycenter",
    }

    positions = {}
    for name in planet_names:
        target = planet_ids[name]
        planet = planets[target]
        astrometric = loc.at(t).observe(planet)
        lon, _, _ = astrometric.ecliptic_latlon()
        positions[name] = lon.degrees

    return positions
```

🔗**core/chart_builder.py**

```
from core.astro.astro_position import get_planet_positions
from core.gate_mapper import get_gate_and_line
from core.channel_center import get_active_channels_and_centers
from core.profile_logic import calculate_profile
from core.authority_logic import determine_authority
from core.definitions.loader import load_gate_definitions,
load_channel_definitions

def build_chart(birth_data):
    positions = get_planet_positions(birth_data)
```

```
    positions["Earth"] = (positions["Sun"] + 180) % 360

    raw_gates = {planet: get_gate_and_line(lon) for planet, lon in
positions.items()}

    gate_defs = load_gate_definitions()
    channel_defs = load_channel_definitions()
    gate_dict = {g["gate"]: g for g in gate_defs}
    channel_dict = {c["channel"]: c for c in channel_defs}

    gates = {
        planet: {
            "gate": g["gate"],
            "line": g["line"],
            **gate_dict.get(g["gate"], {})
        }
        for planet, g in raw_gates.items()
    }

    active_channels, defined_centers =
get_active_channels_and_centers(raw_gates, channel_defs)
    enriched_channels = [channel_dict[c] for c in active_channels if c in
channel_dict]

    profile = calculate_profile(raw_gates)
    authority = determine_authority(raw_gates, defined_centers)

    chart = {
        "planet_positions": positions,
        "gates": gates,
        "active_channels": enriched_channels,
        "defined_centers": defined_centers,
        "profile": profile,
        "authority": authority
    }
    return chart

if __name__ == "__main__":
    birth_data = {
        "date": "1983-05-01",
        "time": "14:35",
        "location": "Aomori, Japan"
    }

    chart = build_chart(birth_data)

    import json
    print(json.dumps(chart, indent=2, ensure_ascii=False))
```

テスト方法：

```
cd /Users/takeoyamada/Library/Mobile\ Documents/iCloud\~md\~obsidian/Documents/codex-collective-archive/common-system/01-system/chronogram-system/chronogram-kairoscope

PYTHONPATH=. python3 core/chart_builder.py
```

📦Chronogram統合用：Kairoscopeチャート構造サンプルセット準備予定

- `samples/sample_chart_full.json`
- `samples/sample_chart_variants.json`（time sweep 含む）
- `samples/sample_structure_schema.json`（Chronogram連携用定義）

以上が修正済の初期実行可能Kairoscopeエンジン構成。Chronogramへの連携と多時間帯対応も見据えた設計へ🔥