main.py

```
"""Main game entry point: title screen, character creation, game loop."""
import sys
import os

sys.path.insert(0, os.path.dirname(__file__))

from data_types import *
from character import create_player_character, JOB_BASE_STATS
from game_state import GameState
from skills_db import SKILL_DB, JOB_SKILL_POOL


def clear():
    os.system('clear' if os.name == 'posix' else 'cls')


def print_title():
    clear()
    print("""
╔══════════════════════════════════════════════════════════════╗
║                                                              ║
║          ✦  CHRONICLES OF THE ETERNAL TURN  ✦                ║
║                  A JRPG CLI ADVENTURE                        ║
║                                                              ║
╚══════════════════════════════════════════════════════════════╝
""")


def create_character() -> tuple:
    clear()
    print("╔══════════════════════════════════════════════════════════════╗")
    print("║                  CHARACTER CREATION                          ║")
    print("╚══════════════════════════════════════════════════════════════╝")

    while True:
```

```
    name = input("\n  Enter your hero's name: ").strip()
    if name:
        break
    print("  Please enter a name.")

weapons = list(WeaponType)
weapon_descs = {
    WeaponType.SWORD:   "Balanced. Warriors, Knights, Paladins",
    WeaponType.SPEAR:   "Long reach. Spearman, Dragoon",
    WeaponType.AXE:     "High power. Berserker, Monk",
    WeaponType.DAGGER:  "Fast & precise. Assassin",
    WeaponType.BOW:     "Ranged. Ranger, Hunter",
    WeaponType.STAFF:   "Magical focus. All Mages and Healers",
}
print("\n  ── Choose Your Weapon ──")
for i, w in enumerate(weapons):
    print(f"    [{i+1}] {w.value:10s} - {weapon_descs[w]}")
while True:
    try:
        wi = int(input("  > ")) - 1
        if 0 <= wi < len(weapons): break
    except ValueError: pass
    print("  Invalid.")
weapon = weapons[wi]

elements = [e for e in Element if e != Element.NONE]
elem_descs = {
    Element.FIRE:      "Offensive burn effects",
    Element.ICE:       "Freeze and slow foes",
    Element.LIGHTNING: "Chain damage, high speed",
    Element.WIND:      "Evasive, speed-based",
    Element.EARTH:     "Defense-break, sturdy",
    Element.LIGHT:     "Holy power, vs Undead/Dark",
    Element.DARK:      "Debuff and drain",
}
print("\n  ── Choose Your Element ──")
for i, e in enumerate(elements):
    print(f"    [{i+1}] {e.value:10s} - {elem_descs[e]}")
while True:
    try:
```

```
        ei = int(input("  > ")) - 1
        if 0 <= ei < len(elements): break
    except ValueError: pass
    print("  Invalid.")
element = elements[ei]

all_jobs = list(JobClass)
job_weapon_affinity = {
    WeaponType.SWORD:  [JobClass.WARRIOR, JobClass.KNIGHT, JobClass.PALADIN,
                        JobClass.ASSASSIN, JobClass.BERSERKER, JobClass.DRAGOON,
                        JobClass.DARK_MAGE, JobClass.LIGHT_MAGE],
    WeaponType.SPEAR:  [JobClass.SPEARMAN, JobClass.DRAGOON, JobClass.WARRIOR,
                        JobClass.KNIGHT, JobClass.RANGER, JobClass.HUNTER],
    WeaponType.AXE:    [JobClass.BERSERKER, JobClass.WARRIOR, JobClass.MONK,
                        JobClass.EARTH_MAGE, JobClass.DRAGOON, JobClass.KNIGHT],
    WeaponType.DAGGER: [JobClass.ASSASSIN, JobClass.RANGER, JobClass.HUNTER,
                        JobClass.WIND_MAGE, JobClass.DARK_MAGE, JobClass.MONK],
    WeaponType.BOW:    [JobClass.RANGER, JobClass.HUNTER, JobClass.ASSASSIN,
                        JobClass.STORM_MAGE, JobClass.WIND_MAGE, JobClass.LIGHT_MAGE],
    WeaponType.STAFF:  [JobClass.CLERIC, JobClass.PRIEST, JobClass.FIRE_MAGE,
                        JobClass.ICE_MAGE, JobClass.STORM_MAGE, JobClass.WIND_MAGE,
                        JobClass.EARTH_MAGE, JobClass.DARK_MAGE, JobClass.LIGHT_MAGE,
                        JobClass.ARCANE_SAGE, JobClass.PALADIN],
}

suggested = job_weapon_affinity.get(weapon, all_jobs)
job_info = {
    JobClass.WARRIOR:    "Melee fighter, high PATK, balanced",
    JobClass.KNIGHT:     "Tank, high HP/DEF, Shield Bash",
    JobClass.PALADIN:    "Holy warrior, heal & attack",
    JobClass.BERSERKER:  "Rage fighter, highest PATK, low DEF",
    JobClass.ASSASSIN:   "Stealth, high CRIT/EVA, poison",
    JobClass.RANGER:     "Bow specialist, multi-hit, area",
    JobClass.HUNTER:     "Traps, dragon slayer, beast lore",
    JobClass.SPEARMAN:   "Spear master, sweep attacks",
    JobClass.DRAGOON:    "Dragon knight, jump attacks",
    JobClass.MONK:       "Unarmed master, combo strikes",
    JobClass.CLERIC:     "Healer & smiter, Light magic",
    JobClass.PRIEST:     "Pure healer, mass heals, revival",
    JobClass.FIRE_MAGE:  "Fire magic, burn effects",
```

```
    JobClass.ICE_MAGE:   "Ice magic, freeze/slow",
    JobClass.STORM_MAGE: "Lightning magic, chain effects",
    JobClass.WIND_MAGE:  "Wind magic, speed buffs",
    JobClass.EARTH_MAGE: "Earth magic, high power AOE",
    JobClass.DARK_MAGE:  "Dark magic, debuffs, drain",
    JobClass.LIGHT_MAGE: "Light magic, barriers, blind",
    JobClass.ARCANE_SAGE:"All-element magic, time manipulation",
}

print(f"\n  ── Choose Your Job (Weapon: {weapon.value}) ──")
print(f"  Recommended jobs shown. Enter A for all jobs.")
print()
for i, job in enumerate(suggested):
    print(f"    [{i+1:2d}] {job.value:15s} - {job_info.get(job,'')}")
print(f"    [ A] Show ALL {len(all_jobs)} jobs")

job = None
while True:
    raw = input("  > ").strip().upper()
    if raw == "A":
        print("\n  ALL JOBS:")
        for i, j in enumerate(all_jobs):
            print(f"    [{i+1:2d}] {j.value:15s} - {job_info.get(j,'')}")
        while True:
            try:
                ji = int(input("  > ")) - 1
                if 0 <= ji < len(all_jobs):
                    job = all_jobs[ji]
                    break
            except ValueError: pass
            print("  Invalid.")
        break
    else:
        try:
            idx = int(raw) - 1
            if 0 <= idx < len(suggested):
                job = suggested[idx]
                break
        except ValueError: pass
        print("  Invalid choice.")
```

```
    # Skill preview
    print(f"\n  ── Skill Preview for {job.value} ──")
    pool = JOB_SKILL_POOL[job]
    basics = [sid for sid in pool if SKILL_DB[sid].tier == SkillTier.BASIC][:4]
    inters = [sid for sid in pool if SKILL_DB[sid].tier == SkillTier.INTERMEDIATE][:4]
    ultims = [sid for sid in pool if SKILL_DB[sid].tier == SkillTier.ULTIMATE][:2]
    print("  Basic (start with 2):")
    for sid in basics:
        sk = SKILL_DB[sid]
        print(f"    • {sk.name:20s} MP:{sk.mp_cost:3d} PWR:{sk.power} - {sk.description}")
    print("  Intermediate (unlock lv5/10):")
    for sid in inters:
        sk = SKILL_DB[sid]
        print(f"    • {sk.name:20s} MP:{sk.mp_cost:3d} PWR:{sk.power} - {sk.description}")
    print("  Ultimate (unlock lv20):")
    for sid in ultims:
        sk = SKILL_DB[sid]
        print(f"    • {sk.name:20s} MP:{sk.mp_cost:3d} PWR:{sk.power} - {sk.description}")

    # Summary
    stats = JOB_BASE_STATS[job]
    print(f"\n  ── Summary ──")
    print(f"  Name: {name}  |  Job: {job.value}  |  Weapon: {weapon.value}  |  Element:
{element.value}")
    print(f"  HP:{stats.hp} MP:{stats.mp} PATK:{stats.patk} MATK:{stats.matk} "
          f"PDEF:{stats.pdef} MDEF:{stats.mdef} SPD:{stats.spd}")

    confirm = input("\n  Confirm? (Y/N) > ").strip().upper()
    if confirm != "Y":
        return create_character()

    return name, job, weapon, element


def main():
    print_title()
    print("""
  Welcome to Chronicles of the Eternal Turn!
```

```
  OBJECTIVE: Survive as many battles as you can.
             Collect companions, level up, grow powerful.

  COMBAT TIPS:
  • Attack enemies' Weaknesses to reduce their Shield Points
  • When Shield = 0, enemy is BROKEN (50% more damage, can't act)
  • Use Defend to reduce incoming damage and gain turn priority
  • If a party member is KO'd, revive within 3 turns or they DIE
  • If YOUR character dies → GAME OVER

  COMPANION SYSTEM:
  • Win battles to earn companions (3★/4★/5★)
  • Party maximum: 4 members
  • Companions can permanently die in battle
""")
    input("  [Press Enter to begin]")

    name, job, weapon, element = create_character()
    clear()

    gs = GameState()
    player = create_player_character(name, job, weapon, element)
    gs.player = player
    gs.party = [player]
    gs.all_party_members = [player]

    print(f"\n  ✦ {player.name} the {player.job.value} sets forth!")
    print(f"  Starting skills: " +
          ", ".join(SKILL_DB[s].name for s in player.unlocked_skills if s in SKILL_DB))
    input("\n  [Press Enter to begin your adventure]")

    # Main game loop
    while not gs.game_over:
        print(f"\n{'='*65}")
        print(f"  ✦ BATTLE {gs.battle_number + 1}")

        if gs.battle_number > 0:
            cont = gs.between_battles_menu()
            if not cont:
                print("\n  Thanks for playing! Farewell, hero.")
```

```
                print(f"  Final record: {gs.battle_number} battles | {gs.player.name}
Lv{gs.player.level}")
                break

        alive = [ch for ch in gs.party if not ch.is_dead]
        if not alive:
            gs.game_over = True
            break

        survived = gs.run_battle()

        if not survived or gs.game_over:
            gs.show_game_over_screen()
            break

    if not gs.game_over and gs.battle_number > 0:
        print(f"\n  Journey ended. {gs.battle_number} battles completed.")
        print(f"  {gs.player.name} reached Level {gs.player.level}. Well done!")


if __name__ == "__main__":
    main()
```

Revision #1
Created 2026-03-18 16:19:57 UTC by Samuel Lee
Updated 2026-03-18 16:20:10 UTC by Samuel Lee