# Pathfinder 1st Edition Battle Simulator

A comprehensive, rule-compliant battle simulator for Pathfinder 1st Edition following the detailed specification in prompt-pathfinder.md.

## 🎯 Current Implementation Status

### ✅ Completed Features (Parts 1-3)

**Part 1: Foundational Elements** - ✅ Complete Combatant data structure with all Pathfinder 1e stats - ✅ Persistent monster database with JSON storage - ✅ Monster template save/load functionality - ✅ Player character creation system

**Part 2: Core Combat Mechanics** - ✅ Initiative system with proper tie-breaking - ✅ Surprise round mechanics - ✅ Round and turn management - ✅ Flat-footed status tracking - ✅ Combat logging system

**Part 3: Actions & Basic Attacks** - ✅ Action economy (Standard, Move, Full-Round, Swift, Free actions) - ✅ Melee and ranged attack mechanics - ✅ Attack roll calculations with all modifiers - ✅ Damage calculation and critical hit system - ✅ Automatic hit/miss determination - ✅ Full-attack actions with iterative attacks

### 📋 Implemented Core Features

* **Complete Combatant System**: HP, AC, BAB, ability scores, saves, skills, feats
* **Weapon/Attack System**: Damage dice, critical ranges, enhancement bonuses
* **Combat Resolution**: Attack rolls, damage, DR, resistances, conditions
* **Database Management**: Persistent monster storage with JSON files
* **Initiative & Rounds**: Proper turn order, surprise rounds, flat-footed tracking
* **Action Types**: Standard attacks, full attacks, movement actions
* **Combat Logging**: Detailed event tracking and results display

## 📁 File Structure

/workspace/  
├── code/  
│ ├── pathfinder\_simulator.py # Core simulator engine (Parts 1-3)  
│ ├── pathfinder\_cli.py # Interactive command-line interface  
│ ├── demo\_combat.py # Comprehensive demonstration script  
│ └── monster\_data/ # JSON monster database  
│ ├── orc\_warrior.json  
│ ├── goblin.json  
│ ├── skeleton.json  
│ └── ...  
├── user\_input\_files/  
│ └── prompt-pathfinder.md # Complete 12-part specification  
└── README.md # This file

## 🚀 Quick Start

### Run the Interactive Demo

cd /workspace/code  
python demo\_combat.py

This demonstrates: - Database operations (save/load monsters) - Combat mechanics breakdown - Full 4-character combat encounter - All implemented features in action

### Use the Interactive CLI

cd /workspace/code  
python pathfinder\_cli.py

Interactive features: - Browse monster database - Create player characters - Set up combat encounters - Run turn-based combat with player input

### Use as a Library

from pathfinder\_simulator import Combatant, CombatEngine, MonsterDatabase  
  
# Load monsters from database  
db = MonsterDatabase()  
orc = db.load\_monster("Orc Warrior")  
  
# Create player character  
fighter = Combatant("Hero", is\_pc=True)  
# ... configure stats ...  
  
# Run combat  
combat = CombatEngine()  
combat.add\_combatant(orc)  
combat.add\_combatant(fighter)  
combat.start\_combat()

## 📊 Example Combat Output

=== COMBAT BEGINS ===  
=== Rolling Initiative ===  
Shadowstep: 14 + 4 = 18  
Goblin Archer: 13 + 3 = 16  
Orc Barbarian: 8 + 1 = 9  
Sir Roderick: 6 + 1 = 7  
  
=== Final Initiative Order ===  
1. Shadowstep: 18  
2. Goblin Archer: 16  
3. Orc Barbarian: 9  
4. Sir Roderick: 7  
  
=== ROUND 1 ===  
Shadowstep attacks Goblin Archer with Rapier  
 Attack roll: 9 + 5 = 14 vs AC 13  
 HIT!  
 Damage roll: 3  
 Goblin Archer takes 3 damage  
 Goblin Archer HP: 3/6

## 🎮 Sample Monsters Included

The system comes with pre-configured monsters: - **Orc Warrior**: Basic humanoid fighter with falchion - **Orc Barbarian**: Stronger orc with greataxe and higher stats - **Goblin**: Small archer with shortbow - **Goblin Archer**: Enhanced goblin with better dexterity - **Skeleton**: Undead with damage reduction

## ⚡ Core Classes & Components

### Combatant

Complete character/monster representation with: - All six ability scores with modifiers - Hit points, armor class, and saves - Base attack bonus and weapon attacks - Skills, feats, and special abilities - Size, type, alignment, and conditions

### CombatEngine

Manages combat flow: - Initiative rolling and turn order - Round progression and timing - Combat state tracking - Victory condition detection

### MonsterDatabase

Persistent storage system: - JSON-based monster templates - Save/load functionality - Monster listing and management

### ActionHandler

Action economy management: - Standard, move, and full-round actions - Attack resolution - Action validation

## 🔮 Future Expansion (Parts 4-12)

The current implementation provides a solid foundation for the remaining specification parts:

**Parts 4-6**: Movement, positioning, attacks of opportunity, conditions, death/dying **Parts 7-9**: Combat maneuvers, spellcasting, mythic rules, advanced UI **Parts 10-12**: Advanced Player’s Guide, universal monster rules, GM utilities

The modular design supports easy extension with additional rules and features.

## 🛠 Technical Details

* **Language**: Python 3.x
* **Data Storage**: JSON files for persistent monster templates
* **Architecture**: Modular classes with clear separation of concerns
* **Rules Compliance**: Strict adherence to Pathfinder 1e Core Rules
* **Extensibility**: Designed for easy addition of new rules and features

## 📝 Usage Examples

### Creating a Custom Monster

# Create a new monster  
dragon = Combatant("Young Red Dragon", is\_pc=False)  
dragon.max\_hp = 78  
dragon.ability\_scores.strength = 21  
dragon.ability\_scores.dexterity = 10  
# ... configure stats ...  
  
# Add breath weapon attack  
breath = Attack("Fire Breath", "6d10", "20", "x2", DamageType.FIRE)  
dragon.attacks.append(breath)  
  
# Save to database  
db = MonsterDatabase()  
db.save\_monster(dragon)

### Running Combat Programmatically

# Set up encounter  
combat = CombatEngine()  
combat.add\_combatant(hero)  
combat.add\_combatant(monster)  
  
# Start combat  
combat.start\_combat()  
  
# Execute turns  
action\_handler = ActionHandler(combat)  
while combat.combat\_active:  
 current = combat.get\_current\_combatant()  
 # ... handle turn logic ...  
 combat.advance\_turn()

**Author**: MiniMax Agent  
**Version**: 1.0 (Parts 1-3 Complete)  
**Specification**: Based on comprehensive 12-part Pathfinder 1e plan