Fantasy Card Battle Game
OOP Final Project

Arcane Duel

By: Bernardo Mejia & Marshall Mangum

Overview

Tools used:

- Python + Pygame

Purpose:

- Turn based card game where players battle using attack, defense, and support cards.
- Create a fun game

Key Features:

- Different card types with unique effects
- AI opponent with selectable difficulty levels
- Mana and health management system
- Card animations and sound effects for gameplay feedback

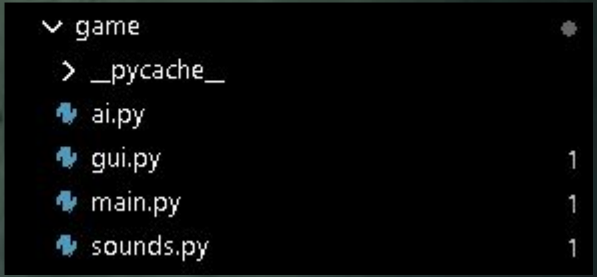
Task Split

Bernardo:

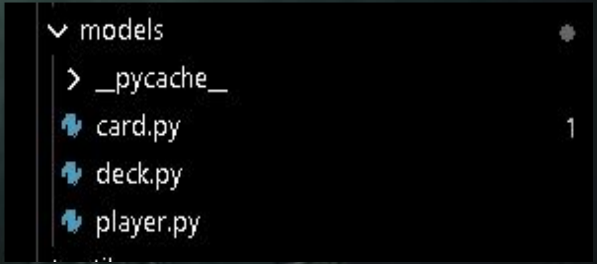
- Game logic & turn system
- Card click handling
- AI turn logic
- Drawing cards, handling mana, health, and effects

Marshall:

- Card classes & Deck creation
- Player class & status effect management
- Attack/Defense/Support card behavior
- All assets (images, sounds, backgrounds)



```
game
  > __pycache__
  ai.py
  gui.py        1
  main.py       1
  sounds.py     1
```



```
models
  > __pycache__
  card.py       1
  deck.py
  player.py
```

AI System

- Enemy AI programmed in ai.py
- Evaluates hand & mana
- Chooses best card or combination

Decision making logic

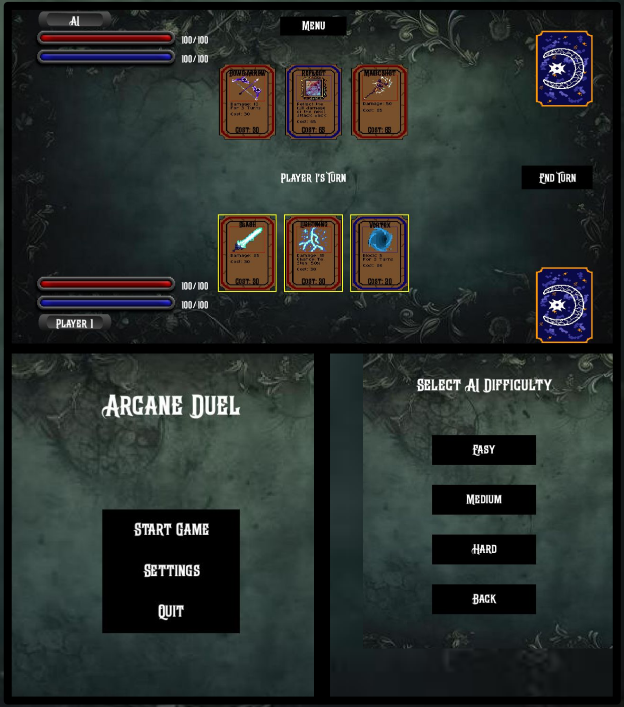
- The AI checks its hand for playable cards based on current mana
- It prioritizes attacks, defenses, or support cards depending on the situation

Simple strategy examples

- If AI health is low ⤐ prioritize healing/support
- If player block is high ⤐ use DOT or unblockable attacks
- If player has attack boost ⤐ prioritize defensive cards

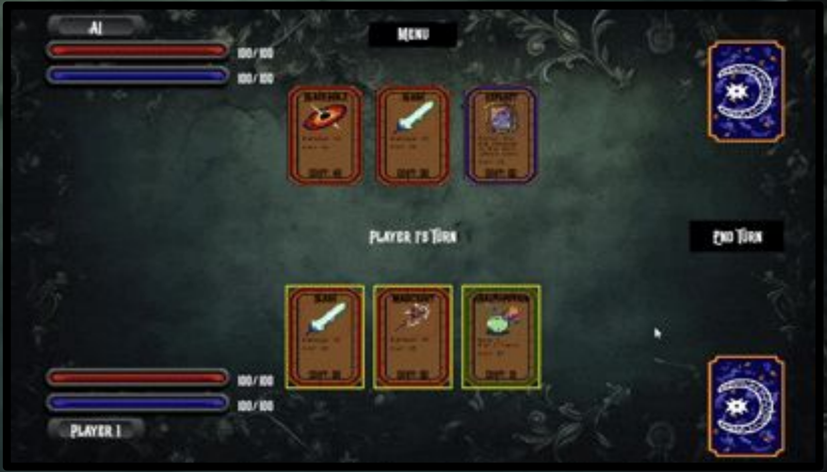
GUI System

- I created the graphics and rendering in gui.py
- Handles card drawing, hover animations, and visual feedback.
- Displays player stats: health, mana, block
- Manages layout for hands and board visuals
- Connects OOP logic to the visuals
- Uses Pygame for rendering, animations, and input handling
- Ensures smooth user interaction, like clicking and hovering over cards

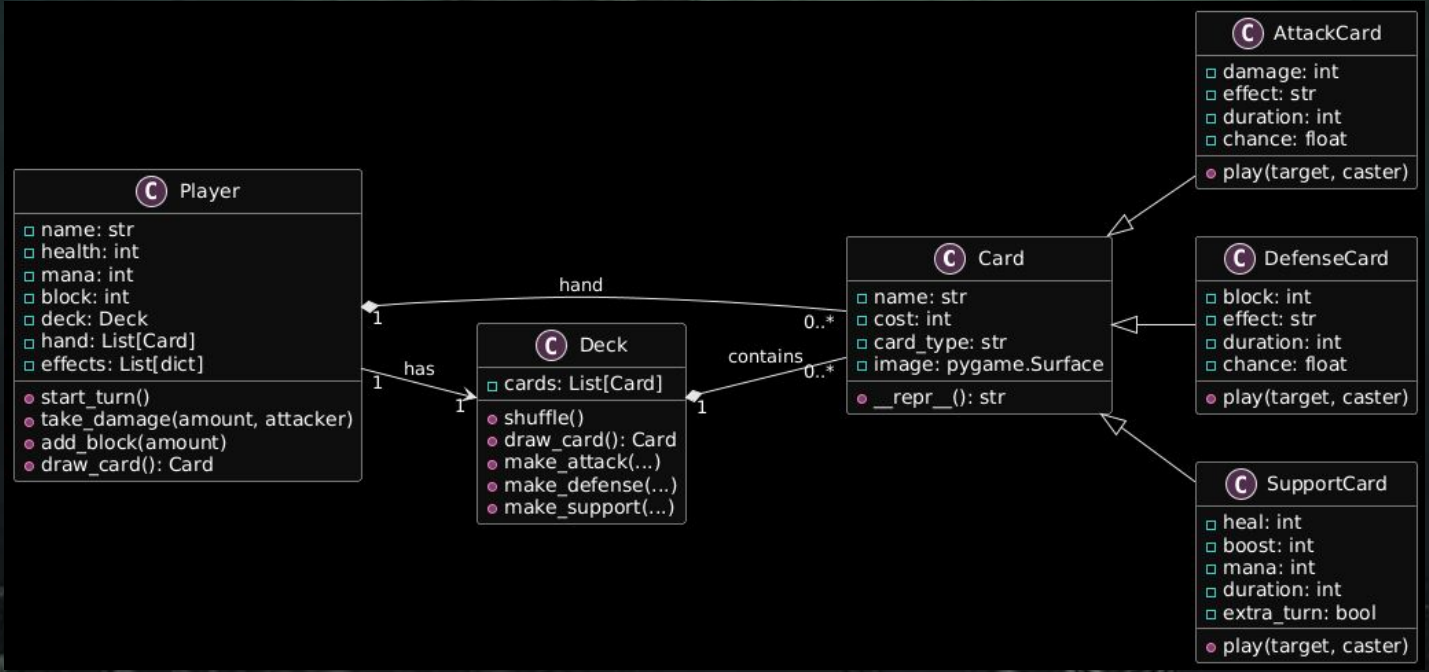


Game Loop

- In main.py, I put everything together into a working game
- Handles Pygame initialization for graphics, sounds, and input
- Manages turn order between player and AI
- Detects player input (like selecting and playing cards)
- Calls the AI logic for the opponent's turn
- Checks win/lose conditions each turn
- Manages animations and sounds triggered by actions



UML Diagram



Player
name: str
health: int
mana: int
block: int
deck: Deck
hand: List[Card]
effects: List[dict]
start_turn()
take_damage(amount, attacker)
add_block(amount)
draw_card(): Card
hand
1
0..*
has
1
1
Deck
cards: List[Card]
shuffle()
draw_card(): Card
make_attack(...)
make_defense(...)
make_support(...)
contains
1
0..*
Card
name: str
cost: int
card_type: str
image: pygame.Surface
__repr__(): str
AttackCard
damage: int
effect: str
duration: int
chance: float
play(target, caster)
DefenseCard
block: int
effect: str
duration: int
chance: float
play(target, caster)
SupportCard
heal: int
boost: int
mana: int
duration: int
extra_turn: bool
play(target, caster)

Card System

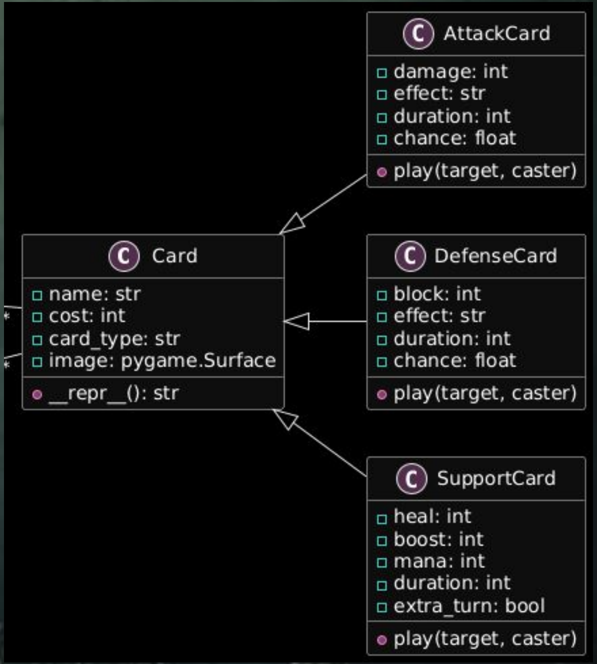
Base Card class:

- Attributes: name, cost, card_type, image
- Method: __repr__()

Subclasses:

- AttackCard: damage, effect, duration, chance ⤜ .play() applies damage/effects
- DefenseCard: block, effect, duration, chance ⤜ .play() applies defensive effects
- SupportCard: heal, boost, mana, duration, extra_turn ⤜ .play() applies buffs or healing

Demonstrates inheritance and polymorphism



Deck Class

Deck contains a list of Card objects

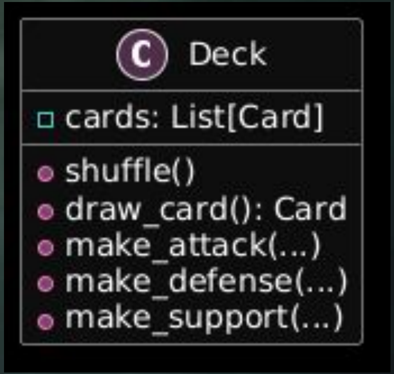
Loads predefined templates for attack, defense, support

Creates card instances via make_attack(), make_defense(), make_support()

Methods:

- .shuffle()
- .draw_card()

Shows composition (Player ⤜ Deck ⤜ Cards) and encapsulation



Player Class

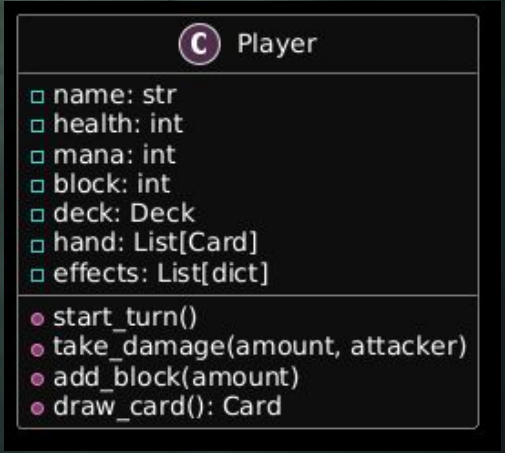
Manages player stats: health, mana, block

Status effects: DOT, stun, dodge, reflect, regen, boosts, extra turns

Methods:

- .start_turn(): applies effects, updates block, manages extra turns
- .take_damage(amount): calculates block, dodge, reflect, forcefield
- .add_block(amount): increases block
- .draw_card(): draws from deck

Demonstrates encapsulation (all player logic is self contained)



OOP Concepts

Classes & Objects	Card, Deck, Player, AI, GUI
Inheritance	AttackCard, DefenseCard, SupportCard extend Card
Encapsulation	Player manages its own stats, effects, and turn logic
Composition	Player ↠ Deck ↠ Cards Deck contains card objects

Design Patterns

Strategy Pattern

- Each card type (Attack, Defense, Support) defines its own play behavior
- allows the game to add new card types without modifying existing code
- Makes card actions flexible and easy to extend

Factory Method Pattern

- make_attack(), make_defense(), make_support() create cards from templates
- Centralizes object creation so decks are built consistently and safely
- Makes it easy to tweak or balance cards without rewriting logic

State Pattern

- Player status effects (stun, poison, regen, block) modify behavior based on current state.
- Effects persist across turns and update automatically
- Keeps turn logic clean by letting each state control its own rules

Thank you!