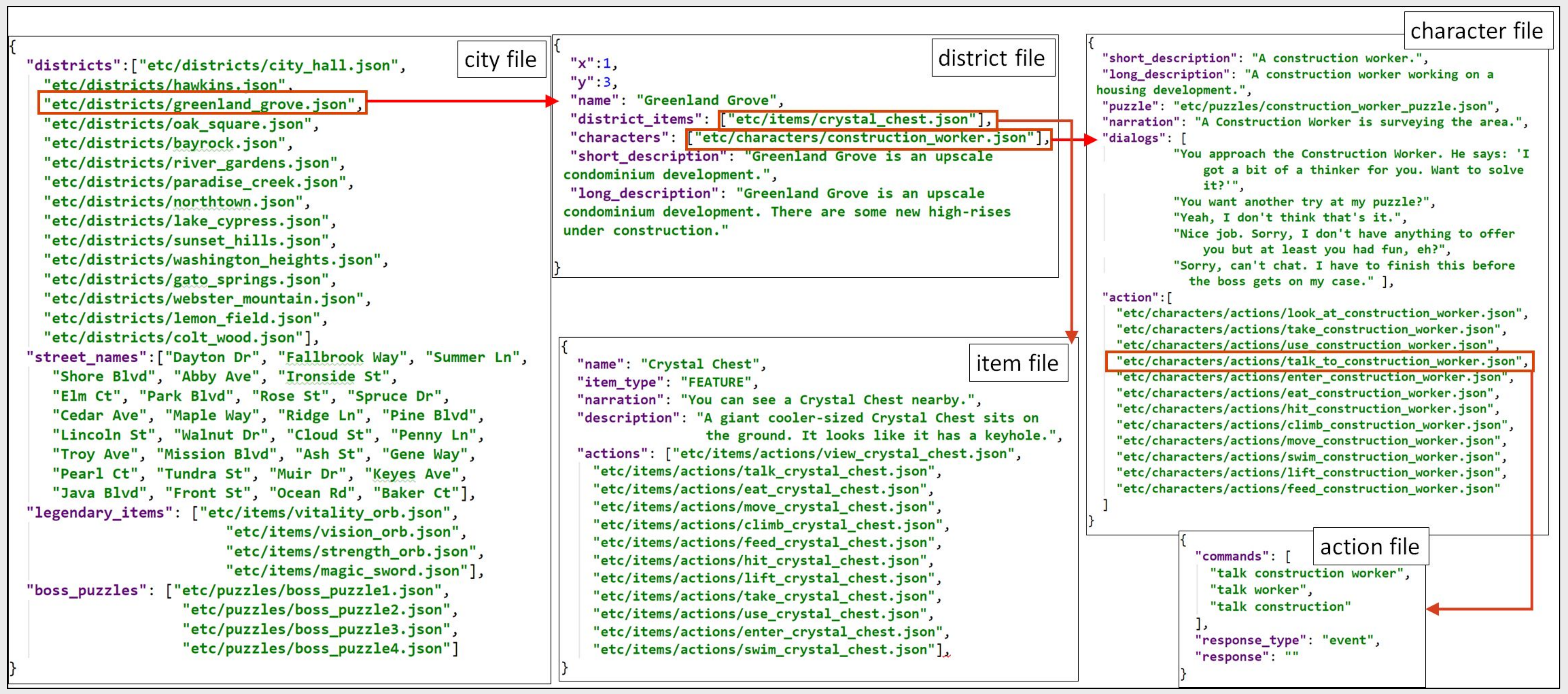
NEW SAN DIEGO SAGA: A Text-Based Adventure-Puzzle Game

By: Derrick Li, Hyoung Sik Won, William Fargo

KEY POINTS

- For our Capstone Project, our team built a text-based adventure-puzzle game using Python 3 programming language. It is playable via the command-line.
- The natural language parsers exists as its own layer and sits on top of the large set of core commands. These two, in tandem, allows the user to enter an expanded set of phrases, including variety of nouns, verbs, and prepositions.
- Nearly all of the game's data is loaded from JSON files. This enables readily accessible and maintainability of game content.
- Object-oriented programming practices were adopted to make it easy to modify and unit test game source code.
- A 'gamestate' class keeps track of the user's progress in the game. A user's progress is saved to and loaded from non human-readable files. This prevents cheating and invalid game states.
- There are 15 different locations in the game for the user to explore. Each with unique "features", i.e. items and characters, to interact with. Characters offer puzzles for the user to solve. Items can be interacted with other items as well.

Oregon State University



```
{
  "districts":["etc/districts/city_hall.json",
    "etc/districts/hawkins.json",
    "etc/districts/greenland_grove.json",
    "etc/districts/oak_square.json",
    "etc/districts/bayrock.json",
    "etc/districts/river_gardens.json",
    "etc/districts/paradise_creek.json",
    "etc/districts/northtown.json",
    "etc/districts/lake_cypress.json",
    "etc/districts/sunset_hills.json",
    "etc/districts/washington_heights.json",
    "etc/districts/gato_springs.json",
    "etc/districts/webster_mountain.json",
    "etc/districts/lemon_field.json",
    "etc/districts/colt_wood.json"],
  "street_names":["Dayton Dr", "Fallbrook Way", "Summer Ln",
     "Shore Blvd", "Abby Ave", "Ironside St",
     "Elm Ct", "Park Blvd", "Rose St", "Spruce Dr",
     "Cedar Ave", "Maple Way", "Ridge Ln", "Pine Blvd",
     "Lincoln St", "Walnut Dr", "Cloud St", "Penny Ln",
     "Troy Ave", "Mission Blvd", "Ash St", "Gene Way",
     "Pearl Ct", "Tundra St", "Muir Dr", "Keyes Ave",
     "Java Blvd", "Front St", "Ocean Rd", "Baker Ct"],
  "legendary_items": ["etc/items/vitality_orb.json",
                      "etc/items/vision_orb.json",
                      "etc/items/strength_orb.json",
                      "etc/items/magic_sword.json"],
  "boss_puzzles": ["etc/puzzles/boss_puzzle1.json",
                   "etc/puzzles/boss_puzzle2.json",
                   "etc/puzzles/boss_puzzle3.json",
                   "etc/puzzles/boss_puzzle4.json"]
}
```

```
{
  "x":1,
  "y":3,
  "name": "Greenland Grove",
  "district_items": ["etc/items/crystal_chest.json"],
  "characters": ["etc/characters/construction_worker.json"],
  "short_description": "Greenland Grove is an upscale
condominium development.",
  "long_description": "Greenland Grove is an upscale
condominium development. There are some new high-rises
under construction."

}
```

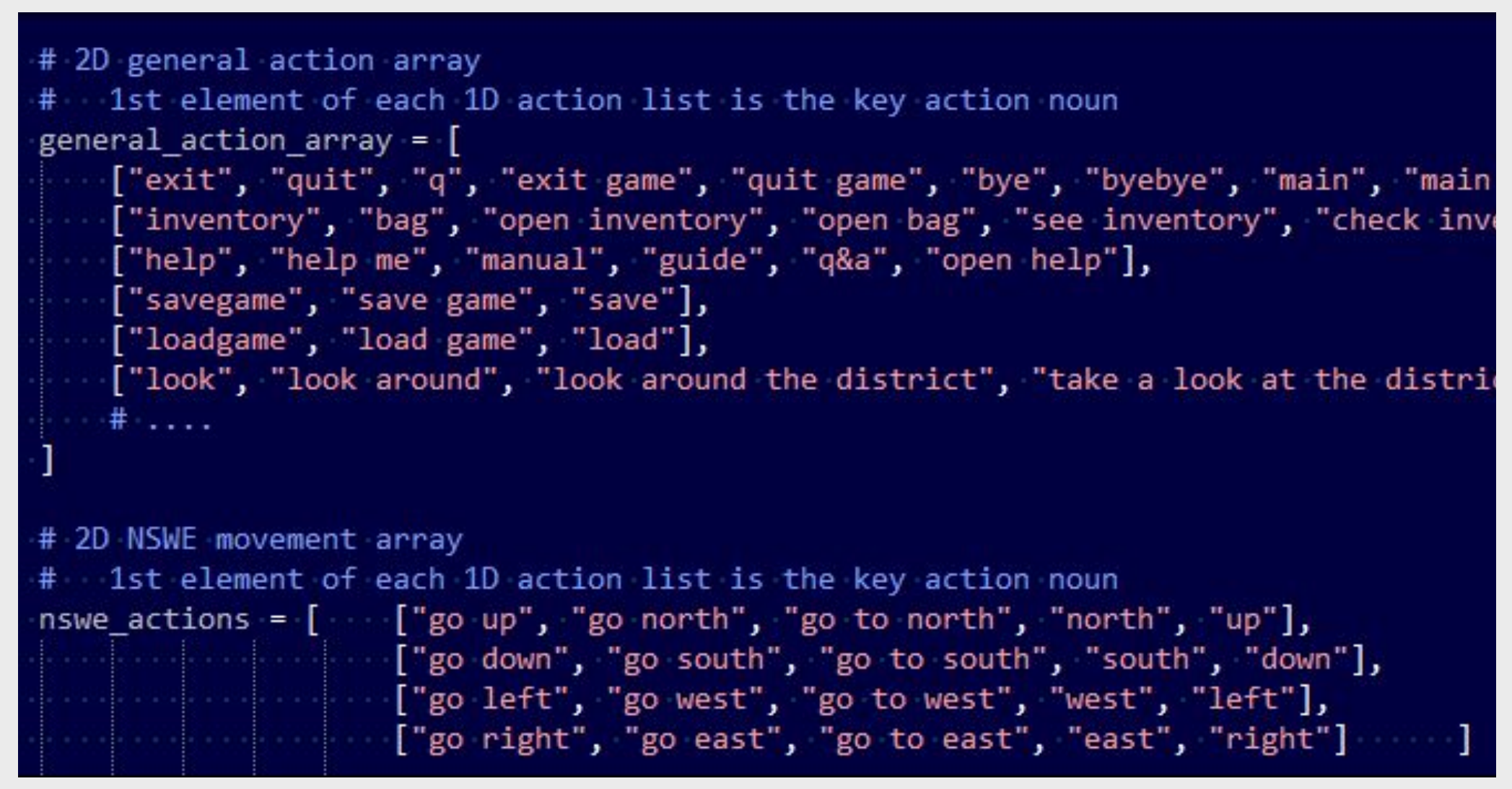
```
{
  "name": "Crystal Chest",
  "item_type": "FEATURE",
  "narration": "You can see a Crystal Chest nearby.",
  "description": "A giant cooler-sized Crystal Chest sits on
                  the ground. It looks like it has a keyhole.",
  "actions": ["etc/items/actions/view_crystal_chest.json",
    "etc/items/actions/talk_crystal_chest.json",
    "etc/items/actions/eat_crystal_chest.json",
    "etc/items/actions/move_crystal_chest.json",
    "etc/items/actions/climb_crystal_chest.json",
    "etc/items/actions/feed_crystal_chest.json",
    "etc/items/actions/hit_crystal_chest.json",
    "etc/items/actions/lift_crystal_chest.json",
    "etc/items/actions/take_crystal_chest.json",
    "etc/items/actions/use_crystal_chest.json",
    "etc/items/actions/enter_crystal_chest.json",
    "etc/items/actions/swim_crystal_chest.json"],
}
```

```
{
  "short_description": "A construction worker.",
  "long_description": "A construction worker working on a
housing development.",
  "puzzle": "etc/puzzles/construction_worker_puzzle.json",
  "narration": "A Construction Worker is surveying the area.",
  "dialogs": [
           "You approach the Construction Worker. He says: 'I
               got a bit of a thinker for you. Want to solve
               it?'",
           "You want another try at my puzzle?",
           "Yeah, I don't think that's it.",
           "Nice job. Sorry, I don't have anything to offer
               you but at least you had fun, eh?",
           "Sorry, can't chat. I have to finish this before
             the boss gets on my case." ],
  "action":[
    "etc/characters/actions/look_at_construction_worker.json",
    "etc/characters/actions/take_construction_worker.json",
    "etc/characters/actions/use_construction_worker.json",
    "etc/characters/actions/talk_to_construction_worker.json",
    "etc/characters/actions/enter_construction_worker.json",
    "etc/characters/actions/eat_construction_worker.json",
    "etc/characters/actions/hit_construction_worker.json",
    "etc/characters/actions/climb_construction_worker.json",
    "etc/characters/actions/move_construction_worker.json",
    "etc/characters/actions/swim_construction_worker.json",
    "etc/characters/actions/lift_construction_worker.json",
    "etc/characters/actions/feed_construction_worker.json"
  ]
}
```

```
{
  "commands": [
    "talk construction worker",
    "talk worker",
    "talk construction"
  ],
  "response_type": "event",
  "response": ""
}
```

Code above is an example of the file structure (above). Uses JSON for game content storage. Recursive loading of files using relative file paths.

The protagonist, Agent Dope (above left), on a mission to stop the antagonist, Dr. Crime (above right).



```
# 2D general action array
#   1st element of each 1D action list is the key action noun
general_action_array = [
    ["exit", "quit", "q", "exit game", "quit game", "bye", "byebye", "main", "main
    ["inventory", "bag", "open inventory", "open bag", "see inventory", "check inv
    ["help", "help me", "manual", "guide", "q&a", "open help"],
    ["savegame", "save game", "save"],
    ["loadgame", "load game", "load"],
    ["look", "look around", "look around the district", "take a look at the distri
    # ....
]

# 2D NSWE movement array
#   1st element of each 1D action list is the key action noun
nswe_actions = [    ["go up", "go north", "go to north", "north", "up"],
                    ["go down", "go south", "go to south", "south", "down"],
                    ["go left", "go west", "go to west", "west", "left"],
                    ["go right", "go east", "go to east", "east", "right"]     ]
```

A snapshot of core commands available during game play and directional navigation (above).

DESCRIPTION

This game is a text-based adventure-puzzle game coded in Python 3. The game uses self-implemented data loaders to load in game content from files encoded in JSON.

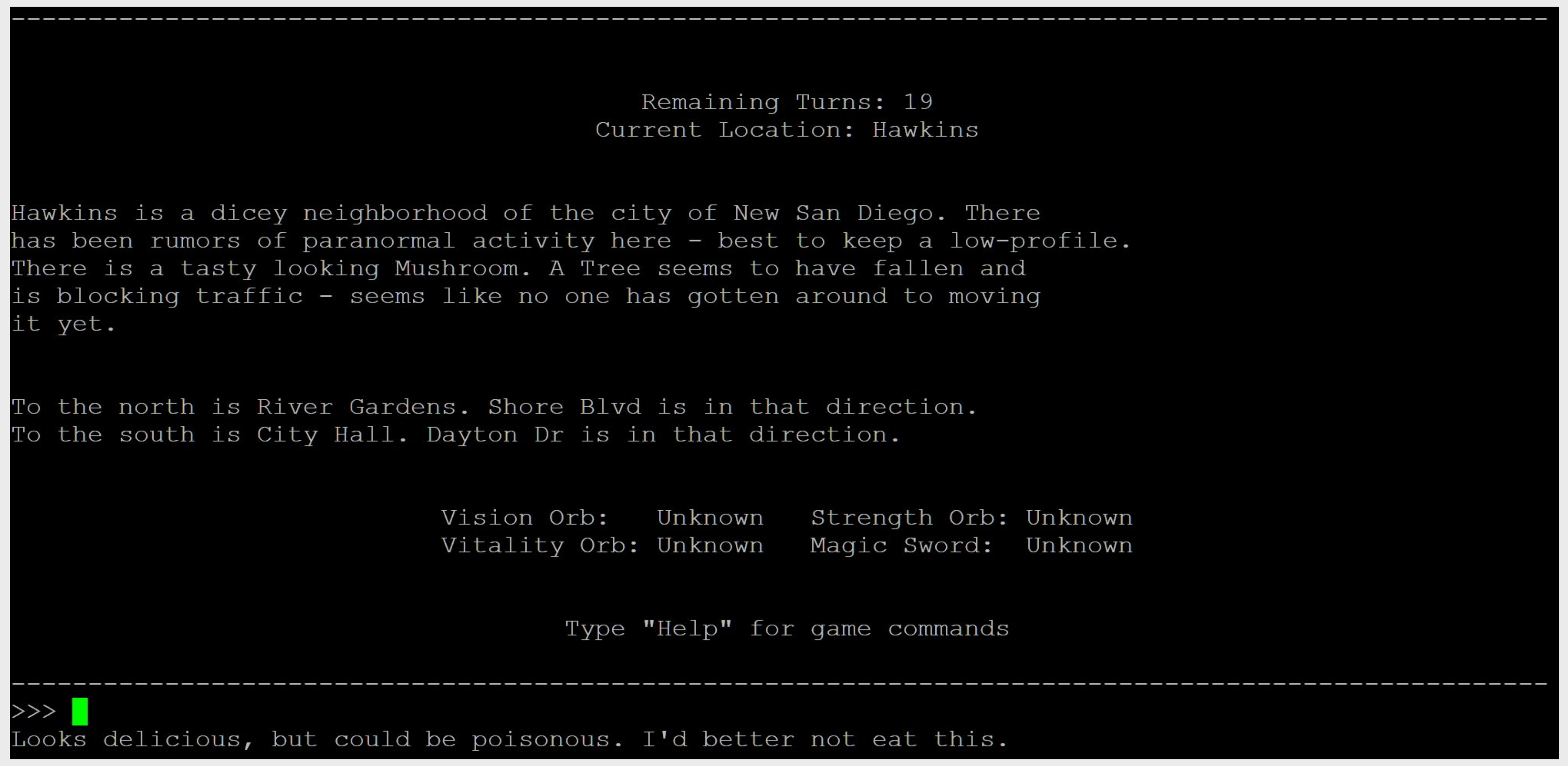
The user takes on the role of Agent Dope and is tasked with mission of stopping Dr. Crime's evil plan. The user must explore the game's fictional setting, the City of New San Diego, solve puzzles, and collect items to do so.

A recognizable aspect of object-oriented programming in the game is the frequent use of "has-a" relationships in the class hierarchy. This facilitated modularization and sustainable expansion.

A large set of core commands combined with a natural language parser enables the use of a variety of command phrases. For example, navigation can be done via relative direction (up / down / left / right), cardinal direction (north / south / west / east), immediate adjacent district names, or street names, e.g. "go up", "walk to hawkins". Characters and items can be interacted, e.g. "speak to student", "look at statue". Many verbs have common aliases, e.g. "pick up", "get", "obtain" are aliases for "take".

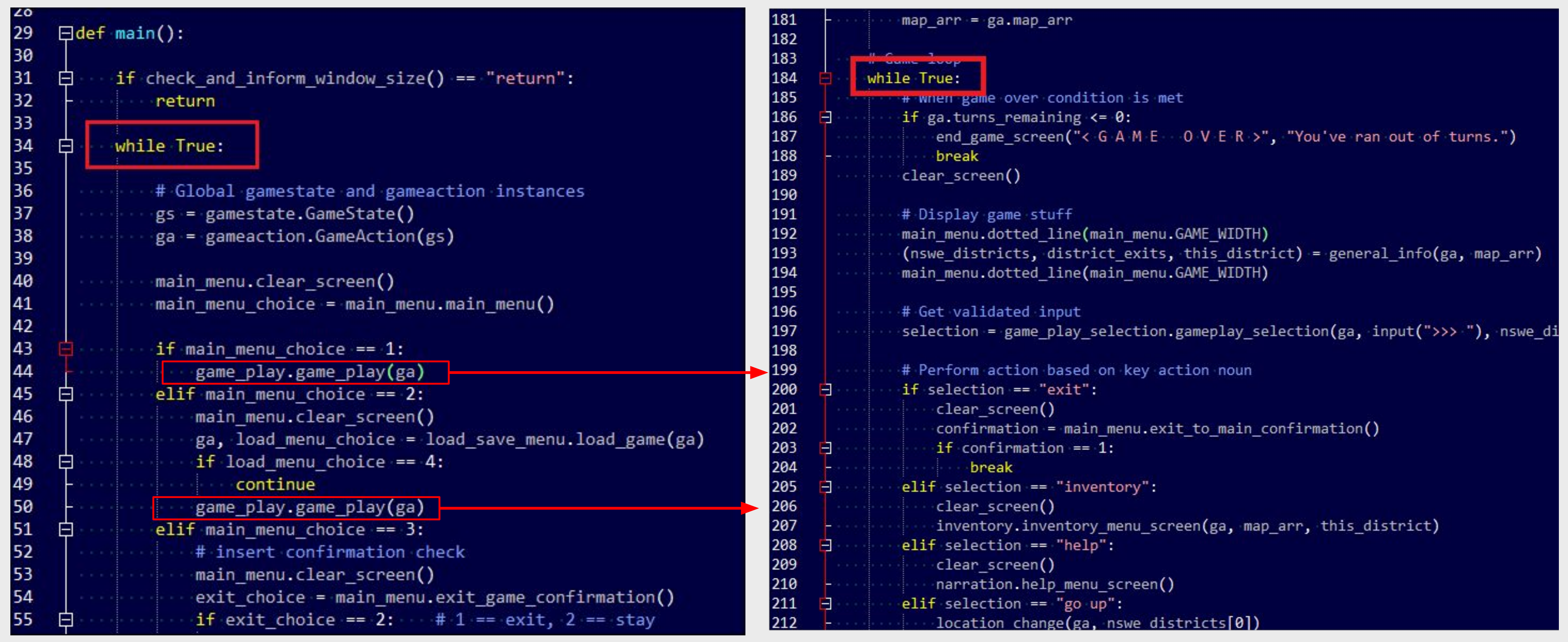
GAME FEATURES

- Intuitive user interface and game flow.
- Fun and entertaining gameplay.
- Smart natural language parser.
- Open-world setting with non-linear gameplay.
- 15 unique districts to explore around.
- Understands 10+ core action verbs and many common aliases.
- Many character, puzzles, and items to interact with.
- Game consists of over 500 files.
- Easy to expand game content with aid of self-implemented JSON loaders.
- Save and load game functionality to non human-readable files prevents invalid and corrupted game states.
- Interactive help and inventory menu increases user's experience.



```
--------------------------------------------------------------------------------

                         Remaining Turns: 19
                      Current Location: Hawkins


Hawkins is a dicey neighborhood of the city of New San Diego. There
has been rumors of paranormal activity here - best to keep a low-profile.
There is a tasty looking Mushroom. A Tree seems to have fallen and
is blocking traffic - seems like no one has gotten around to moving
it yet.


To the north is River Gardens. Shore Blvd is in that direction.
To the south is City Hall. Dayton Dr is in that direction.


                  Vision Orb:    Unknown     Strength Orb: Unknown
                  Vitality Orb: Unknown      Magic Sword:  Unknown


                       Type "Help" for game commands

--------------------------------------------------------------------------------
>>>
Looks delicious, but could be poisonous. I'd better not eat this.
```

Example of user interface during gameplay (above).



```
def main():

    if check_and_inform_window_size() == "return":
        return

    while True:

        # Global gamestate and gameaction instances
        gs = gamestate.GameState()
        ga = gameaction.GameAction(gs)

        main_menu.clear_screen()
        main_menu_choice = main_menu.main_menu()

        if main_menu_choice == 1:
            game_play.game_play(ga)
        elif main_menu_choice == 2:
            main_menu.clear_screen()
            ga, load_menu_choice = load_save_menu.load_game(ga)
            if load_menu_choice == 4:
                continue
            game_play.game_play(ga)
        elif main_menu_choice == 3:
            # insert confirmation check
            main_menu.clear_screen()
            exit_choice = main_menu.exit_game_confirmation()
            if exit_choice == 2:    # 1 == exit, 2 == stay
```

```
        map_arr = ga.map_arr

    # Game loop
    while True:
        # when game over condition is met
        if ga.turns_remaining <= 0:
            end_game_screen("< G A M E   O V E R >", "You've ran out of turns.")
            break
        clear_screen()

        # Display game stuff
        main_menu.dotted_line(main_menu.GAME_WIDTH)
        (nswe_districts, district_exits, this_district) = general_info(ga, map_arr)
        main_menu.dotted_line(main_menu.GAME_WIDTH)

        # Get validated input
        selection = game_play_selection.gameplay_selection(ga, input(">>> "), nswe_di

        # Perform action based on key action noun
        if selection == "exit":
            clear_screen()
            confirmation = main_menu.exit_to_main_confirmation()
            if confirmation == 1:
                break
        elif selection == "inventory":
            clear_screen()
            inventory.inventory_menu_screen(ga, map_arr, this_district)
        elif selection == "help":
            clear_screen()
            narration.help_menu_screen()
        elif selection == "go up":
            location_change(ga, nswe_districts[0])
```

A simple pair of nested infinite loops keeps the running until an end condition is satisfied (above).