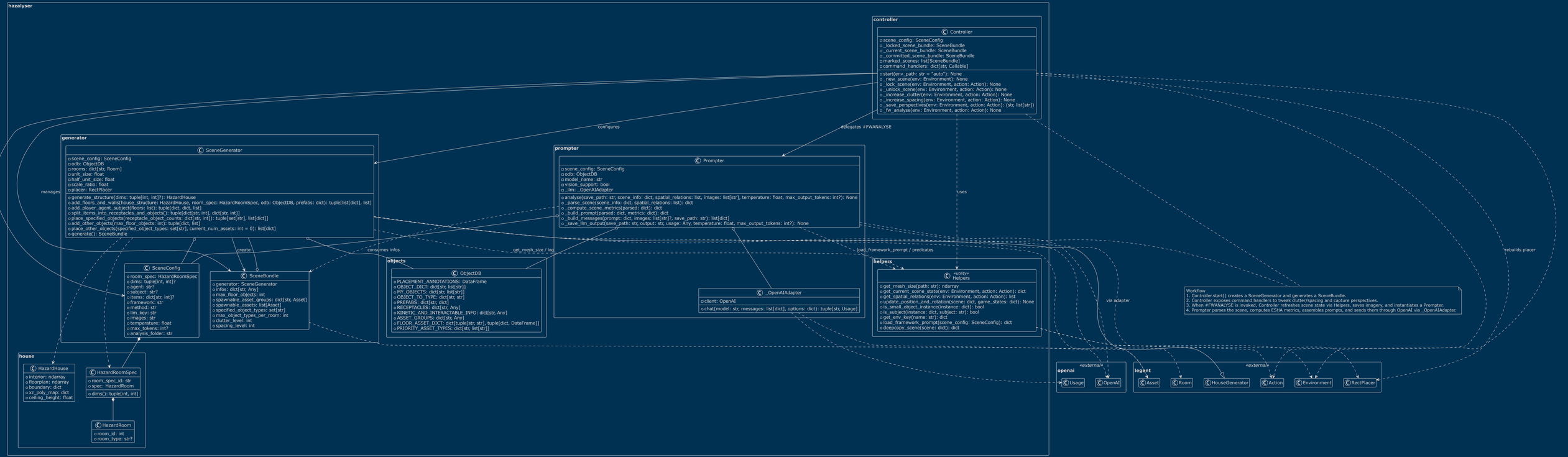
hazalyser
controller
Controller
scene_config: SceneConfig
_locked_scene_bundle: SceneBundle
_current_scene_bundle: SceneBundle
_committed_scene_bundle: SceneBundle
marked_scenes: list[SceneBundle]
command_handlers: dict[str, Callable]
start(env_path: str = "auto"): None
_new_scene(env: Environment): None
_lock_scene(env: Environment, action: Action): None
_unlock_scene(env: Environment, action: Action): None
_increase_clutter(env: Environment, action: Action): None
_increase_spacing(env: Environment, action: Action): None
_save_perspectives(env: Environment, action: Action): (str, list[str])
_fw_analyse(env: Environment, action: Action): None
configures
delegates #FWANALYSE
uses
manages
generator
SceneGenerator
scene_config: SceneConfig
odb: ObjectDB
rooms: dict[str, Room]
unit_size: float
half_unit_size: float
scale_ratio: float
placer: RectPlacer
generate_structure(dims: tuple[int, int]?): HazardHouse
add_floors_and_walls(house_structure: HazardHouse, room_spec: HazardRoomSpec, odb: ObjectDB, prefabs: dict): tuple[list[dict], list]
add_player_agent_subject(floors: list): tuple[dict, dict, list]
split_items_into_receptacles_and_objects(): tuple[dict[str, int], dict[str, int]]
place_specified_objects(receptacle_object_counts: dict[str, int]): tuple[set[str], list[dict]]
add_other_objects(max_floor_objects: int): tuple[dict, list]
place_other_objects(specified_object_types: set[str], current_num_assets: int = 0): list[dict]
generate(): SceneBundle
create
consumes infos
get_mesh_size / log
prompter
Prompter
scene_config: SceneConfig
odb: ObjectDB
model_name: str
vision_support: bool
_llm: _OpenAIAdapter
analyse(save_path: str, scene_info: dict, spatial_relations: list, images: list[str], temperature: float, max_output_tokens: int?): None
_parse_scene(scene_info: dict, spatial_relations: list): dict
_compute_scene_metrics(parsed: dict): dict
_build_prompt(parsed: dict, metrics: dict): dict
_build_messages(prompt: dict, images: list[str]?, save_path: str): list[dict]
_save_llm_output(save_path: str, output: str, usage: Any, temperature: float, max_output_tokens: int?): None
load_framework_prompt / predicates
rebuilds placer
via adapter
SceneConfig
room_spec: HazardRoomSpec
dims: tuple[int, int]?
agent: str?
subject: str?
items: dict[str, int]?
framework: str
method: str
llm_key: str
images: str
temperature: float
max_tokens: int?
analysis_folder: str
SceneBundle
generator: SceneGenerator
infos: dict[str, Any]
max_floor_objects: int
spawnable_asset_groups: dict[str, Asset]
spawnable_assets: list[Asset]
specified_object_types: set[str]
max_object_types_per_room: int
clutter_level: int
spacing_level: int
objects
ObjectDB
PLACEMENT_ANNOTATIONS: DataFrame
OBJECT_DICT: dict[str, list[str]]
MY_OBJECTS: dict[str, list[str]]
OBJECT_TO_TYPE: dict[str, str]
PREFABS: dict[str, dict]
RECEPTACLES: dict[str, Any]
KINETIC_AND_INTERACTABLE_INFO: dict[str, Any]
ASSET_GROUPS: dict[str, Any]
FLOOR_ASSET_DICT: dict[tuple[str, str], tuple[dict, DataFrame]]
PRIORITY_ASSET_TYPES: dict[str, list[str]]
_OpenAIAdapter
client: OpenAI
chat(model: str, messages: list[dict], options: dict): tuple[str, Usage]
helpers
«utility»
Helpers
get_mesh_size(path: str): ndarray
get_current_scene_state(env: Environment, action: Action): dict
get_spatial_relations(env: Environment, action: Action): list
update_position_and_rotation(scene: dict, game_states: dict): None
is_small_object_instance(instance: dict): bool
is_subject(instance: dict, subject: str): bool
get_env_key(name: str): dict
load_framework_prompt(scene_config: SceneConfig): dict
deepcopy_scene(scene: dict): dict
Workflow
1. Controller.start() creates a SceneGenerator and generates a SceneBundle.
2. Controller exposes command handlers to tweak clutter/spacing and capture perspectives.
3. When #FWANALYSE is invoked, Controller refreshes scene state via Helpers, saves imagery, and instantiates a Prompter.
4. Prompter parses the scene, computes ESHA metrics, assembles prompts, and sends them through OpenAI via _OpenAIAdapter.
house
HazardHouse
interior: ndarray
floorplan: ndarray
boundary: dict
xz_poly_map: dict
ceiling_height: float
HazardRoomSpec
room_spec_id: str
spec: HazardRoom
dims(): tuple[int, int]
HazardRoom
room_id: int
room_type: str?
«external»
openai
Usage
OpenAI
«external»
legant
Asset
Room
HouseGenerator
Action
Environment
RectPlacer