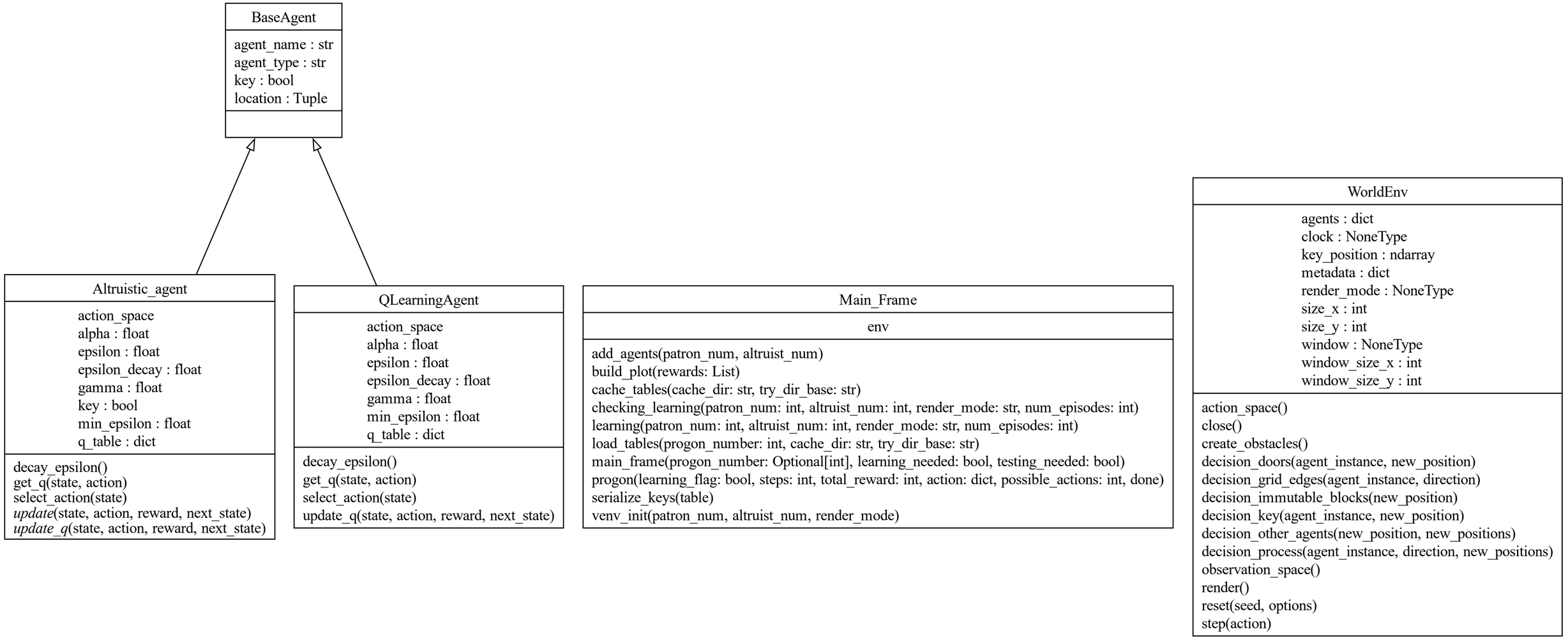
BaseAgent
agent_name : str
agent_type : str
key : bool
location : Tuple
Altruistic_agent
action_space
alpha : float
epsilon : float
epsilon_decay : float
gamma : float
key : bool
min_epsilon : float
q_table : dict
decay_epsilon()
get_q(state, action)
select_action(state)
update(state, action, reward, next_state)
update_q(state, action, reward, next_state)
QLearningAgent
action_space
alpha : float
epsilon : float
epsilon_decay : float
gamma : float
min_epsilon : float
q_table : dict
decay_epsilon()
get_q(state, action)
select_action(state)
update_q(state, action, reward, next_state)
Main_Frame
env
add_agents(patron_num, altruist_num)
build_plot(rewards: List)
cache_tables(cache_dir: str, try_dir_base: str)
checking_learning(patron_num: int, altruist_num: int, render_mode: str, num_episodes: int)
learning(patron_num: int, altruist_num: int, render_mode: str, num_episodes: int)
load_tables(progon_number: int, cache_dir: str, try_dir_base: str)
main_frame(progon_number: Optional[int], learning_needed: bool, testing_needed: bool)
progon(learning_flag: bool, steps: int, total_reward: int, action: dict, possible_actions: int, done)
serialize_keys(table)
venv_init(patron_num, altruist_num, render_mode)
WorldEnv
agents : dict
clock : NoneType
key_position : ndarray
metadata : dict
render_mode : NoneType
size_x : int
size_y : int
window : NoneType
window_size_x : int
window_size_y : int
action_space()
close()
create_obstacles()
decision_doors(agent_instance, new_position)
decision_grid_edges(agent_instance, direction)
decision_immutable_blocks(new_position)
decision_key(agent_instance, new_position)
decision_other_agents(new_position, new_positions)
decision_process(agent_instance, direction, new_positions)
observation_space()
render()
reset(seed, options)
step(action)