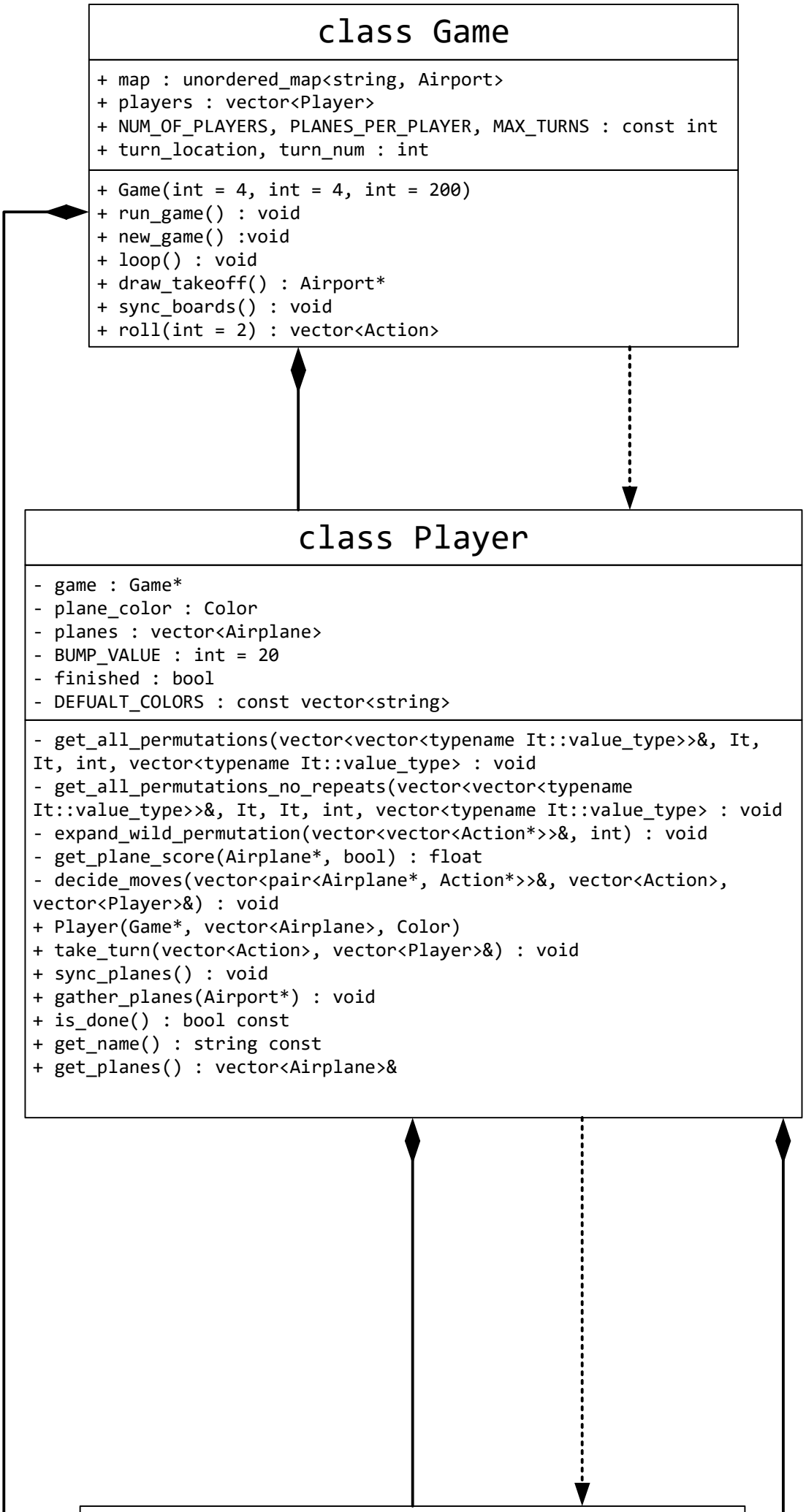
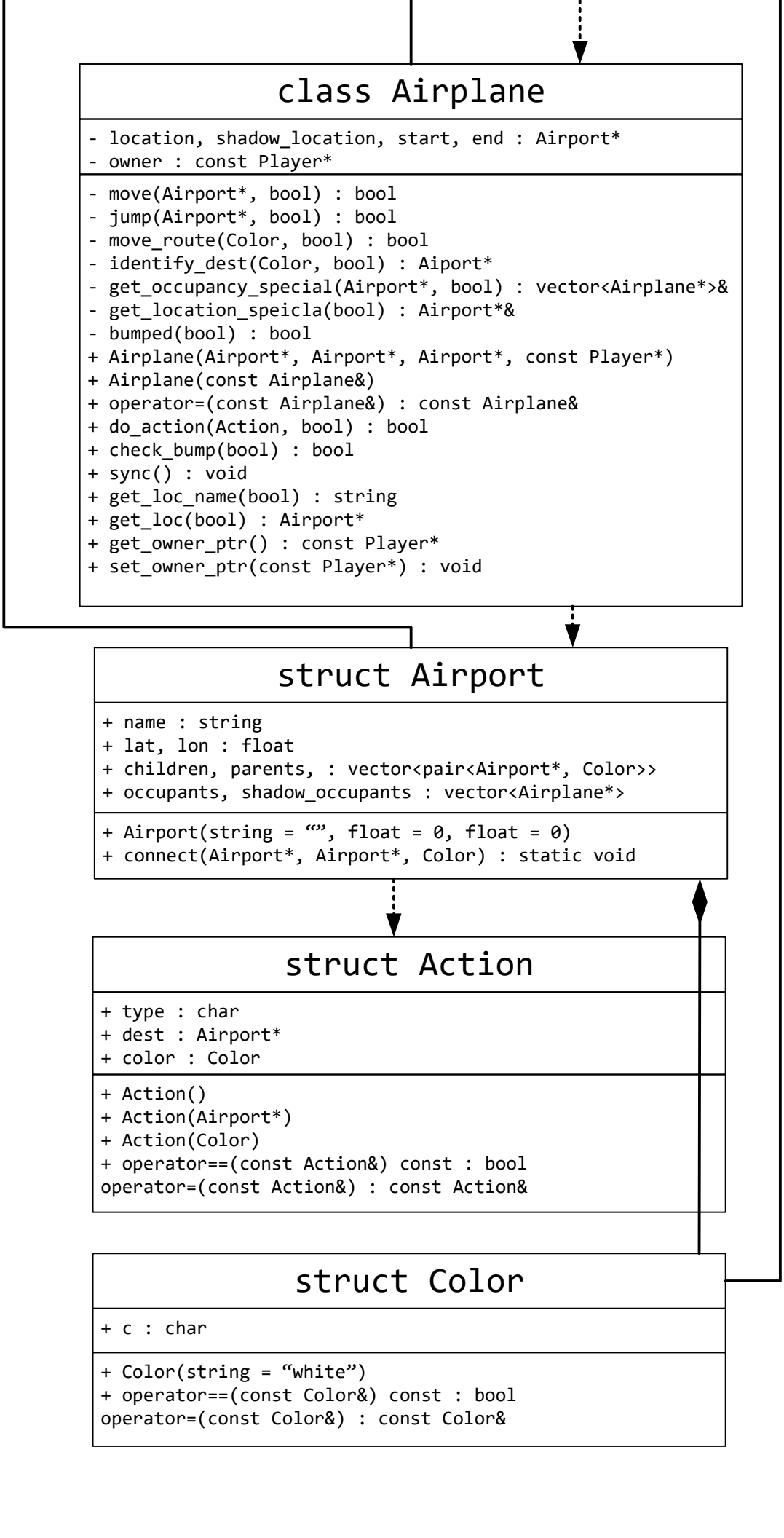

class Game
+ map : unordered_map<string, Airport>
+ players : vector<Player>
+ NUM_OF_PLAYERS, PLANES_PER_PLAYER, MAX_TURNS : const int
+ turn_location, turn_num : int
+ Game(int = 4, int = 4, int = 200)
+ run_game() : void
+ new_game() :void
+ loop() : void
+ draw_takeoff() : Airport*
+ sync_boards() : void
+ roll(int = 2) : vector<Action>
class Player
- game : Game*
- plane_color : Color
- planes : vector<Airplane>
- BUMP_VALUE : int = 20
- finished : bool
- DEFUALT_COLORS : const vector<string>
- get_all_permutations(vector<vector<typename It::value_type>>&, It, It, int, vector<typename It::value_type> : void
- get_all_permutations_no_repeats(vector<vector<typename It::value_type>>&, It, It, int, vector<typename It::value_type> : void
- expand_wild_permutation(vector<vector<Action*>>&, int) : void
- get_plane_score(Airplane*, bool) : float
- decide_moves(vector<pair<Airplane*, Action*>>&, vector<Action>, vector<Player>&) : void
+ Player(Game*, vector<Airplane>, Color)
+ take_turn(vector<Action>, vector<Player>&) : void
+ sync_planes() : void
+ gather_planes(Airport*) : void
+ is_done() : bool const
+ get_name() : string const
+ get_planes() : vector<Airplane>&


class Airplane
- location, shadow_location, start, end : Airport*
- owner : const Player*
- move(Airport*, bool) : bool
- jump(Airport*, bool) : bool
- move_route(Color, bool) : bool
- identify_dest(Color, bool) : Aiport*
- get_occupancy_special(Airport*, bool) : vector<Airplane*>&
- get_location_speicla(bool) : Airport*&
- bumped(bool) : bool
+ Airplane(Airport*, Airport*, Airport*, const Player*)
+ Airplane(const Airplane&)
+ operator=(const Airplane&) : const Airplane&
+ do_action(Action, bool) : bool
+ check_bump(bool) : bool
+ sync() : void
+ get_loc_name(bool) : string
+ get_loc(bool) : Airport*
+ get_owner_ptr() : const Player*
+ set_owner_ptr(const Player*) : void
struct Airport
+ name : string
+ lat, lon : float
+ children, parents, : vector<pair<Airport*, Color>>
+ occupants, shadow_occupants : vector<Airplane*>
+ Airport(string = "", float = 0, float = 0)
+ connect(Airport*, Airport*, Color) : static void
struct Action
+ type : char
+ dest : Airport*
+ color : Color
+ Action()
+ Action(Airport*)
+ Action(Color)
+ operator==(const Action&) const : bool
operator=(const Action&) : const Action&
struct Color
+ c : char
+ Color(string = "white")
+ operator==(const Color&) const : bool
operator=(const Color&) : const Color&