WIN), WinConditionStrategy (abstract class) was created WinConditionStrategy ______ c-----+ check_win(player: Player, worker: Worker, board: Board): bool + check_lose(player: Player, board: Board): bool TutorialStep Composite Win Condition that allows combination of + get_step_name(): str multiple win strategy. + get_instructions(): str + is_valid_tile_click(tile: Tile, current_player: Player, current_worker: Worker, phase: str): bool + get_highlighted_tiles(current_player: Player, TimerWinCondition CompositeWinCondition current_worker: Worker, phase: str, board: Board): bool + is_step_complete(current_player: Player, current_worker: win_conditions: List[WinConditionStrategy] Worker, phase: str, board: Board): bool + add_win_condition(win_condition: WinConditionStrategy): None + handles_click_progression(): bool + remove_win_condition(win_condition: WinConditionStrategy: + check_win(player: Player, worker: Worker, board: Board): bool + check_win(player: Player, worker: Worker, board: Board): bool + handle_tile_click(tile: Tile, current_player: Player, + *check_lose*(player: Player, board: Board): bool + check_lose(player: Player, board: Board): bool current_worker: Worker, phase: str): None + check_win(player: Player, worker: Worker, board: Board): bool + should_auto_advance(): bool check_lose(player: Player, board: Board): bool Standard Win Condition (lv2 Updated Sprint 2's Class Diagram When time runs out =-> lv3), lose when both workers unable to move Extensions planned to be implemented in Sprint 3: - Timer for both players Human Value Extension : GamePhaseManager Tutorial actions: Optional[Sequence] WinConditionChecker - New God Card: Triton worker: Optional[Worker] GameInputHandler strategy: WinConditionStrategy + action_sequence(): Optional[Sequence] - turn_manager: TurnManager SelectWorkerStep MoveToTrapStep MoveToLevel3Step IntroductionStep + initialize_turn(action_sequence: Optional[Sequence]): None + check_win(player: Player, worker: Worker, board: Board) + get_current_phase(): str ------ move_completed: bool - tutorial_type: str - tutorial_type: str + check_lose(player: Player, board: Board) + handle_tile_click(tile: Tile): None is_current_phase_optional(): bool demonstration_shown: bool + determine_winner(players: List[Player], current_player: Player, + advance_phase(): None - handle_worker_selection(tile: Tile): None To handle user worker: Worker, board: Board): Player - handle_action_execution(tile: Tile): None + get_step_name(): str + is_turn_complete(): bool It's colored so + get_instructions(): str + get_instructions(): str + get_step_name(): str + get_instructions(): str + get_instructions(): str + get_instructions(): str + get_instructions(): str + get_current_action(): None input such as + is_valid_tile_click(tile: Tile, current_player: Player, is_valid_tile_click(tile: Tile, current_player: Player, + has_worker_selected(): bool + get_instructions(): str worker selection. current_worker: Worker, phase: str): bool current_worker: Worker, phase: str): bool is_valid_tile_click(tile: Tile, current_player: Player, current_worker: Worker, phase: str): bool + handle_action_result(result: ActionResult): None + get_highlighted_tiles(current_player: Player, tile clicking, + get_highlighted_tiles(current_player: Player, current_worker: Worker, phase: str): bool + get_highlighted_tiles(current_player: Player, + get_highlighted_tiles(current_player: Player, + get_highlighted_tiles(current_player: Player, + get_highlighted_tiles(current_player: Player, association— current_worker: Worker, phase: str, board: Board): bool current_worker: Worker, phase: str, board: Board): bool + get_highlighted_tiles(current_player: Player, current_worker: Worker, phase: str, board: Board): bool differentiate the is_step_complete(current_player: Player, current_worker: + is_step_complete(current_player: Player, current_worker: is_step_complete(current_player: Player, current_worker: is_step_complete(current_player: Player, current_worker: + is_step_complete(current_player: Player, current_worker: current_worker: Worker, phase: str, board: Board): bool is_step_complete(current_player: Player, current_worker: Worker, phase: str, board: Board): bool Worker, phase: str, board: Board): bool Worker, phase: str, board: Board): bool is_step_complete(current_player: Player, current_worker: Worker, phase: str, board: Board): bool Worker, phase: str, board: Board): bool Worker, phase: str, board: Board): bool + should_auto_advance(): bool Worker, phase: str, board: Board): bool + should_auto_advance(): bool + handle_tile_click(tile: Tile, current_player: Player, + handles_click_progression(): bool _______ current_worker: Worker, phase: str): None + should_auto_advance(): bool ______ TutorialObserver - height: int Sequence - items: List[Any] + in_bounds(pos: Position): bool - index: int + get_all_empty_tiles(): List[Tile] + on_tutorial_step_changed(step: TutorialStep): None methods/attributes + get_tile(pos: Position): Tile | None + on_tutorial_completed(): None SantoriniApp ______ xample_method(): None + current(): Any Validator + advance(): None + initialize_game(players: List[Player], board: Board): None - root: tk.Tk + reset_index(): None + is_valid_tile_click(tile: Tile, current_player: Player, _____ - game: Game + handle_action_result(result: ActionResult): None current_worker: Worker, phase: str): bool - rect_ids: dict[tuple[int, int], int] TutorialManager s_complete(): bool + get_highlighted_tiles(current_player: Player, - player1_name_var: tk.StringVar - players: Sequence + get_valid_move_tiles(worker: current_worker: Worker, phase: str, board: Board): bool - player2_name_var: tk.StringVar - steps: List[TutorialStep] Game board: Board Worker, board: Board): list[Tiles] + get_guidance_message(tile: Tile, current_player: Player, player1_color_var: tk.StringVar - current_step_index: int phase_manager: GamePhaseManager + get_valid_build_tiles(worker: current_worker: Worker, phase: str): str player2_color_var: tk.StringVar - observers: List[TutorialObserver] Worker, board: Board): list[Tiles] + should_end_game(): bool players: list[Player] - main_menu_frame: tk.Frame is_tutorial_complete: bool - timer_manager: TimerManager + get_mode_name(): str turn_manager: TurnManager - setup_frame: tk.Frame - introduction_clicked: bool + handle_post_action_update(urrent_player: Player, god_card_deck: GodCardDeck - game_frame: tk.Frame tutorial_type: str initializes and | - canvas: tk.Canvas + get_phase(): str current_worker: Worker, phase: str, board: Board): None timer_manager: TimerManager Notes/brief explanation + current_phase_optional(): bool + handle_tile_click(current_player: Player, current_worker: input_handler: GameInputHandle - skip_button: tk.Button + add_observer(observer: TutorialObserver): None bout the modified/new class Worker, phase: str, board: Board): None + skip_phase(): None - turn_label: tk.Label + remove_observer(observer: TutorialObserver): None + start_turn(): None + get_current_phase(): str - phase_label: tk.Label + get_current_step(): TutorialStep + end_turn(): None current_phase_optional(): bool - god_name: tk.Label + get_current_instructions(): str + get_game_result(): Player - skip_phase(): bool - worker_icon: tk.PhotoImage - advance_step(): None place_random_workers(player: Player) + complete_tutorial(): None pick_random_god(player: Player) + start_game(): None - notify_step_changed(step: TutorialStep): None + click_cell(bx: int, by: int): bool GodCardFactory - build_main_menu_ui(): None - notify_tutorial_completed(): None + selected_worker_pos(): Tuple(int,int) | additional_actions - build_setup_ui(): None registered_cards - show_setup_screen(): None StandardGameMode TutorialGameMode + has_additional_actions(): List[Action - build_game_ui(): None - draw_board(): None + is_empty(): bool + create_card() - tutorial_manager: TutorialManager + on_click(evt: tk.Event): None + register_card() tutorial_type: str + on_skip(): None + get_available_card_names() \bigvee \bigvee \bigvee GodCardDeck + initialize_game(players: List[Player], board: Board): None initialize_game(players: List[Player], board: Board): None Position To handle post action + get_guidance_message(tile: Tile, current_player: Player, get_guidance_message(tile: Tile, current_player: Player, cards: List[GodCard] execution current_worker: Worker, phase: str): str current_worker: Worker, phase: str): str creates/register + should_end_game(): bool + should_end_game(): bool owns draw(): GodCard + get_mode_name(): str + get_mode_name(): str - setup_tutorial_board(player: Player, board: Board): None + add_card(god_card: GodCard): None - place_random_workers(player: Player, board: Board): None + *execute*(worker: Worker, board: - clear_board(board: Board): None is_empty(): None Board, target_tile: Tile): ActionResu + validate(worker: Worker, board: + handle_post_action_update(urrent_player: Player, + remaining(): List[GodCard] current_worker: Worker, phase: str, board: Board): None Board, target_tile: Tile): None + handle_tile_click(current_player: Player, current_worker: + get_name(): str Worker, phase: str, board: Board): None GodCard - position: Position Building - worker: Worker | None MoveAction - name: str - building: Building | None _____ - description wields— Modified Action classes <<enumeration>> cute() method to return + has_worker(): bool + has_dome(): bool + get_action_sequence(): List[Action] Color ActionResult execute(worker: Worker, board: execute(worker: Worker, board: + get_level(): int Board, target_tile: Tile): ActionResul Board, target_tile: Tile): ActionRe + increase_level(): None validate(worker: Worker, board: + validate(worker: Worker, board: Board, target_tile: Tile): bool Board, target_tile: Tile): bool ArtemisMoveAction DemeterBuildAction Artemis Demeter - player_name: str K----- player_age: str player_color: Color execute(worker: Worker, board: - level: int + validate(worker: Worker, board: - workers: List[Worker] | None has Board, tile: Tile) Board, target_tile: Tile): None - player_god: GodCard | None + validate(worker: Worker, board: is_perimeter_space(position: Board, target_tile: Tile): None Position, board; Board) L______J - previous_position: Position - set_position(new_position: current_timer_player: Player timer_enabled: bool remaining_time: float start_player_timer(player: Player): None start_time: float pause_current_timer(): None is_active: bool switch_timer_to_player(player: Player): None is_expired: bool + get_current_player_timer_info(): TimerInfo + get_all_players_timer_info(): Dict[[str, PlayerTimerInfo] - start(): None + check_for_timer_expiration(): Player - pause(): None + has_any_timer_expired: bool - reset(): None + get_expired_players(): List[Player] - add_time(seconds: float): None format_time(seconds: float): str + reset_all_timers(): None get_formatted_time(): str + get_players_with_timers(): List[Player]

Since there are extra win conditions introduced (TIMER