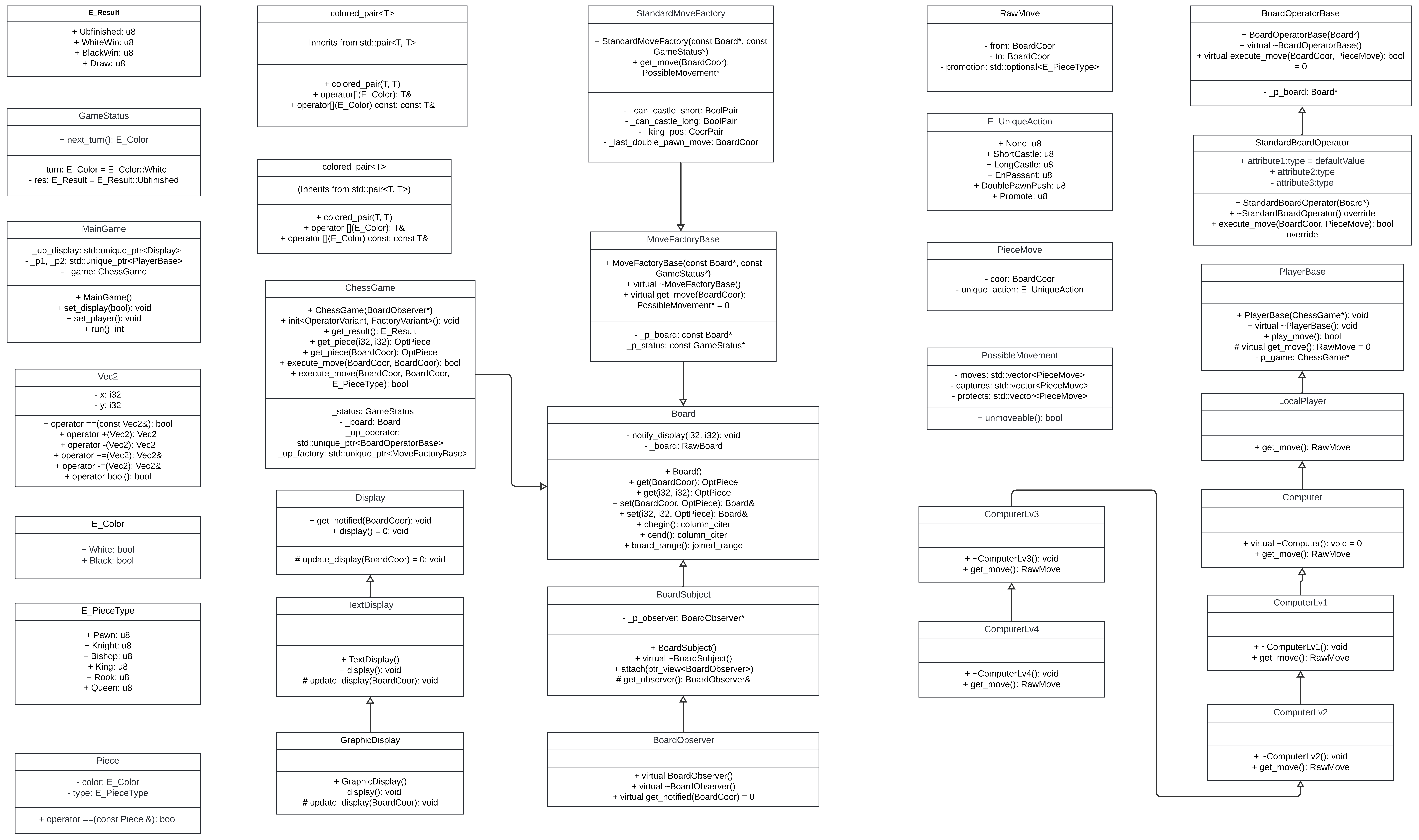
E_Result
+ Ubfinished: u8
+ WhiteWin: u8
+ BlackWin: u8
+ Draw: u8
GameStatus
+ next_turn(): E_Color
- turn: E_Color = E_Color::White
- res: E_Result = E_Result::Ubfinished
MainGame
- _up_display: std::unique_ptr<Display>
- _p1, _p2: std::unique_ptr<PlayerBase>
- _game: ChessGame
+ MainGame()
+ set_display(bool): void
+ set_player(): void
+ run(): int
Vec2
- x: i32
- y: i32
+ operator ==(const Vec2&): bool
+ operator +(Vec2): Vec2
+ operator -(Vec2): Vec2
+ operator +=(Vec2): Vec2&
+ operator -=(Vec2): Vec2&
+ operator bool(): bool
E_Color
+ White: bool
+ Black: bool
E_PieceType
+ Pawn: u8
+ Knight: u8
+ Bishop: u8
+ King: u8
+ Rook: u8
+ Queen: u8
Piece
- color: E_Color
- type: E_PieceType
+ operator ==(const Piece &): bool
colored_pair<T>
Inherits from std::pair<T, T>
+ colored_pair(T, T)
+ operator[](E_Color): T&
+ operator[](E_Color) const: const T&
colored_pair<T>
(Inherits from std::pair<T, T>)
+ colored_pair(T, T)
+ operator [](E_Color): T&
+ operator [](E_Color) const: const T&
ChessGame
+ ChessGame(BoardObserver*)
+ init<OperatorVariant, FactoryVariant>(): void
+ get_result(): E_Result
+ get_piece(i32, i32): OptPiece
+ get_piece(BoardCoor): OptPiece
+ execute_move(BoardCoor, BoardCoor): bool
+ execute_move(BoardCoor, BoardCoor, E_PieceType): bool
- _status: GameStatus
- _board: Board
- _up_operator: std::unique_ptr<BoardOperatorBase>
- _up_factory: std::unique_ptr<MoveFactoryBase>
Display
+ get_notified(BoardCoor): void
+ display() = 0: void
# update_display(BoardCoor) = 0: void
TextDisplay
+ TextDisplay()
+ display(): void
# update_display(BoardCoor): void
GraphicDisplay
+ GraphicDisplay()
+ display(): void
# update_display(BoardCoor): void
StandardMoveFactory
+ StandardMoveFactory(const Board*, const GameStatus*)
+ get_move(BoardCoor): PossibleMovement*
- _can_castle_short: BoolPair
- _can_castle_long: BoolPair
- _king_pos: CoorPair
- _last_double_pawn_move: BoardCoor
MoveFactoryBase
+ MoveFactoryBase(const Board*, const GameStatus*)
+ virtual ~MoveFactoryBase()
+ virtual get_move(BoardCoor): PossibleMovement* = 0
- _p_board: const Board*
- _p_status: const GameStatus*
Board
- notify_display(i32, i32): void
- _board: RawBoard
+ Board()
+ get(BoardCoor): OptPiece
+ get(i32, i32): OptPiece
+ set(BoardCoor, OptPiece): Board&
+ set(i32, i32, OptPiece): Board&
+ cbegin(): column_citer
+ cend(): column_citer
+ board_range(): joined_range
BoardSubject
- _p_observer: BoardObserver*
+ BoardSubject()
+ virtual ~BoardSubject()
+ attach(ptr_view<BoardObserver>)
# get_observer(): BoardObserver&
BoardObserver
+ virtual BoardObserver()
+ virtual ~BoardObserver()
+ virtual get_notified(BoardCoor) = 0
RawMove
- from: BoardCoor
- to: BoardCoor
- promotion: std::optional<E_PieceType>
E_UniqueAction
+ None: u8
+ ShortCastle: u8
+ LongCastle: u8
+ EnPassant: u8
+ DoublePawnPush: u8
+ Promote: u8
PieceMove
- coor: BoardCoor
- unique_action: E_UniqueAction
PossibleMovement
- moves: std::vector<PieceMove>
- captures: std::vector<PieceMove>
- protects: std::vector<PieceMove>
+ unmoveable(): bool
ComputerLv3
+ ~ComputerLv3(): void
+ get_move(): RawMove
ComputerLv4
+ ~ComputerLv4(): void
+ get_move(): RawMove
BoardOperatorBase
+ BoardOperatorBase(Board*)
+ virtual ~BoardOperatorBase()
+ virtual execute_move(BoardCoor, PieceMove): bool = 0
- _p_board: Board*
StandardBoardOperator
+ attribute1:type = defaultValue
+ attribute2:type
- attribute3:type
+ StandardBoardOperator(Board*)
+ ~StandardBoardOperator() override
+ execute_move(BoardCoor, PieceMove): bool override
PlayerBase
+ PlayerBase(ChessGame*): void
+ virtual ~PlayerBase(): void
+ play_move(): bool
# virtual get_move(): RawMove = 0
- p_game: ChessGame*
LocalPlayer
+ get_move(): RawMove
Computer
+ virtual ~Computer(): void = 0
+ get_move(): RawMove
ComputerLv1
+ ~ComputerLv1(): void
+ get_move(): RawMove
ComputerLv2
+ ~ComputerLv2(): void
+ get_move(): RawMove