Wildfrost Remake Manual

This manual explains all custom functions included in the project as well as the purpose of each object. You can also find a tutorial on how to add new cards to the game below.

Adding a New Card

Add the card ID to the enum

- Open constants.gml and find the enum CardID.
- Add your new card ID to this list. Place it anywhere above “Size”
    - Follow this naming convention:
        - TYPE_MyName
    - Types include:
        - CO = Commander
        - ME = Mercenary
        - SP = Spell

Define the card’s data

- Open init_card_data() in card_functions.gml.
- Add a new entry in global.card_data with the new unique ID.
- Define custom stats including keywords, hp, attack and time.
- Make sure to update the type and subtype depending on the card.
- For spells: Don’t forget to add the “effect” attribute.
    - Define special behavior in scripts if the card uses custom effects.

Unit Example

```
global.card_data[CardID.ME_JudyHops] =
{
      name: "Judy Hops",
      type: CardType.Mercenary,
      subtype: UnitType.Unit,
      sprite: sJudyHopsUnit,
      keywords: [],

      hp: 20,
      attack: 8,
      time: 5
};
```

Spell Example

```
global.card_data[CardID.SP_ThrowingAxe] =
{
	name: "Throwing Axe",
	type: CardType.Spell,
	subtype: SpellType.Damage,
	effect: spell_attack,
	sprite: sThrowingAxeSpell,
	keywords: [],

	attack: 7
};
```

Functions

Card Functions

init_card_data
populates global.card_data with all card definitions (units, spells, stats, keywords).

get_cards_by_type(_type, _subtype = undefined)
returns an array of card IDs matching type (and optional subtype).

get_cards_by_keyword(_keyword)
returns an array of card IDs that have the keyword.

has_stat(_stat)
true if a stat exists (not -1).

has_keyword(_inst, _keyword)
true if the card on this instance has the keyword.

array_index_of(_arr, _val)
returns index of _val in _arr or -1.

add_keyword_to_card(_id, _keyword)
adds keyword to card definition if not already present.

remove_keyword_from_card(_id, _keyword)
removes keyword from card definition if present.

create_stats(_hp, _atk, _time)
returns a { hp, attack, time } struct.

create_card(_id, _team = Team.Player, _x = 0, _y = 0)
creates an oCard instance for given card ID/team and sets its position.

remove_card_from_hand(_inst)
removes a card instance from global.current_hand and reflows hand.

play_unit_card(_inst, _slot)
places a unit card onto the grid at _slot and removes it from hand.

play_spell_card(_inst)
plays a spell card: applies its effect (if defined) and usually removes from hand.

Deck Functions

build_starting_hand()
fills the opening hand (e.g., using specific keywords/ids).

build_deck()
constructs global.deck from defined card pools.

clear_hand()
clears global.current_hand.

reposition_cards()
re-lays out cards in the player's hand.

draw_card(_amount, _deck)
draws _amount from _deck into hand (with spacing/layout updates).

Draw Functions

draw_set_settings(_ha, _va, _col, _fnt)
sets text/font/align/color for subsequent draws.

draw_stats(box_w = 20, box_h = 28, _offset = 4)
draws the card's HP/ATK/time stat boxes.

draw_spell_arrow()
draws the targeting arrow/line for spells while aiming.

Game Functions

start_battle()
bootstraps battle state and UI (sets up grid/hand/wave, etc.).

start_combat(_team)
starts a turn for _team and triggers attacks/flow as needed.

end_turn()
ends current team's turn and advances flow (e.g., to the other team or next phase).

victory()
triggers win sequence/UI/state.

fail()
triggers loss sequence/UI/state.

Grid Functions

slot_index(_team, _row, _col)
returns linear index for a team's grid slot.

get_slot_inst(_team, _row, _col)
gets the oSlot instance at that position (or noone).

move_unit_to_slot(_unit, _slot)
moves a unit instance to _slot and updates occupancy.

grid_place(_unit, _team, _row, _col)
places a unit at a grid coordinate (sets slot occupied, unit pos).

grid_remove(_unit)
clears the unit from its slot and marks slot unoccupied.

find_frontmost(_team)
returns the index or slot of the frontmost occupied slot for _team.

shift_left_from(_team, _fromIndex)
shifts units toward the front from a given index.

find_gap_indices(_team)
returns indices of empty slots.

slot_has_unit(_team, _row, _col)
true if slot has a unit.

get_unit_in_slot(_team, _row, _col)
returns unit instance at slot or noone.

advance_unit_toward_front(_team, _index)
tries to slide a unit forward to close gaps.

team_has_movable_gap(_team)
true if there's at least one gap a unit can slide into.

fill_all_gaps(_team)
repeatedly slides units forward until no gaps are movable.

Spell Functions

spell_attack()
applies the current spell's attack to spell_target (reads damage from the card's data).

spell_heal()
heals spell_target based on the card's data.

UI Functions

card_hover_effect(_normal_scale, _lerp_amt)
handles hover scaling/animation for cards.

start_drag()
begins dragging the hovered card (sets flags/drag ref).

update_drag()
updates drag position/preview and aim behavior (for spells).

stop_drag()
releases the card; plays/places it if valid, otherwise cancels/returns to hand.

Unit Functions

unit_attack(_target, _attacker)
applies attacker's damage to target (with logging/FX).

kill_unit(_inst)
kills a unit instance, logs event, frees the slot, and triggers gap-filling.

Wave Functions

start_next_wave()
increments the wave counter and kicks off the next wave.

spawn_enemy(_enemy)
creates an enemy unit for the wave and places it on the grid.

Objects

Managers

oGameManager
central controller for the game flow. sets up globals, manages turns, waves, and transitions between states (menu → battle → victory/defeat).

oGridManager
manages the battle grid. responsible for creating slots, placing units, and shifting units forward to fill gaps.

oWaveManager
tracks wave progression, spawns enemies, and advances to the next wave after conditions are met.

oMenuManager
title/menu button controller. handles transitions between menu and battle.

oConsoleManager
developer/debug console. allows running functions/commands at runtime (spawning units, ending turns, etc.).

UI

oCard
represents a card in the player's hand or deck. handles hover effects, dragging, playing units/spells, and displaying stats.

oMenuPlay / oMenuExit

basic buttons for menu navigation. starts and ends the game from the main menu.

oMapIcon

placeholder for a map/progression system. represents a battle location the player can click to start a fight.

oBattleStart

allows the player to end the deployment phase and start the battle.