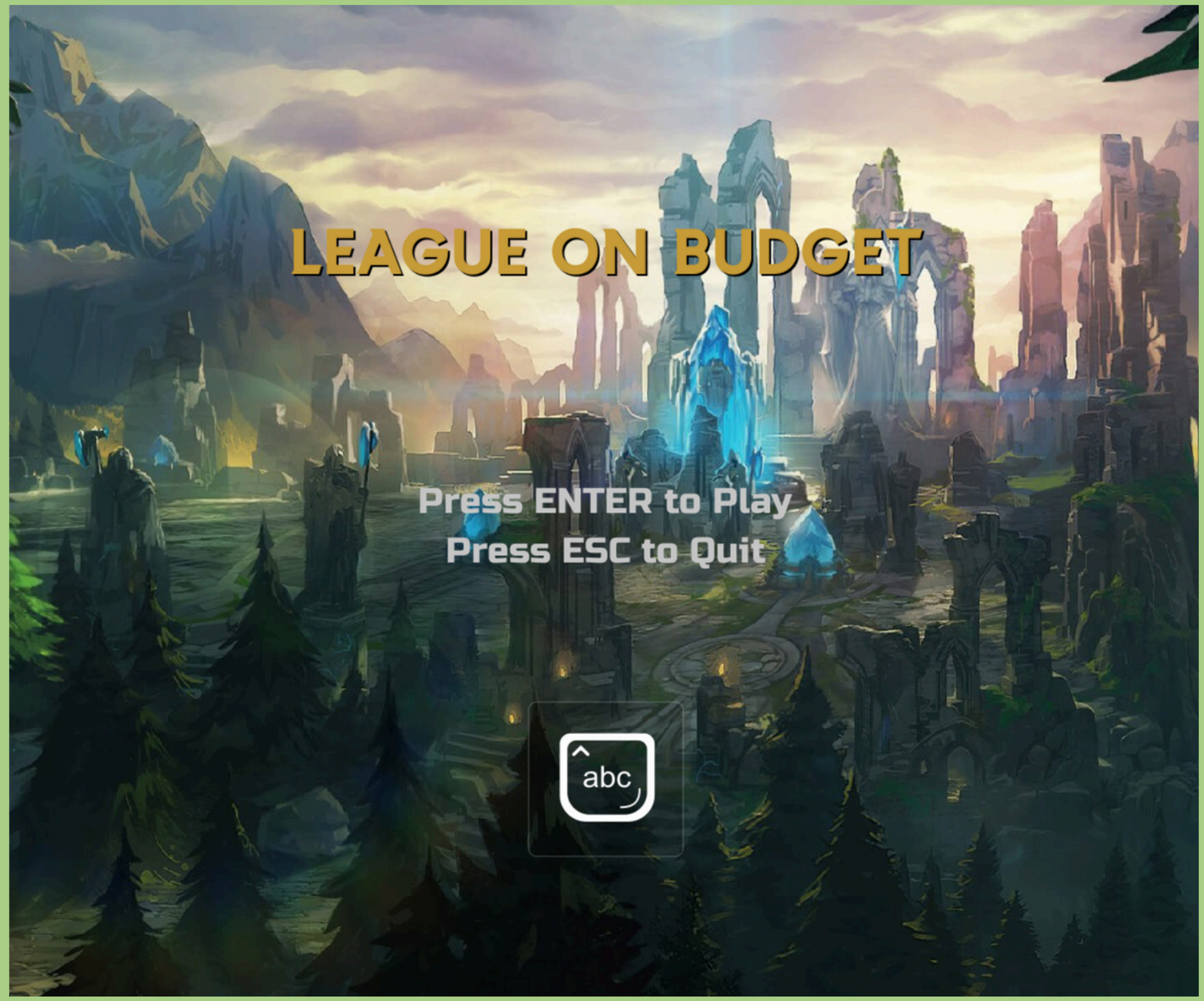
LEAGUE ON BUDGET

OOP project

Yasser BOUHAI
Ghozlene HANAFI
Aya CHELFAT

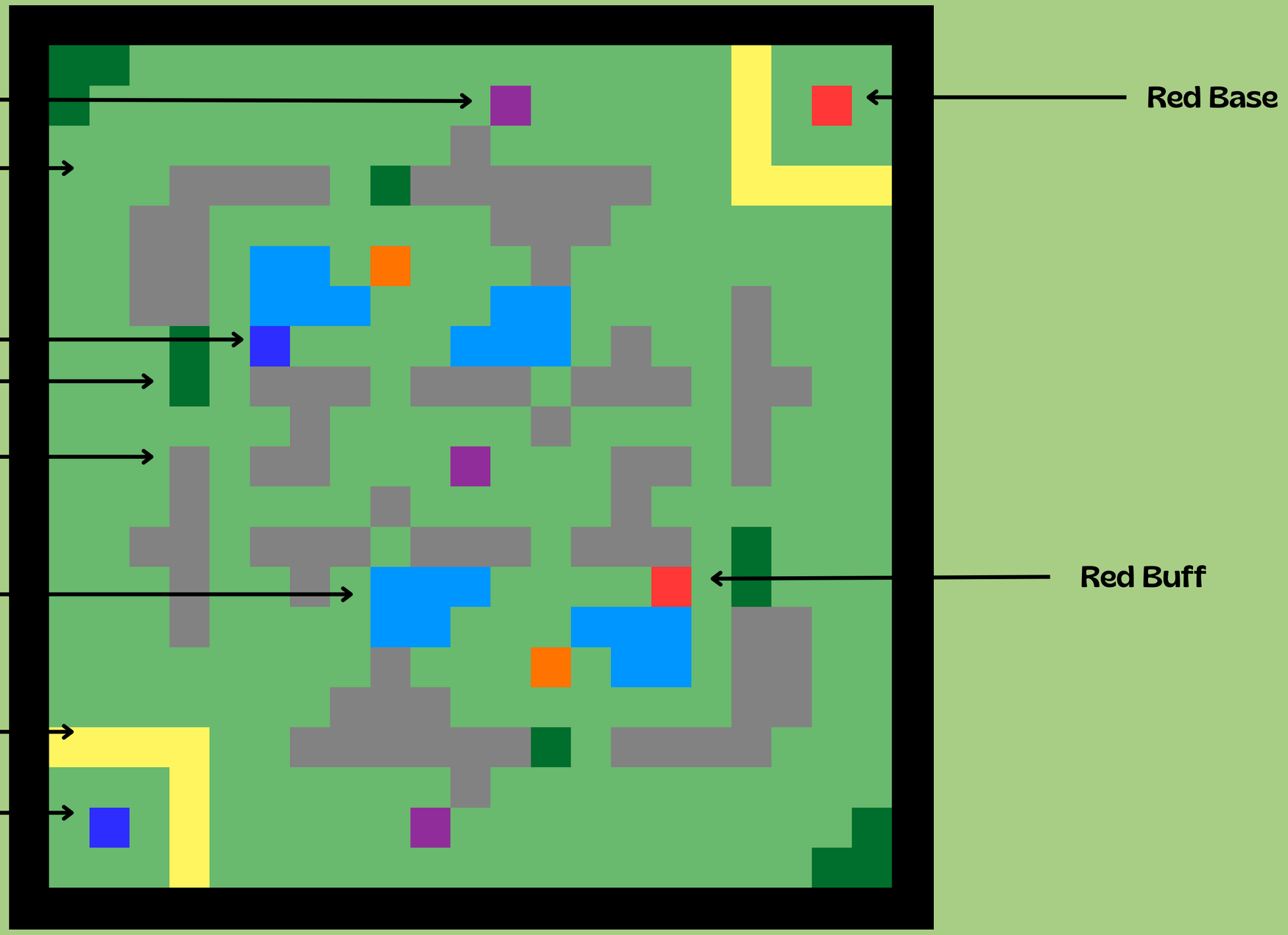
Introduction

Menu Principal

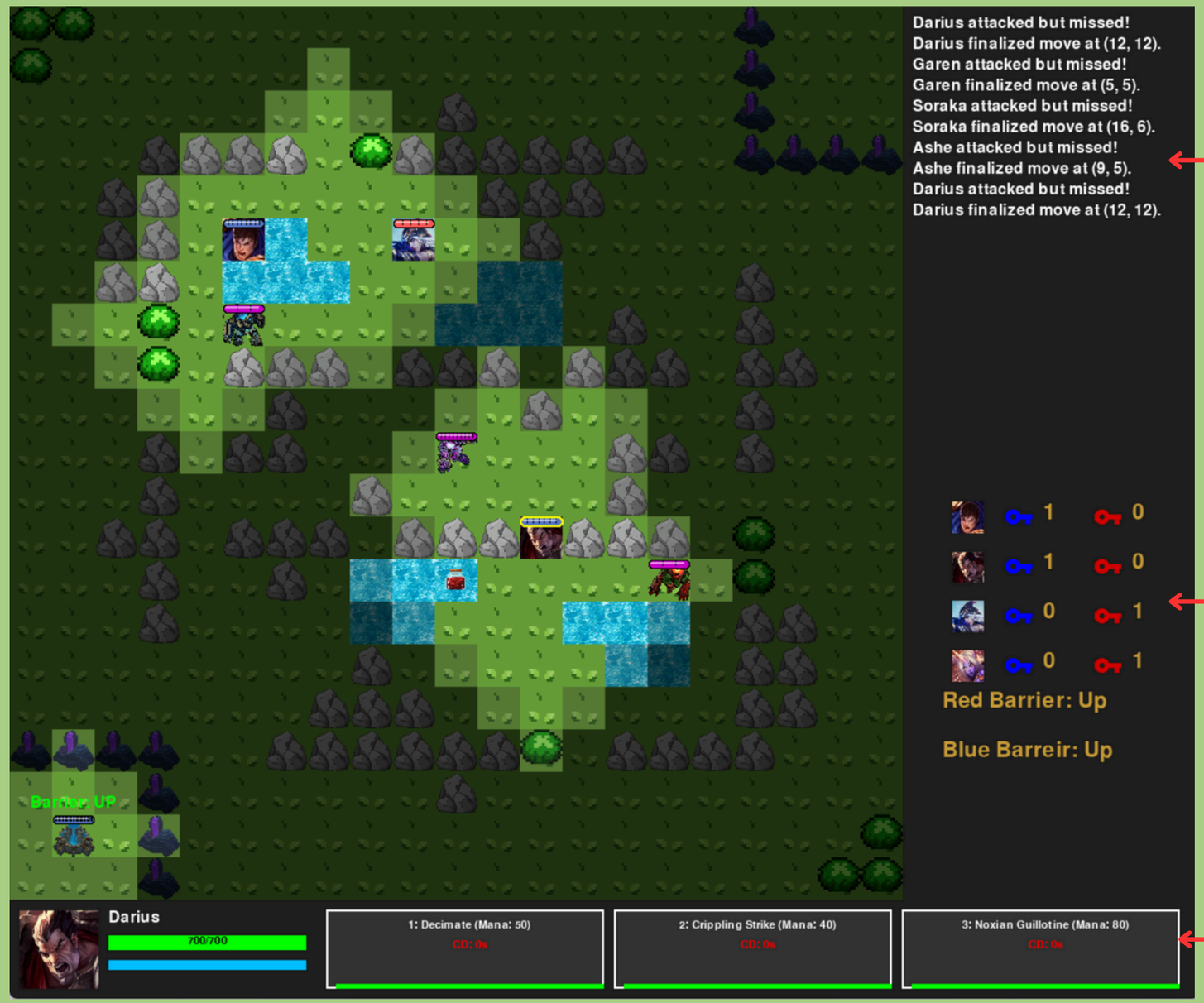


Choix des Joueurs

La Grille


BigBuff
Grass
Blue Buff
Bushes
Rocks
Water
Barriers
Blue Base
Red Base
Red Buff

Interface:



info panel

keys and barrier status

Player status and abilities

Systeme de mouvement

Systeme d’attaque

types d’abilities et recompenses:

degats physique

degats magique

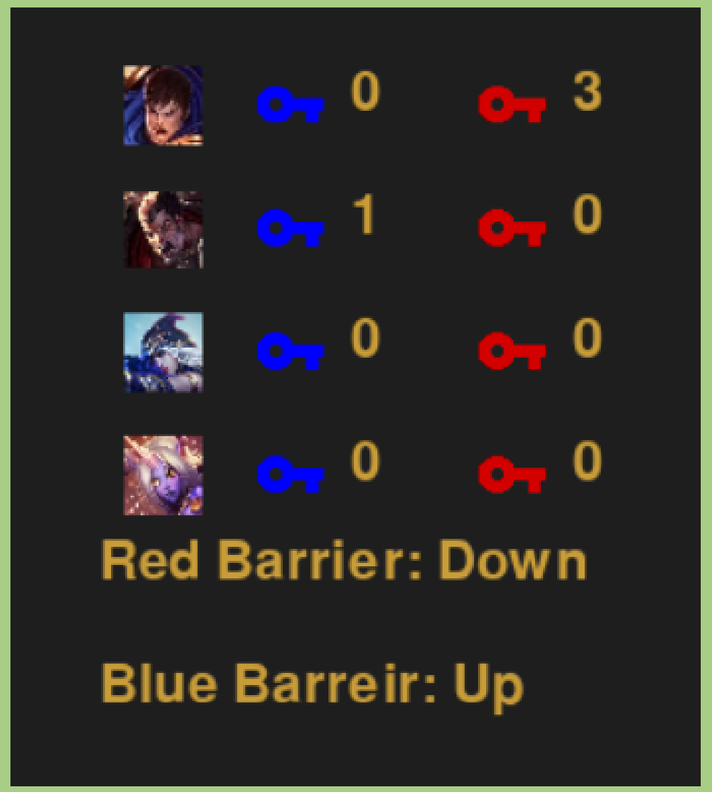
soin

augmentation des stats

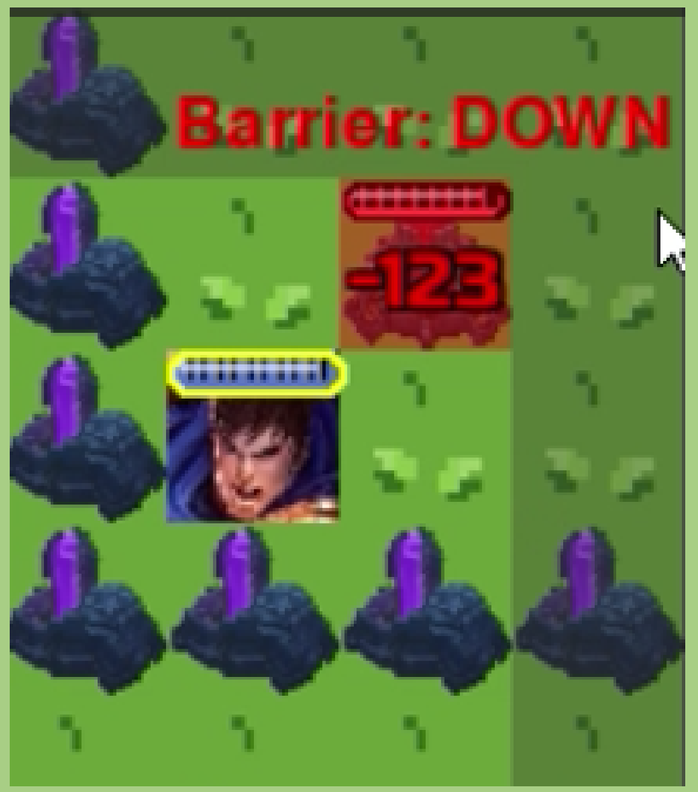
diminution de stats

systeme des clés et comment gagner

debut du jeu



vers la fin du jeu



attaque sur la base enemi

Game Over

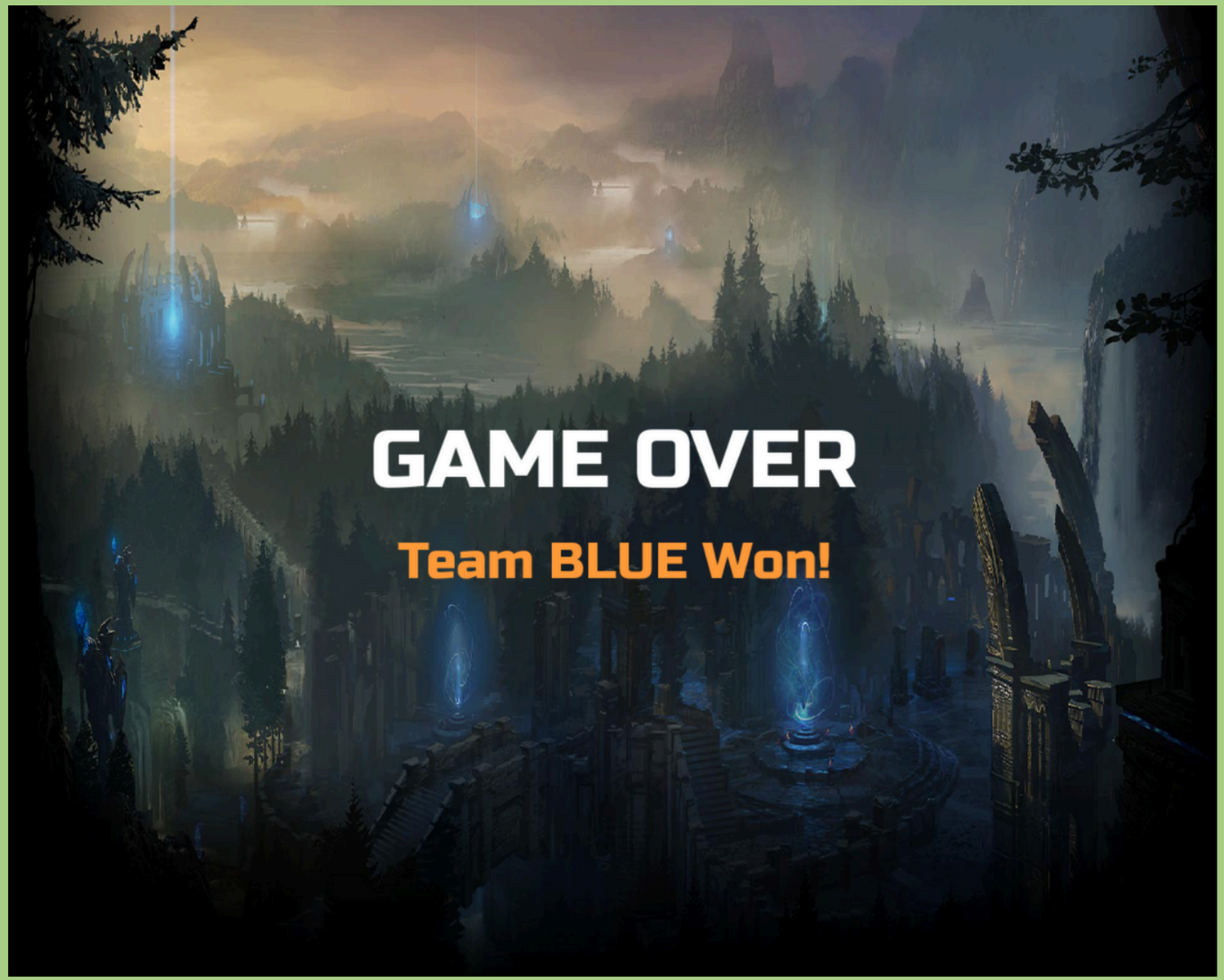
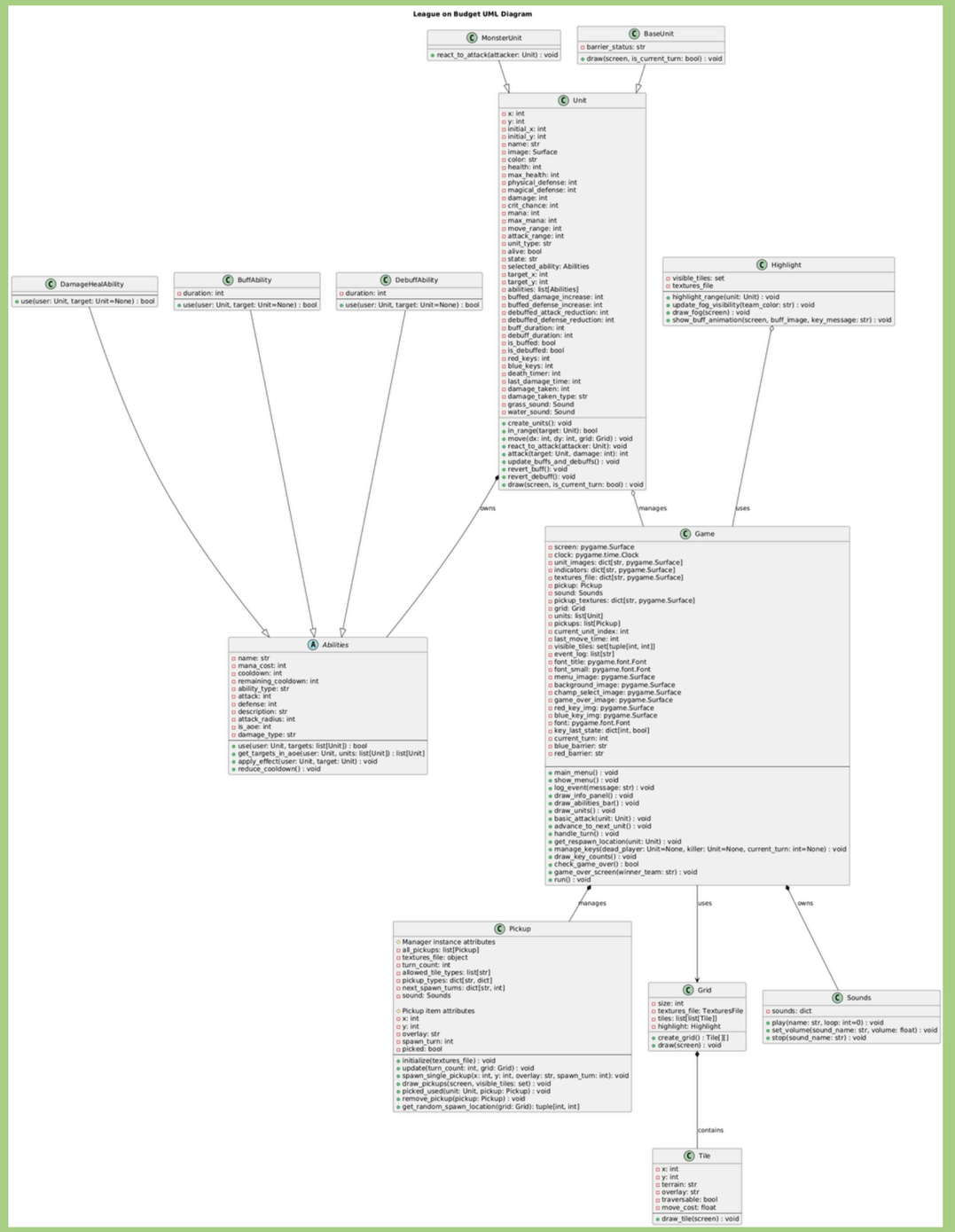


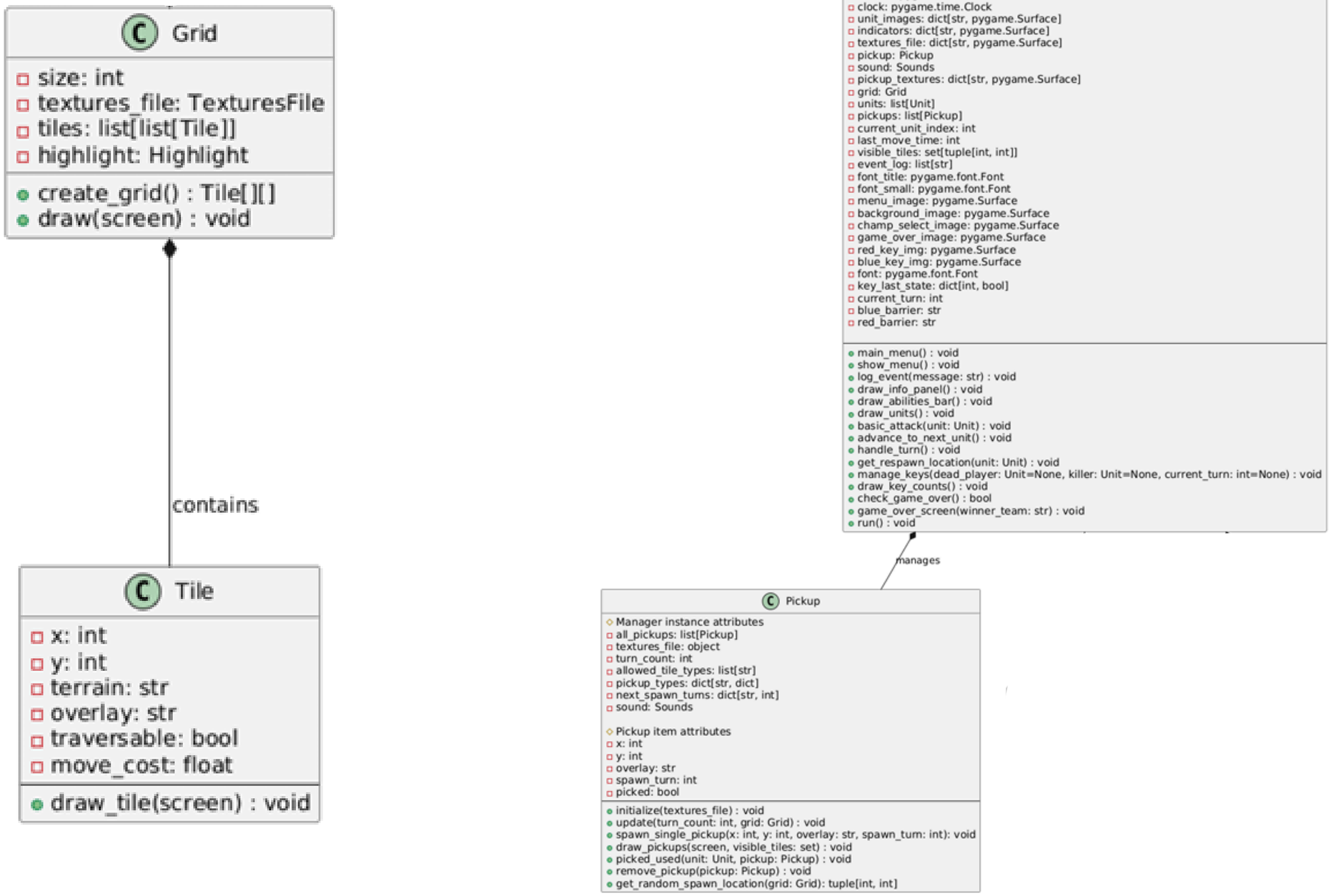
Diagramme UML

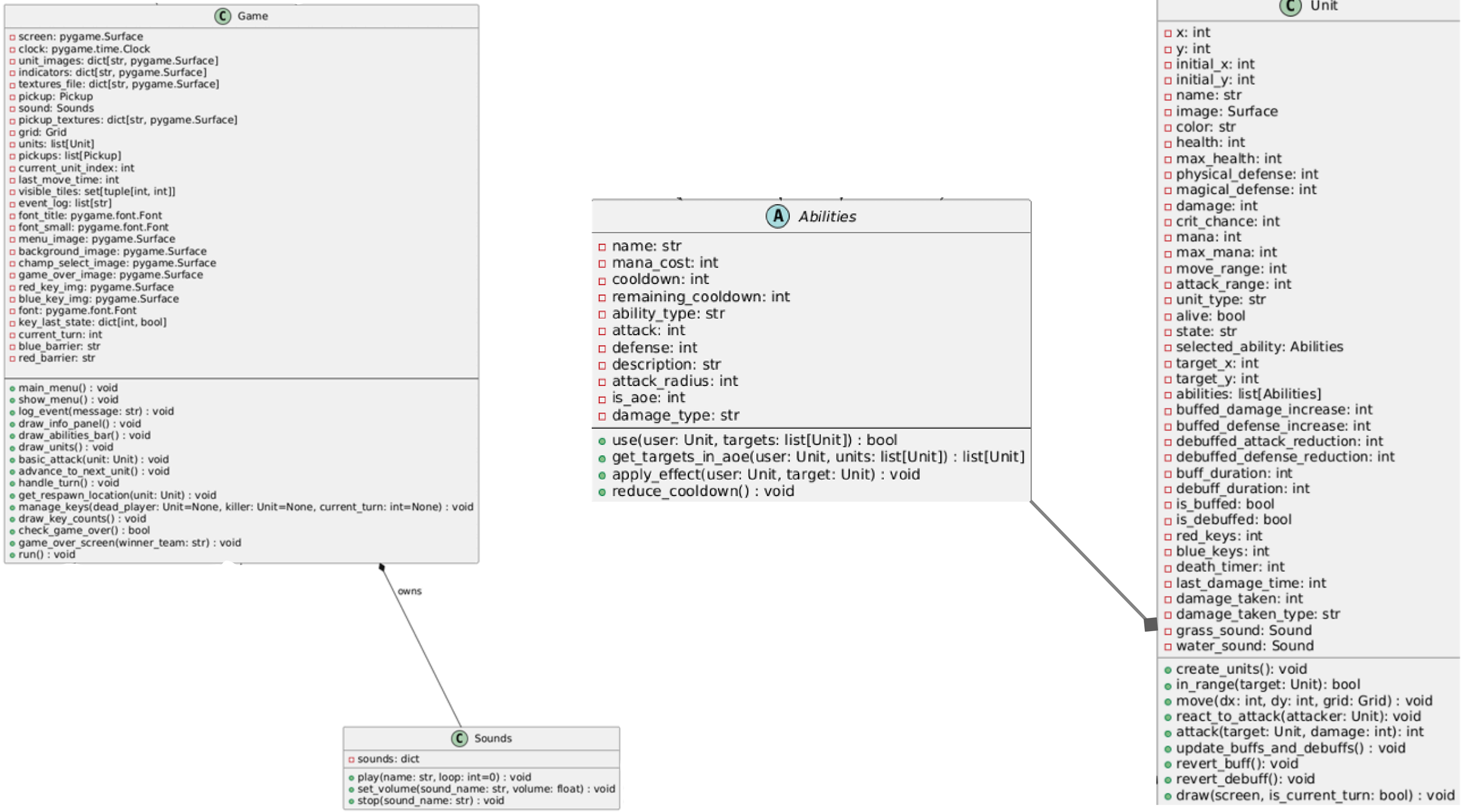


Relation d’héritage

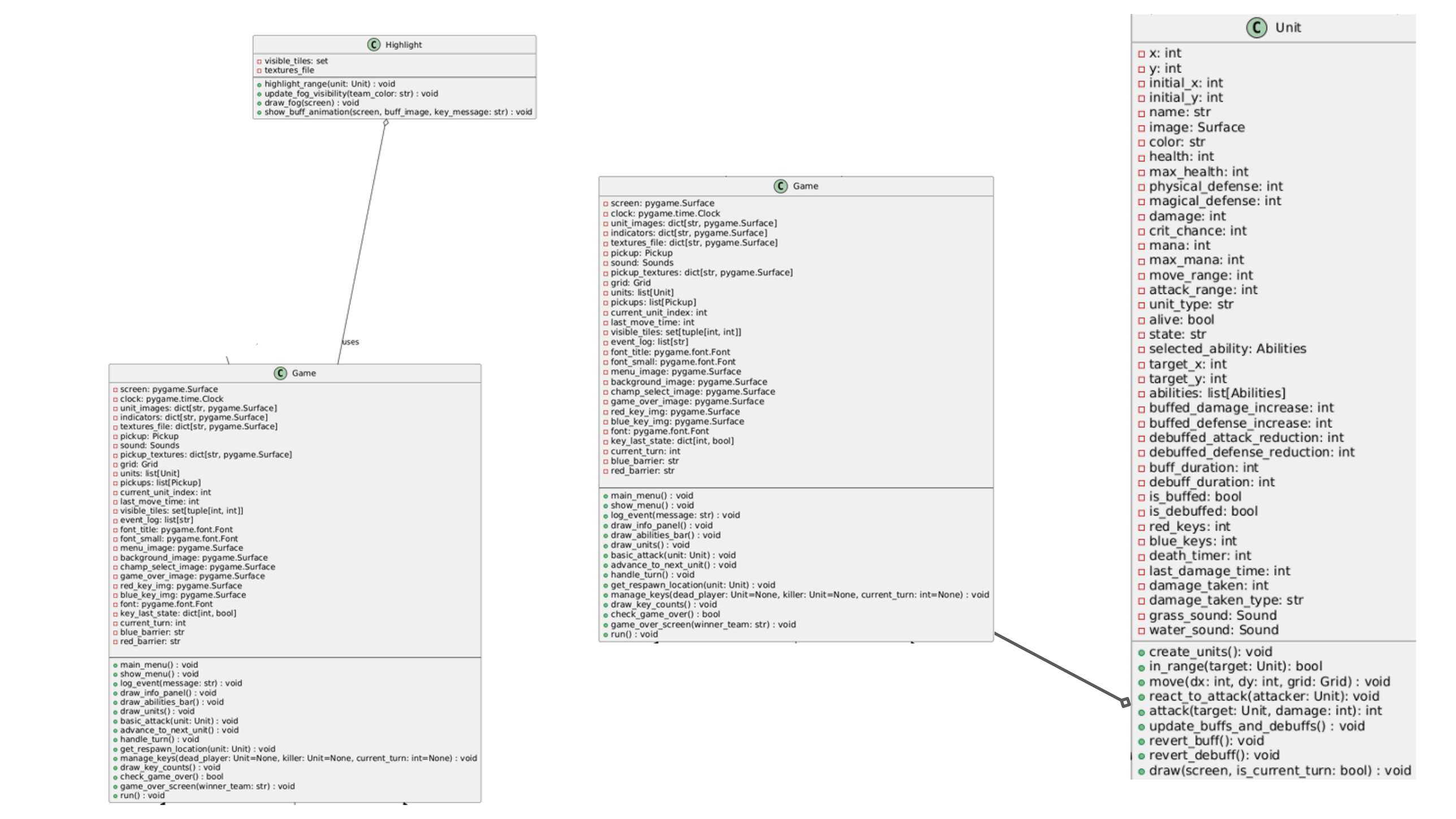

MonsterUnit
react_to_attack(attacker: Unit) : void
BaseUnit
barrier_status: str
draw(screen, is_current_turn: bool) : void
Unit
x: int
y: int
initial_x: int
initial_y: int
name: str
image: Surface
color: str
health: int
max_health: int
physical_defense: int
magical_defense: int
damage: int
crit_chance: int
mana: int
max_mana: int
move_range: int
attack_range: int
unit_type: str
alive: bool
state: str
selected_ability: Abilities
target_x: int
target_y: int
abilities: list[Abilities]
buffed_damage_increase: int
buffed_defense_increase: int
debuffed_attack_reduction: int
debuffed_defense_reduction: int
buff_duration: int
debuff_duration: int
is_buffed: bool
is_debuffed: bool
red_keys: int
blue_keys: int
death_timer: int
last_damage_time: int
damage_taken: int
damage_taken_type: str
grass_sound: Sound
water_sound: Sound
create_units(): void
in_range(target: Unit): bool
move(dx: int, dy: int, grid: Grid) : void
react_to_attack(attacker: Unit): void
attack(target: Unit, damage: int): int
update_buffs_and_debuffs() : void
revert_buff(): void
revert_debuff(): void
draw(screen, is_current_turn: bool) : void
DamageHealAbility
use(user: Unit, target: Unit=None) : bool
BuffAbility
duration: int
use(user: Unit, target: Unit=None) : bool
DebuffAbility
duration: int
use(user: Unit, target: Unit=None) : bool
Abilities
name: str
mana_cost: int
cooldown: int
remaining_cooldown: int
ability_type: str
attack: int
defense: int
description: str
attack_radius: int
is_aoe: int
damage_type: str
use(user: Unit, targets: list[Unit]) : bool
get_targets_in_aoe(user: Unit, units: list[Unit]) : list[Unit]
apply_effect(user: Unit, target: Unit) : void
reduce_cooldown() : void

Relation de composition


Grid
size: int
textures_file: TexturesFile
tiles: list[list[Tile]]
highlight: Highlight
create_grid() : Tile[][]
draw(screen) : void
contains
Tile
x: int
y: int
terrain: str
overlay: str
traversable: bool
move_cost: float
draw_tile(screen) : void
Game
screen: pygame.Surface
clock: pygame.time.Clock
unit_images: dict[str, pygame.Surface]
indicators: dict[str, pygame.Surface]
textures_file: dict[str, pygame.Surface]
pickup: Pickup
sound: Sounds
pickup_textures: dict[str, pygame.Surface]
grid: Grid
units: list[Unit]
pickups: list[Pickup]
current_unit_index: int
last_move_time: int
visible_tiles: set[tuple[int, int]]
event_log: list[str]
font_title: pygame.font.Font
font_small: pygame.font.Font
menu_image: pygame.Surface
background_image: pygame.Surface
champ_select_image: pygame.Surface
game_over_image: pygame.Surface
red_key_img: pygame.Surface
blue_key_img: pygame.Surface
font: pygame.font.Font
key_last_state: dict[int, bool]
current_turn: int
blue_barrier: str
red_barrier: str
main_menu() : void
show_menu() : void
log_event(message: str) : void
draw_info_panel() : void
draw_abilities_bar() : void
draw_units() : void
basic_attack(unit: Unit) : void
advance_to_next_unit() : void
handle_turn() : void
get_respawn_location(unit: Unit) : void
manage_keys(dead_player: Unit=None, killer: Unit=None, current_turn: int=None) : void
draw_key_counts() : void
check_game_over() : bool
game_over_screen(winner_team: str) : void
run() : void
manages
Pickup
Manager instance attributes
all_pickups: list[Pickup]
textures_file: object
turn_count: int
allowed_tile_types: list[str]
pickup_types: dict[str, dict]
next_spawn_turns: dict[str, int]
sound: Sounds
Pickup item attributes
x: int
y: int
overlay: str
spawn_turn: int
picked: bool
initialize(textures_file) : void
update(turn_count: int, grid: Grid) : void
spawn_single_pickup(x: int, y: int, overlay: str, spawn_turn: int): void
draw_pickups(screen, visible_tiles: set) : void
picked_used(unit: Unit, pickup: Pickup) : void
remove_pickup(pickup: Pickup) : void
get_random_spawn_location(grid: Grid): tuple[int, int]


Game
screen: pygame.Surface
clock: pygame.time.Clock
unit_images: dict[str, pygame.Surface]
indicators: dict[str, pygame.Surface]
textures_file: dict[str, pygame.Surface]
pickup: Pickup
sound: Sounds
pickup_textures: dict[str, pygame.Surface]
grid: Grid
units: list[Unit]
pickups: list[Pickup]
current_unit_index: int
last_move_time: int
visible_tiles: set[tuple[int, int]]
event_log: list[str]
font_title: pygame.font.Font
font_small: pygame.font.Font
menu_image: pygame.Surface
background_image: pygame.Surface
champ_select_image: pygame.Surface
game_over_image: pygame.Surface
red_key_img: pygame.Surface
blue_key_img: pygame.Surface
font: pygame.font.Font
key_last_state: dict[int, bool]
current_turn: int
blue_barrier: str
red_barrier: str
main_menu() : void
show_menu() : void
log_event(message: str) : void
draw_info_panel() : void
draw_abilities_bar() : void
draw_units() : void
basic_attack(unit: Unit) : void
advance_to_next_unit() : void
handle_turn() : void
get_respawn_location(unit: Unit) : void
manage_keys(dead_player: Unit=None, killer: Unit=None, current_turn: int=None) : void
draw_key_counts() : void
check_game_over() : bool
game_over_screen(winner_team: str) : void
run() : void
owns
Sounds
sounds: dict
play(name: str, loop: int=0) : void
set_volume(sound_name: str, volume: float) : void
stop(sound_name: str) : void
Abilities
name: str
mana_cost: int
cooldown: int
remaining_cooldown: int
ability_type: str
attack: int
defense: int
description: str
attack_radius: int
is_aoe: int
damage_type: str
use(user: Unit, targets: list[Unit]) : bool
get_targets_in_aoe(user: Unit, units: list[Unit]) : list[Unit]
apply_effect(user: Unit, target: Unit) : void
reduce_cooldown() : void
Unit
x: int
y: int
initial_x: int
initial_y: int
name: str
image: Surface
color: str
health: int
max_health: int
physical_defense: int
magical_defense: int
damage: int
crit_chance: int
mana: int
max_mana: int
move_range: int
attack_range: int
unit_type: str
alive: bool
state: str
selected_ability: Abilities
target_x: int
target_y: int
abilities: list[Abilities]
buffed_damage_increase: int
buffed_defense_increase: int
debuffed_attack_reduction: int
debuffed_defense_reduction: int
buff_duration: int
debuff_duration: int
is_buffed: bool
is_debuffed: bool
red_keys: int
blue_keys: int
death_timer: int
last_damage_time: int
damage_taken: int
damage_taken_type: str
grass_sound: Sound
water_sound: Sound
create_units(): void
in_range(target: Unit): bool
move(dx: int, dy: int, grid: Grid) : void
react_to_attack(attacker: Unit): void
attack(target: Unit, damage: int): int
update_buffs_and_debuffs() : void
revert_buff(): void
revert_debuff(): void
draw(screen, is_current_turn: bool) : void

Relation d’agrégation


Highlight
visible_tiles: set
textures_file
highlight_range(unit: Unit) : void
update_fog_visibility(team_color: str) : void
draw_fog(screen) : void
show_buff_animation(screen, buff_image, key_message: str) : void
uses
Game
screen: pygame.Surface
clock: pygame.time.Clock
unit_images: dict[str, pygame.Surface]
indicators: dict[str, pygame.Surface]
textures_file: dict[str, pygame.Surface]
pickup: Pickup
sound: Sounds
pickup_textures: dict[str, pygame.Surface]
grid: Grid
units: list[Unit]
pickups: list[Pickup]
current_unit_index: int
last_move_time: int
visible_tiles: set[tuple[int, int]]
event_log: list[str]
font_title: pygame.font.Font
font_small: pygame.font.Font
menu_image: pygame.Surface
background_image: pygame.Surface
champ_select_image: pygame.Surface
game_over_image: pygame.Surface
red_key_img: pygame.Surface
blue_key_img: pygame.Surface
font: pygame.font.Font
key_last_state: dict[int, bool]
current_turn: int
blue_barrier: str
red_barrier: str
main_menu() : void
show_menu() : void
log_event(message: str) : void
draw_info_panel() : void
draw_abilities_bar() : void
draw_units() : void
basic_attack(unit: Unit) : void
advance_to_next_unit() : void
handle_turn() : void
get_respawn_location(unit: Unit) : void
manage_keys(dead_player: Unit=None, killer: Unit=None, current_turn: int=None) : void
draw_key_counts() : void
check_game_over() : bool
game_over_screen(winner_team: str) : void
run() : void
Game
screen: pygame.Surface
clock: pygame.time.Clock
unit_images: dict[str, pygame.Surface]
indicators: dict[str, pygame.Surface]
textures_file: dict[str, pygame.Surface]
pickup: Pickup
sound: Sounds
pickup_textures: dict[str, pygame.Surface]
grid: Grid
units: list[Unit]
pickups: list[Pickup]
current_unit_index: int
last_move_time: int
visible_tiles: set[tuple[int, int]]
event_log: list[str]
font_title: pygame.font.Font
font_small: pygame.font.Font
menu_image: pygame.Surface
background_image: pygame.Surface
champ_select_image: pygame.Surface
game_over_image: pygame.Surface
red_key_img: pygame.Surface
blue_key_img: pygame.Surface
font: pygame.font.Font
key_last_state: dict[int, bool]
current_turn: int
blue_barrier: str
red_barrier: str
main_menu() : void
show_menu() : void
log_event(message: str) : void
draw_info_panel() : void
draw_abilities_bar() : void
draw_units() : void
basic_attack(unit: Unit) : void
advance_to_next_unit() : void
handle_turn() : void
get_respawn_location(unit: Unit) : void
manage_keys(dead_player: Unit=None, killer: Unit=None, current_turn: int=None) : void
draw_key_counts() : void
check_game_over() : bool
game_over_screen(winner_team: str) : void
run() : void
Unit
x: int
y: int
initial_x: int
initial_y: int
name: str
image: Surface
color: str
health: int
max_health: int
physical_defense: int
magical_defense: int
damage: int
crit_chance: int
mana: int
max_mana: int
move_range: int
attack_range: int
unit_type: str
alive: bool
state: str
selected_ability: Abilities
target_x: int
target_y: int
abilities: list[Abilities]
buffed_damage_increase: int
buffed_defense_increase: int
debuffed_attack_reduction: int
debuffed_defense_reduction: int
buff_duration: int
debuff_duration: int
is_buffed: bool
is_debuffed: bool
red_keys: int
blue_keys: int
death_timer: int
last_damage_time: int
damage_taken: int
damage_taken_type: str
grass_sound: Sound
water_sound: Sound
create_units(): void
in_range(target: Unit): bool
move(dx: int, dy: int, grid: Grid) : void
react_to_attack(attacker: Unit): void
attack(target: Unit, damage: int): int
update_buffs_and_debuffs() : void
revert_buff(): void
revert_debuff(): void
draw(screen, is_current_turn: bool) : void

Relation d’association


Game
screen: pygame.Surface
clock: pygame.time.Clock
unit_images: dict[str, pygame.Surface]
indicators: dict[str, pygame.Surface]
textures_file: dict[str, pygame.Surface]
pickup: Pickup
sound: Sounds
pickup_textures: dict[str, pygame.Surface]
grid: Grid
units: list[Unit]
pickups: list[Pickup]
current_unit_index: int
last_move_time: int
visible_tiles: set[tuple[int, int]]
event_log: list[str]
font_title: pygame.font.Font
font_small: pygame.font.Font
menu_image: pygame.Surface
background_image: pygame.Surface
champ_select_image: pygame.Surface
game_over_image: pygame.Surface
red_key_img: pygame.Surface
blue_key_img: pygame.Surface
font: pygame.font.Font
key_last_state: dict[int, bool]
current_turn: int
blue_barrier: str
red_barrier: str
main_menu() : void
show_menu() : void
log_event(message: str) : void
draw_info_panel() : void
draw_abilities_bar() : void
draw_units() : void
basic_attack(unit: Unit) : void
advance_to_next_unit() : void
handle_turn() : void
get_respawn_location(unit: Unit) : void
manage_keys(dead_player: Unit=None, killer: Unit=None, current_turn: int=None) : void
draw_key_counts() : void
check_game_over() : bool
game_over_screen(winner_team: str) : void
run() : void
uses
Grid
size: int
textures_file: TexturesFile
tiles: list[list[Tile]]
highlight: Highlight
create_grid() : Tile[][]
draw(screen) : void


Merci pour votre attention