# Projet Long : Shovel Knight

Par ABDOROHMANG Mathaui Yves et Pariat Simon

# Introduction

But : Implémenter le jeu en OCaml

SHOVEL KNIGHT

# Introduction


# Architecture, conception et gestion de projet

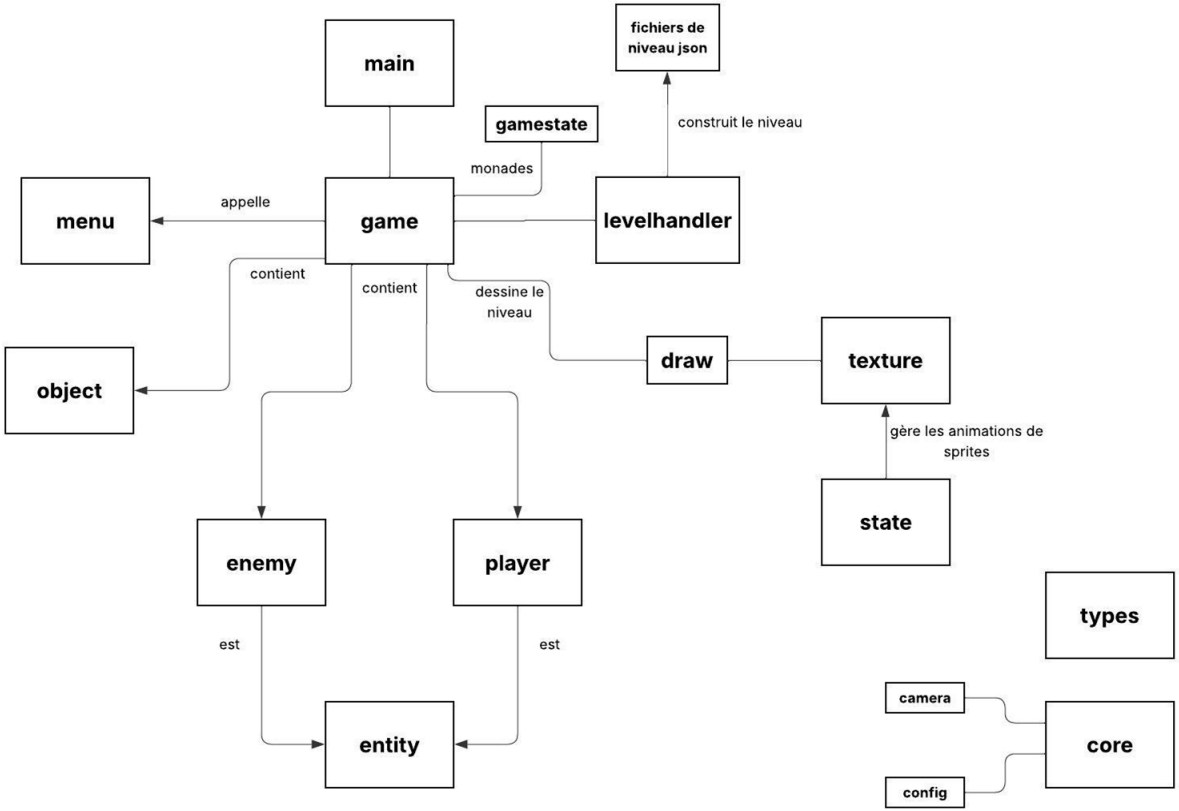
## Notre calendrier prévisionnel des taches à effectuer.

### Calendrier

les dates associées aux étapes sont approximatives et peuvent évoluer

| Etape | Date fin de la tache |
|---|---|
| Réflechir à la structure de notre projet et se refamiliariser avec ocaml | Novembre 2024 |
| Créer les Objets et l'environnement et définir leurs propriétés : le joueur, les plateformes, les objets/bonus, les ennemies. Structurer le modèle. Commencer à travailler la vue | fin du semestre 1 |
| Créer les interactions entre objets : comment le joueur interagit avec les plateformes, les sauts du joueur, avoir une vue qui marche avec une bonne gestion de la caméra, les sprites du joueur et de la map, une première plateforme | fin janvier/ début février |
| Implémenter différent types d'objets, d'ennemies et de blocs (exemples : échelles), en les testant sur des map vides. Créer le système d'attaque du jeu (attaque sauté, corps à corps etc ...). | fin février |
| Implémenter des animations (courir, sauté, attaquer) pour le joueur et les ennemies. Gérer les hitbox et les knockbacks pour les attaques | mi-mars |
| implémenter un premier niveau simple. Stylisé le jeu et l'interface de jeu (ajout de son, de nouvelles animations, des menus de pauses ...) | début avril |
| Selon le temps disponible et l'avancement des objectifs précédents : implémenter un système de score(temps à finir le niveau, golds récupérés ...), créer d'autres niveaux peut être plus difficile, avec potentielement un combat de boss. Implémenter les objectifs supplémentaire (si le temps le permet) | fin avril/ début mai |

# Architecture, conception et gestion de projet

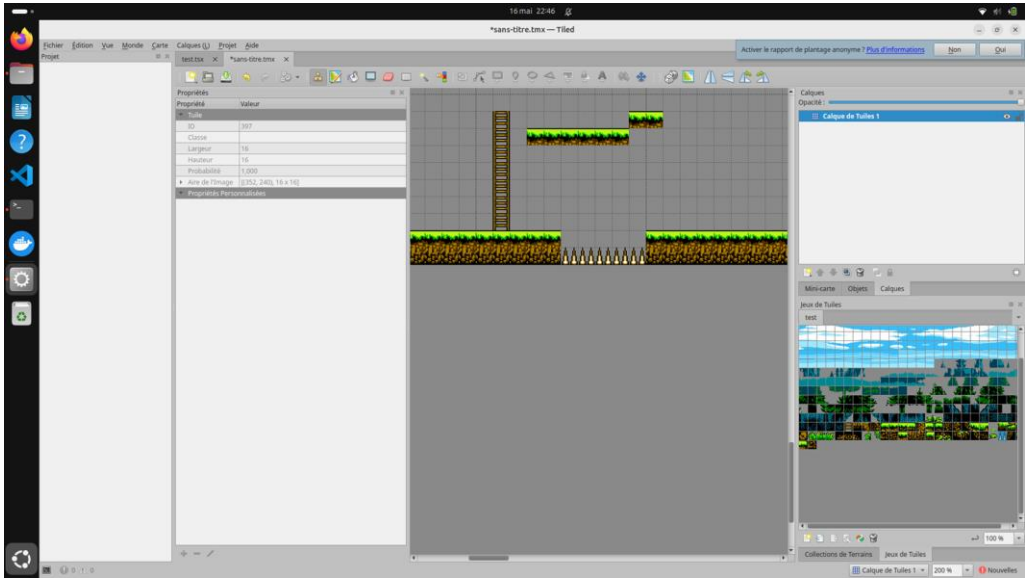



```
home > simon > Documents > projetlong > abdorohmang-pariat-pprog-2024 > shovelknight > lib > enemy.mli
1  (** Ennemis du jeu *)
2
3
4  (** [initialize_enemy skin at mv h mh pos hw hh] crée un nouvel ennemi avec son skin, type d'attaque, type de mouvement
5      pv, pv_max, hauteur de l'hitbox ainsi que la longueur *)
6  val initialize_enemy :
7    Types.enemy_skin ->
8    Types.attack_type ->
9    Types.enemy_movement -> int -> int -> Types.position -> float -> float -> Types.game_entity
10
11 (** [create_beeto x y] crée un ennemi Beeto *)
12 val create_beeto : float -> float -> Types.game_entity
13
14 (** [create_blorb x y] crée un ennemi Blorb*)
15 val create_blorb : float -> float -> Types.game_entity
16
17 (** [create_boneclang x y] crée un ennemi Boneclang*)
18 val create_boneclang : float -> float -> Types.game_entity
19
20 (** [create_dozedrake x y] crée un ennemi Dozedrake*)
21 val create_dozedrake : float -> float -> Types.game_entity
22
```

Exemple de fichier .mli

# Architecture, conception et gestion de projet

## Tiled pour faire les fichiers .json




## Tests unitaires

```
test_enemies.ml
home > simon > Documents > projetlong > abdorohmang-pariat-pprog-2024 > shovelknight > test > test_enemies.ml
1  open Shovelknight
2  open Enemy
3  open Types
4
5
6  (*let empty_map : Types.block array array =
7      let map = Array.init 3 (fun _ -> Array.make 3 Types.Empty) in
8      Array.iteri (fun i _ -> map.(2).(i) <- Solid Dirt) map.(2);
9      map*)
10
11
12 let check_health name enemy =
13     let { entity_type; _ } = enemy in
14     match entity_type with
15     | Enemy e ->
16         Alcotest.(check bool) (name ^ " current_health > 0") true (e.current_health > 0);
17         Alcotest.(check bool) (name ^ " max_health > 0") true (e.max_health > 0)
18     | _ -> Alcotest.fail (name ^ " is not an Enemy")
19   ;;
20
21 let test_create_beeto () = check_health "Beeto" (create_beeto 0. 0.)
22 let test_create_blorb () = check_health "Blorb" (create_blorb 0. 0.)
23 let test_create_boneclang () = check_health "Boneclang" (create_boneclang 0. 0.)
24 let test_create_dozedrake () = check_health "Dozedrake" (create_dozedrake 0. 0.)
25 let test_create_firedrake () = check_health "Firedrake" (create_firedrake 0. 0.)
26
27 let test_attack_enemy () =
28     let attacker = Player.initialize_player "attacker" in
29     let target = create_beeto 0. 0. in
30     match target.entity_type with
31     | Enemy _ ->
32         let old_health =
33           match attacker.entity_type with
34           | Player p -> p.current_health
35           | _ -> Alcotest.fail "Attacker should be a Player"
36         in
37         let new_attacker = attack_enemy target attacker in
38         (match new_attacker.entity_type with
39          | Player p' ->
40              Alcotest.(check bool) "health decreased" true (p'.current_health < old_health)
41          | _ -> Alcotest.fail "Attacker should still be a Player")
42     | _ -> Alcotest.fail "Target should be an Enemy"
43
44
45 let test_string_conversion () =
46     let open Types in
47     let all_skins = [Beeto; Blorb; Boneclang; Dozedrake; Firedrake] in
48     List.iter (fun skin ->
```

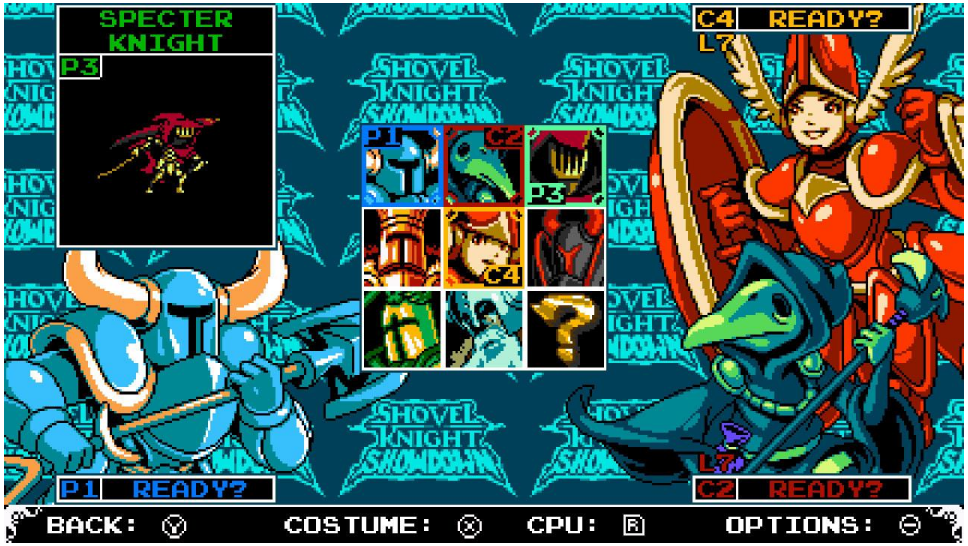
# Code

```
let draw_player player px py vx hitbox config =
  let state = get_sprite_index player in
  let w_factor = if vx < 0. then -1. else 1. in
  let (hw,hh) = Entity.get_hitbox_dims hitbox in
  let source_rect = find_sprite state knight_sprites.textures in
  let new_rect = Rectangle.create (Rectangle.x source_rect) (Rectangle.y source_rect) (Rectangle.width source_rect *. w_factor) (Rectangle.height source_rect) in
  let dest_rect = Rectangle.create (px) py (hw *. config.block_width) (hh *. config.block_height) in
  draw_texture_pro (get_texture knight_sprites) new_rect dest_rect (Vector2.create 0.0 0.0) 0.0 Color.white;
  draw_player_profile_square player


let draw_object obj bx by config =
  let obj_skin = match obj.entity_type with
    | Object o -> (match o.obj_type with
      | Gold t -> t
      | Heal f -> f
      | Mana m -> m
      | Projectile p -> p)
    | _ -> failwith "impossible"
  in
  let source_rect = find_sprite obj_skin object_sprites.textures in
  let dest_rect = Rectangle.create bx by config.block_width config.block_height in
  draw_texture_pro (get_texture object_sprites) source_rect dest_rect (Vector2.create 0.0 0.0) 0.0 Color.white

let draw_moving_platform mp bx by config =
  let (hw, hh) = Entity.get_hitbox_dims mp.hitbox in
  let bw = config.block_width in
  let bh = config.block_height in
  match mp.entity_type with
  | MovingPlatform m ->
      let block = m.skin in
      for i = 0 to int_of_float (hw -. 0.001) do
        for j = 0 to int_of_float (hh -. 0.001) do
          let x = bx +. (float_of_int i *. bw) in
          let y = by +. (float_of_int j *. bh) in
          draw_block block x y config
        done
      done
  | _ -> ()
```

# Conclusion





(images tirées du vrai jeu)