Chloé Boucher, Amélia Desgagné et Luca Reymann

Projet synthèse

420-C61-VM / Gr. 0001

Conception du jeu Cartoon Quest

Travail présenté à

Jean-Christophe Demers

Département d’informatique

Cégep du Vieux Montréal

3 mars 2022

**Table des matières**

[**PRÉSENTATION GÉNÉRALE**](#_86y025m0i6w9) **2**

[**TECHNOLOGIES UTILISÉES**](#_rwnwanf52mvm) **2**

[**MAQUETTES DE L’INTERFACE GRAPHIQUE**](#_kxyuwhowcqvc) **3**

[PAGE DE CONNEXION](#_1fvn95uwm3j) 3

[DESCRIPTION GÉNÉRALE](#_m447qmfmv9rs) 4

[MENU D’ACCUEIL](#_uir93c9hg74g) 5

[DESCRIPTION GÉNÉRALE](#_j22lfxi4t2j6) 5

[ÉLÉMENTS TECHNIQUES](#_j9szruqthy0g) 6

[ACHIEVEMENTS](#_vj213x7kv9sq) 7

[DESCRIPTION GÉNÉRALE](#_x57wxz6ve6p) 7

[ÉLÉMENTS TECHNIQUES](#_smisb7vkbp5b) 7

[FENÊTRE DE JEU](#_unadqnkjiyzk) 8

[DESCRIPTION GÉNÉRALE](#_jd6ie94lu4dt) 9

[ÉLÉMENTS TECHNIQUES](#_mh9x2n985nol) 9

[**CONCEPTION UML**](#_n35jj5vv4ln7) **10**

[DIAGRAMMES DES CAS D’USAGE](#_90qki2ew1sg) 10

[Usager dans le login](#_r5mcb7og48im) 10

[Usager dans le lobby](#_ht1yl4e9u20e) 10

[Usager dans le jeu](#_l5d0acckufvx) 11

[DIAGRAMMES DE CLASSE](#_nktgaqhl2r70) 12

[**SCHÉMA DE LA STRUCTURE DE DONNÉES EXTERNES**](#_6c4owuerncfv) **13**

[**COMPORTEMENT DES ENNEMIS**](#_jh3l3690hf4t) **14**

[STATES](#_oaapg57hb3f7) 14

[BEHAVIOR](#_fsity0ok1suz) 14

[Movement Behaviour](#_am1xyf36rtcx) 14

[Shoot Behaviour](#_9uil9qbk12pb) 14

[**GÉNÉRATION D’UN NIVEAU - (ALGORITHME)**](#_c5erpamhkbhw) **15**

[**SYSTÈME DE COLLISION - (DATA STRUCTURE)**](#_sj2gure3ssad) **16**

[OPTION 1: ordre n2](#_4l71uwwdliby) 16

[OPTION 2: ordre n x log(n)](#_vy1tu1wkucq1) 16

[OPTION 3: ordre n(1)](#_vjkr7o2nysx8) 16

[**PATRONS DE CONCEPTION**](#_9diidhtzqvid) **17**

[PROTOTYPE (PROTOTYPE):](#_xcojuoqgs1ya) 17

[STRATÉGIE (STRATEGY):](#_magbejrzt39h) 17

[ÉTAT (STATE MACHINE):](#_jyp4bhbvp53x) 17

[**INDEX DES COLLECTABLES**](#_t8fcx2z57w0c) **18**

[BUTINS (6)](#_qnexbef2grx3) 18

[OBJETS (14)](#_fifnxq6f9uej) 18

[RELIQUES (4)](#_f9uctystxw95) 18

[**INDEX DES ENNEMIS**](#_1wq5s08nmpo) **19**

[**INDEX DES BOSS**](#_mobloj3mj1rx) **20**

[**RÉFÉRENCES**](#_s41weqkwsg9t) **21**

# PRÉSENTATION GÉNÉRALE

Ce document se veut comme un outil qui servira par la suite à guider le programmeur lors de la production du projet. Il se veut comme une guide par étape des différentes tâches à réaliser pour être en mesure de livrer un produit fini de qualité. L’objectif de ce document est que le programmeur n’est pas à réfléchir très longtemps aux enjeux de conception et qu’il puisse mettre son énergie sur les problèmes de programmation.

Ce document n’étant pas parfait malgré les nombreuses heures passées à faire la conception, plusieurs changements seront donc effectués en cours de route lors des différents sprints pour permettre de réaliser le projet. Dans le cadre du cours, ce document ne sera pas mis à jour, cependant dans un environnement d’entreprise, ce document devrait être tenu à jour par le chef de scrum[[1]](#footnote-0) pour permettre à l’ensemble des programmeurs d’avoir un document qui est à jour.

# TECHNOLOGIES UTILISÉES

Vous trouverez ci-joint une liste des technologies / langages de programmation qui seront utilisés dans le projet. Les bannières suivantes seront utilisées au début de chaque sous-section du document de conception pour indiquer quelles technologies seront utilisées pour accomplir la tâche requise.

\_HMTL\_ : Ce langage est utilisé pour décrire la structure de chacune des pages webs. Il utilise des balises imbriquées les unes dans les autres pour indiquer au navigateur web (Google Chrome, Firefox, Edge) de quelle façon afficher le contenu de la page. Les moteurs de recherche utilisent l’information qui est mentionnée dans le HTML pour faire apparaître le site dans les recherches des usagers. (Front-End[[2]](#footnote-1))

\_CSS\_ : (Cascading Style Sheet) Ce langage permet de contrôler la façon dont les balises HTML sont affichées à l’écran. En effet, il permet de déterminer l’apparence d’une page en modifiant plusieurs éléments de design tels que la couleur, la police de texte, les marges, la position, etc. (Front-End)

\_SASS\_ : (Syntactically Awesome Style Sheet) Tout comme le CSS, il permet de définir l’apparence d’un site. Cependant, ce langage est utilisé pour écrire le code car il permet d’utiliser des fonctionnalités supplémentaires qui permettent au programmeur d’avoir plusieurs outils qui ne sont pas disponibles dans le CSS de base. En effet, le SCSS introduit les variables qui ne sont pas disponibles dans le CSS de base. Il permet également d’imbriquer des bouts de code les uns dans les autres pour réduire la quantité de code à écrire. Finalement, il permet d’obtenir un code qui sera plus facile à maintenir sur une longue période. (Front-End)

\_JAVASCRIPT\_ : Javacript est un langage orienté objet de haut niveau, c’est-à-dire que nous n’avons pas besoin de penser à l’allocation de la mémoire. Il permettra de créer des applications web, en d’autres mots, il permettra de créer des fonctionnalités pour notre site. Le HTML représente le contenu du site, le CSS l’aspect visuel et le javascript est utilisé pour les fonctionnalités.

\_NODEJS\_ : Ce langage permet d’exécuter du javascript à l’extérieur du navigateur web. Il utilise l’engin V8 de Google pour exécuter le code à l’extérieur. Il permettra de l’utiliser du côté serveur pour bâtir rapidement un réseau d’applications qui pourra facilement être adapté au fur et à mesure que le projet grossit. Il est exécuté sur un seul ‘’Thread’’ et il utilise les évènements. Il peut être utilisé pour construire un API (Application Programming Interface) qui utilise une base de données non relationnelle. (Back-End[[3]](#footnote-2))

\_React\_ : React est une librairie Javascript qui permet de créer des interfaces utilisateur. Elle permet de définir des vues simples pour chaque état de notre jeu, et lorsque nos données changeront, mettre à jour les composantes qui en auront besoin. React utilisent un système de composante ce qui permet de garder de la modularité et maintenir plus facilement le logiciel dans le futur.

# MAQUETTES DE L’INTERFACE GRAPHIQUE

## **PAGE DE CONNEXION**

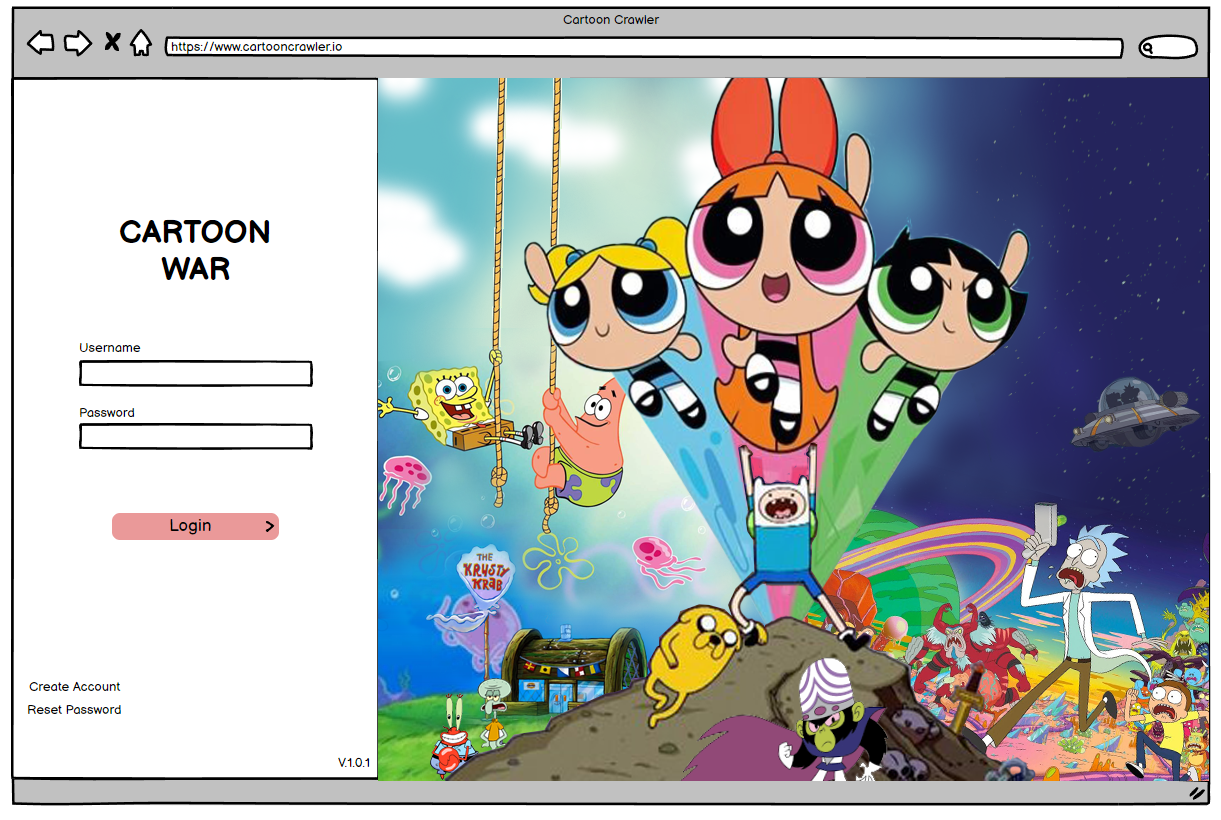
[[4]](#footnote-3)

Figure 1 - Page de connexion

### DESCRIPTION GÉNÉRALE

La page de connexion/inscription est la première page sur laquelle arrive l’usager. Il devra se connecter ou s’inscrire avec un pseudo et un mot de passe. Les données seront envoyées à la base de données pour vérifier si l’usager existe et peut se connecter. S’il s’agit d’une inscription, les données seront enregistrées dans la base de données

Le fenêtre de connexion, située à gauche sur la figure1, permettra également à l’usager de créer un nouveau compte s’il n’en a pas. Lorsqu’on clique sur l’option de création, les champs suivants remplaceront ceux affichés pour la connexion.

Les informations suivantes devront être demandé lors de la création

* Nom d’usager (permet la connexion au compte)
* Mot de passe (par mesure de sécurité, devra contenir 1 chiffre, 1 majuscule, 1 caractère spécial et 8 caractères minimum). **Cette validation devra se faire avec une expression régulière.**
* Confirmation du mot de passe
* email (permettre la récupération du mot de passe en cas d’oubli)

**ÉLÉMENTS TECHNIQUES**

HTML \_CSS\_ \_SASS\_ \_JAVASCRIPT\_

Pour ce qui est du thème et des choix de couleurs, la liberté créative est laissée aux développeurs lors de la réalisation du projet. Par contre, les couleurs devront être harmonisées au travers du projet pour créer une application de qualité. Il sera important d'utiliser les variables pour que les couleurs et thèmes soient facilement ajustable à un seul endroit dans le code. Cette fonctionnalité est disponible en utilisant le langage SASS et permet également d’accélérer le développement des interfaces du jeu.

Un élément de l’interface devra également permettre d'arrêter ou reprendre la lecture de la musique de la page d’accueil.

(Optionnel) Lorsque le joueur se trouve sur la page d’accueil, une légère animation de l’image pourrait ajouter du réalisme au jeu et créer une ambiance plus dynamique pour le joueur. Puisque l’objectif de ce projet est principalement l’aspect programmation, cet aspect devra être travaillé uniquement si le jeu est 100% fonctionnel.

## **MENU D’ACCUEIL**

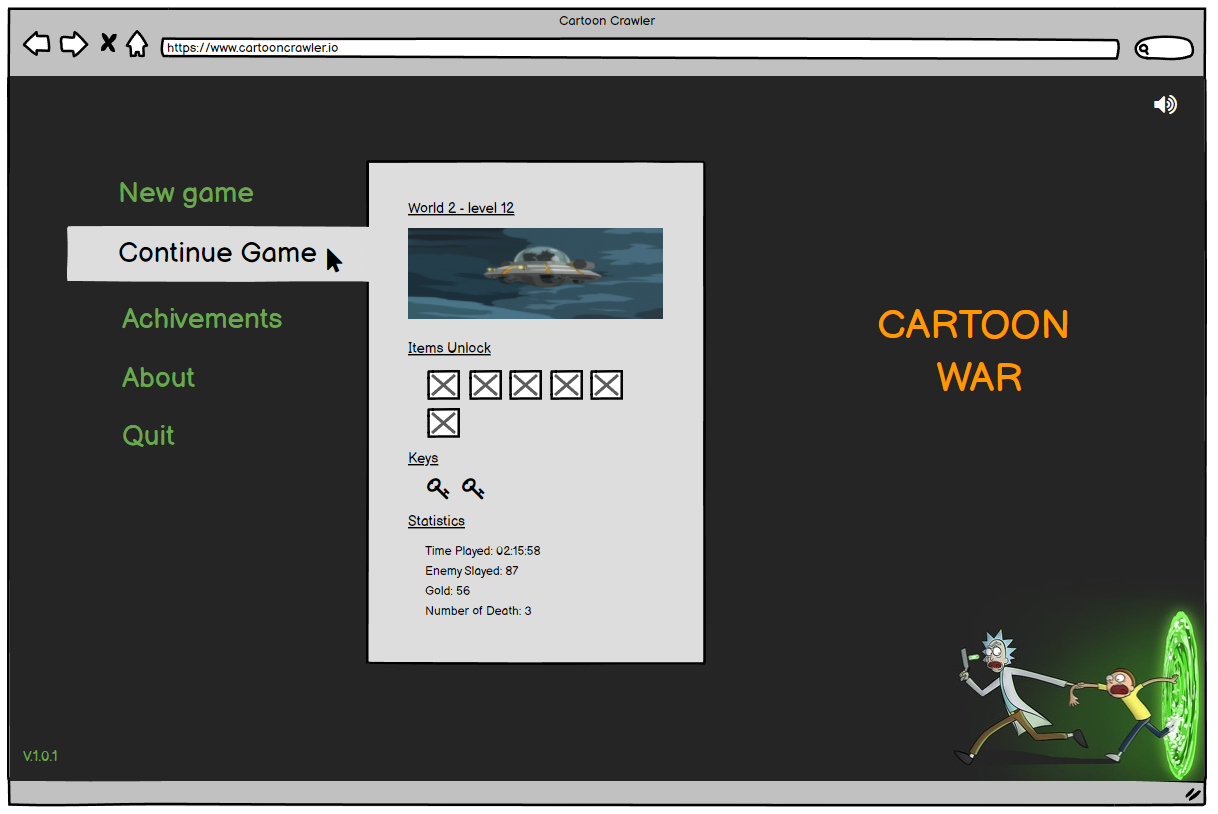


Figure 2 - Menu d’accueil

### DESCRIPTION GÉNÉRALE

La page de lobby est celle qui apparaît une fois que l’usager est connecté. L’usager doit choisir le personnage qu’il veut incarner parmi 4 choix. Il peut sélectionner les items qu’il veut utiliser pendant la partie en cliquant dessus. Les items sélectionnés doivent être en surbrillance pour que l’usager voit quels sont ceux sélectionnés. Une fois que le joueur clique sur *Start Game*, les items sélectionnés seront conservés dans la liste *items* de la classe *Player*, puis la page changera pour la page de jeu.

Plusieurs options supplémentaires s’offre également au joueur à partir de ce menu:

* Continuer une partie en cours. (Cette option chargera depuis le début, le dernier niveau non complété par le joueur. Voir la section base de données pour plus de détails sur la sauvegarde d’une partie)
* Trophées (Permet de voir la progression du joueur dans le jeu)
* À propos (Cette option permet de comprendre le fonctionnement du jeu ainsi que de voir les informations des développeurs qui ont travaillé sur ce jeu).
* Quitter (cette option permet de se déconnecter et retourner au menu principal)

### ÉLÉMENTS TECHNIQUES

HTML \_CSS\_ \_SASS\_ \_JAVASCRIPT\_

Plusieurs éléments techniques devront être réalisés sur cette page. Le menu devra être simple et aéré pour permettre d’afficher l’information essentielle pour le joueur. Lorsque l’usager clique sur *créer une partie* ou *continuer une partie*, des sous-menus devront apparaître. Ils devront contenir les informations suivantes:

* Menu ‘’Nouvelle Partie’’: Cette option permet d’effacer la dernière partie sauvegardée et en débuter une nouvelle. Plusieurs options seront offert à l'usager pour lui permettre de personnaliser son expérience de jeu:
  + Choix d’un personnage. Chaque personnage est visuellement différent, possède des habiletés et bonus différents. Le joueur devra donc choisir celui qui correspond le mieux à l’expérience souhaitée.
  + Choix de la difficulté. Cette option sera ajoutée seulement si le temps alloué pour la réalisation du jeu le permet. Plusieurs choix sont disponibles et affectent plusieurs caractéristiques des ennemis.
  + Choix de la relique qui sera active. Pour plus d’information sur le rôle et le fonctionnement des reliques, consulter le document du mandat.
* Menu ‘’Continuer une partie’’. Cette option permettra de poursuivre une partie qui a déjà été entamée par le joueur. Lors que le joueur choisit cette option, il obtient un résumé des éléments principaux de sa dernière partie soit:
  + Niveau atteint
  + Temps de jeu pour cette partie
  + Niveau de difficulté
  + Nombre de clés amassées
  + Liste des objets qui ont été acheté ou collecter
  + Vie restante
  + Nombre de pièces de monnaie

Tout comme la page précédente, le choix des thèmes et couleurs est laissé à la discrétion des programmeurs. L’important est bien sûr que les pages conservent une forme d’uniformité entre elles.

(Optionnel) Pour ajouter de dynamisme, et engagé le joueur dans l’histoire, une courte mise en situation du contexte du jeu pourrait être ajouté lors du lancement d’une nouvelle partie

## **ACHIEVEMENTS**

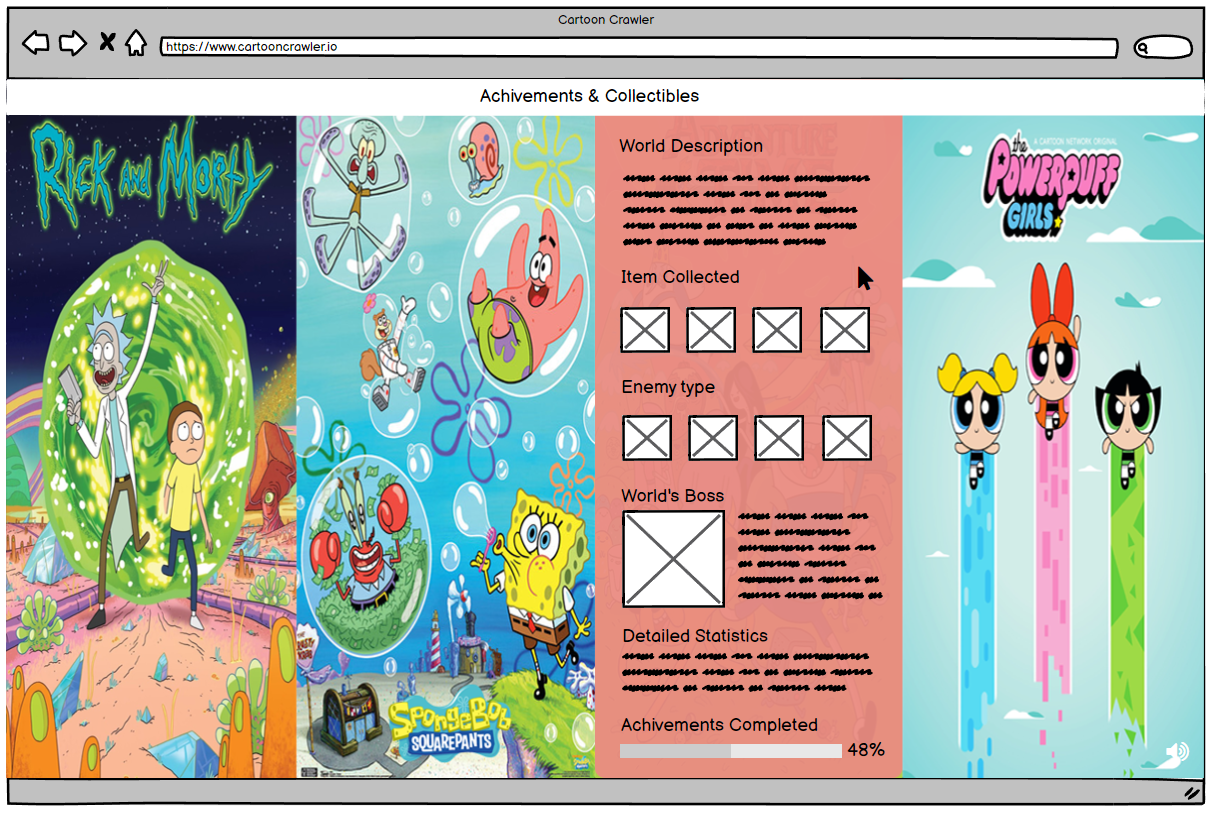


Figure 3 - Progression

### DESCRIPTION GÉNÉRALE

La page de progression permet au joueur de consulter les différents objets, trophées et sa progression générale dans les différents univers du jeu. Cette page donne également une foule d'informations utiles sur les différents univers et les ennemis qui le composent. Cet outil se veut un guide pour que le joueur puisse obtenir différentes informations sur les ennemis qu’il a déjà rencontrés ainsi que les effets des objets qui ont été amassés durant les parties précédentes.

### ÉLÉMENTS TECHNIQUES

HTML \_CSS\_ \_SASS\_ \_JAVASCRIPT\_

Le design de cette page utilise la fonctionnalité des grids qui est disponible en CSS. En effet, la page devra être séparée en 4 sections, soit une pour chacun des univers de dessin animé. Pour consulter l’information de chacun des mondes, il suffit de survoler avec la souris le monde souhaité. Un résumé de l’information apparaît alors à l’écran. Pour permettre à l’usager de consulter le détail complet, il peut simplement cliquer dans la fenêtre souhaitée et une page complète s’ouvrira.

L’information de cette page provient directement de la base de données et elle est rechargée à chaque fois que l'usager accède à la page de progression à partir du menu principal.

## **FENÊTRE DE JEU**

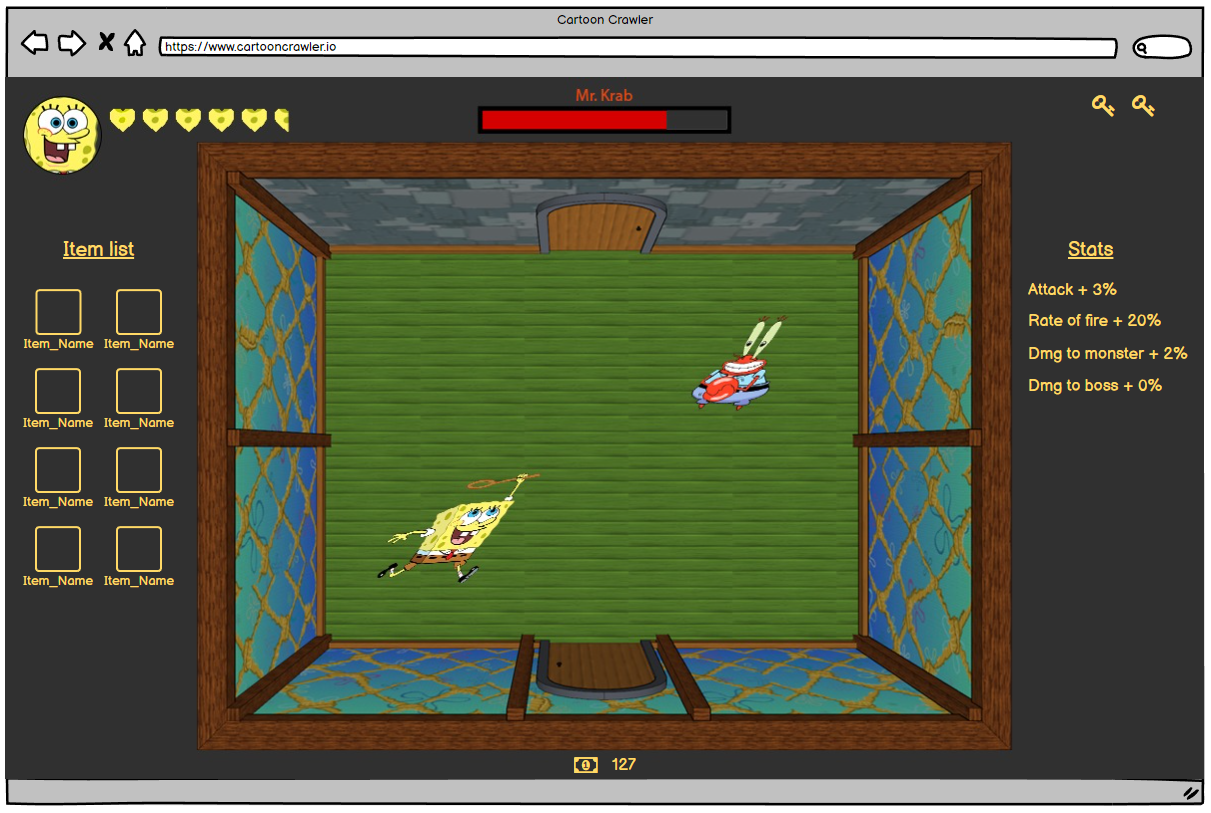


Figure 4 - Fenêtre de jeu

Figure 5 - Mise en pause du jeu

### DESCRIPTION GÉNÉRALE

L’écran de jeu est la page principale que l'usager verra le plus souvent. Elle est mise à jour en temps réel et permet d’afficher le jeu, ainsi que les différents éléments d’information.

### ÉLÉMENTS TECHNIQUES

HTML \_CSS\_ \_SASS\_ \_JAVASCRIPT\_

Cette fenêtre de jeu est composé de deux sections principales:

* La zone de jeu: Cette zone correspond à la salle au centre de l’écran dans laquelle le joueur peut déplacer son personnage et affronter les différents ennemis qui se dressent sur son chemin. Puisqu’un niveau est composé de plusieurs salles, cette pièce changera en fonction de la position du joueur dans le niveau. Les pièces contiennent 4 portes même si certaines d’entre elles ne mènent à rien pour ajouter une difficulté. Ces portes sont situées aux quatres côtés de la salle (Haut, Bas, Gauche, Droite).
* Le menu d’information à l’usager: Cette zone permet d’afficher les différents indicateurs qui permettent au joueur de voir l’état de sa partie. Il est composé des élément suivants:
  + Liste d’objets achetés dans les boutiques
  + Point de vie du joueur
  + Image du personnage choisi par le joueur
  + Statistique (Cette zone permet de voir les différents attribut du joueur avec les bonus offert par les items qu’il a acheté)
  + Clé amassé
  + Nombre de pièces amassées
  + Nom de l'ennemi et point de vie restant à celui-ci

# 

# CONCEPTION UML

## DIAGRAMMES DES CAS D’USAGE

### Usager dans le login

Figure 6 - Diagramme cas d'usage sur la page login

### **Usager dans le lobby**

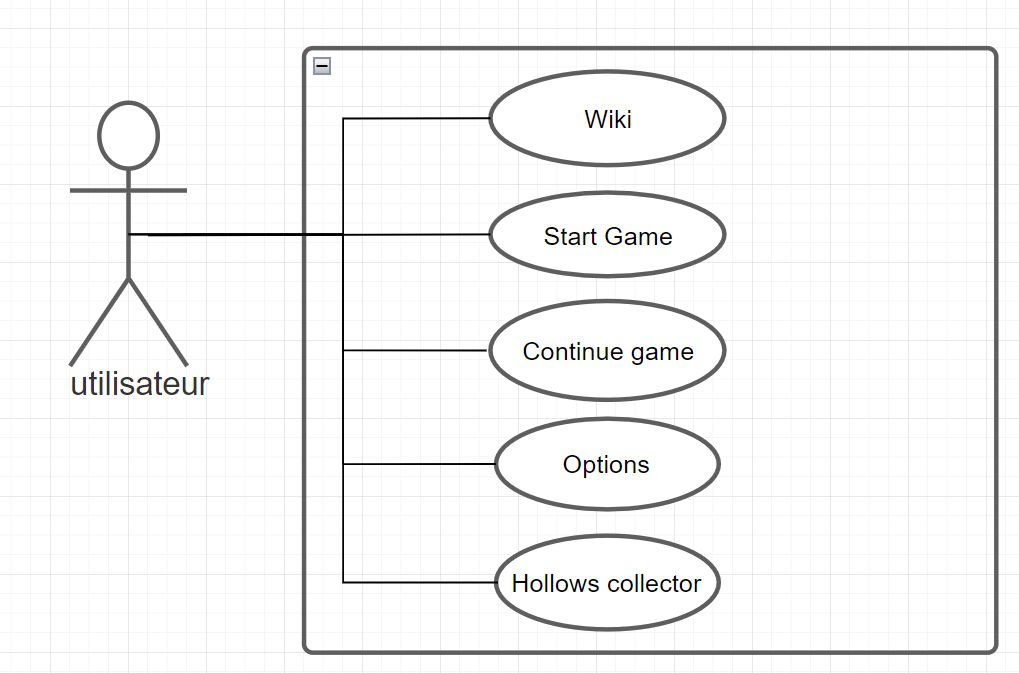


Figure 7 - Diagramme cas d'usage dans le lobby

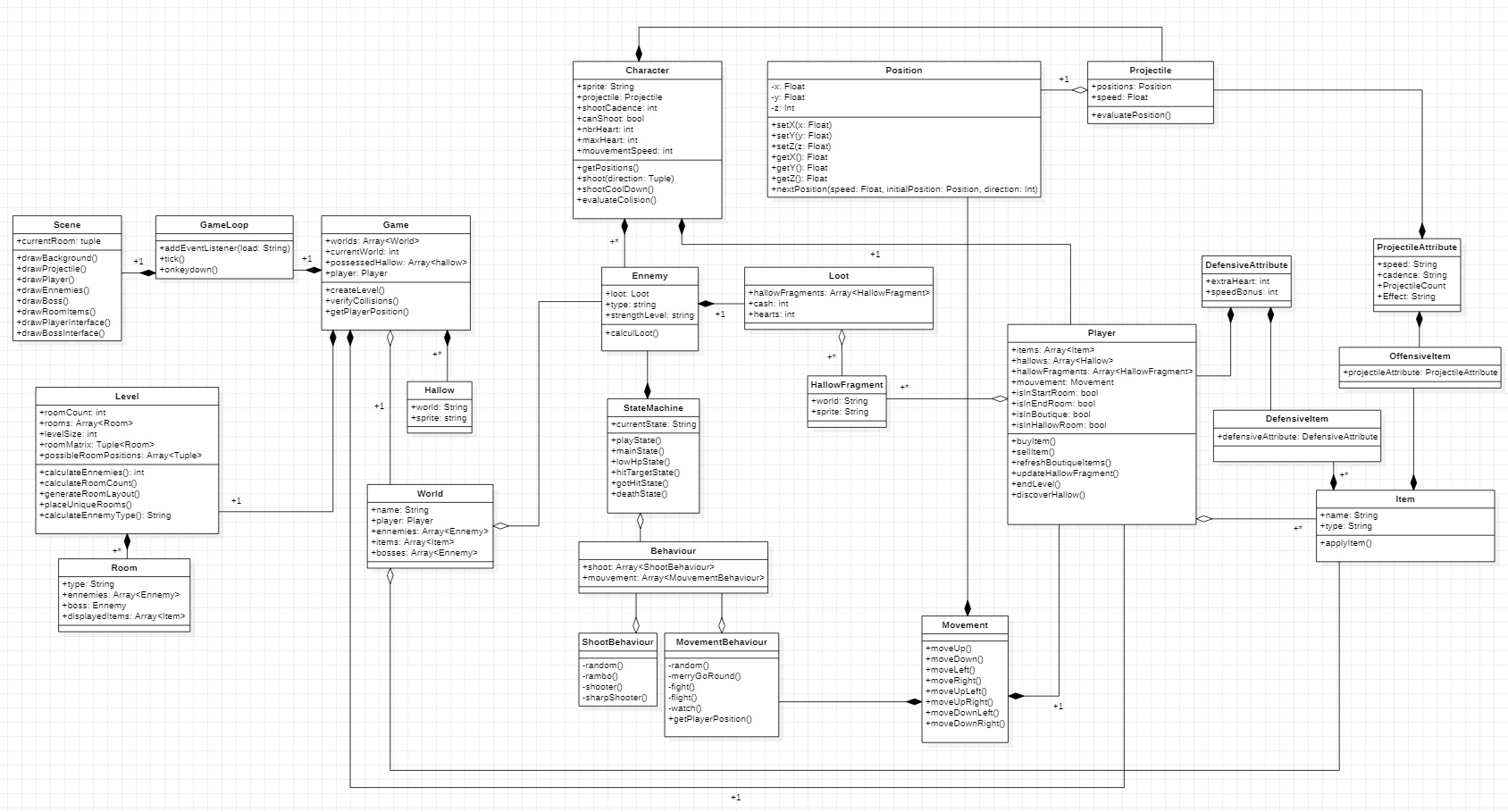
### **Usager dans le jeu**

*Figure 8* - Diagramme cas d'usage dans le jeu. Pour une meilleure vision, voir le fichier *diagramme cas usage.drawer* dans le dossier *doc* du Sprint 0

## 

## DIAGRAMMES DE CLASSE

## 



*Figure 9* - Diagramme de classe. Voir le fichier original *diagramme classes.mdj* se trouvant dans le dossier *doc* du Sprint 0

## 

# 

# SCHÉMA DE LA STRUCTURE DE DONNÉES EXTERNES

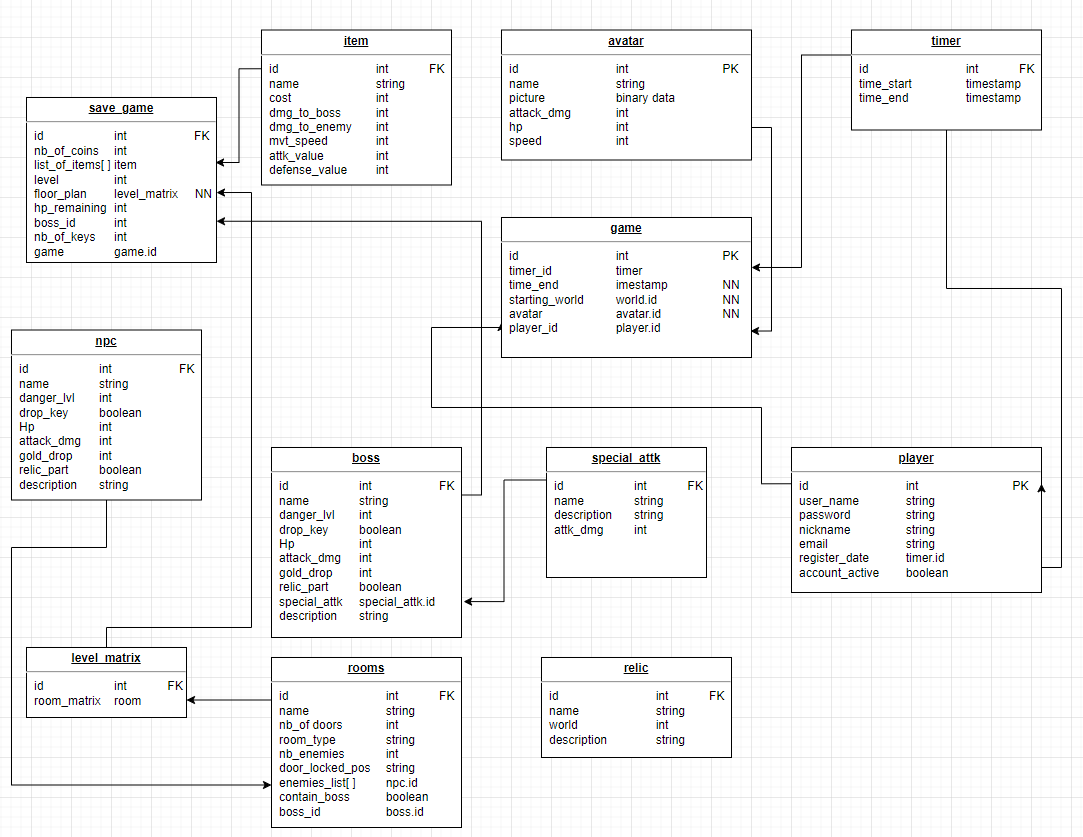
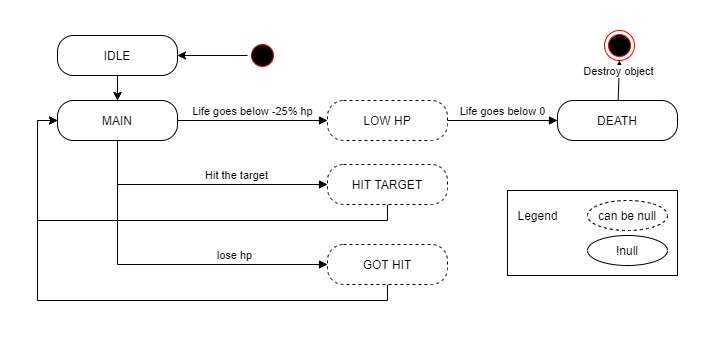


Figure 10 - Diagramme de base de données

# COMPORTEMENT DES ENNEMIS

## STATES

Les ennemis fonctionnent sous la forme de ”state machine”. Les comportements de chaque ennemi se basent sur le même “state machine”. Ce dernier contient trois états principaux et trois potentiels états intermédiaires. Ces structures d’état se retrouvent chez tous les ennemis.



NB: Le state d'idle ne sera jamais vu par le joueur car il change dès que le joueur entre dans la pièce.

## BEHAVIOR

Les états contiennent des comportements, appelés “behaviours”. Ces comportements varient de façon différente pour chacun des ennemis..

### Movement Behaviour

**Random**: L'entité se déplace de façon complètement aléatoire. Il avance dans une direction pendant un temps puis peut la changer n’importe quand.

**Merry-Go-Round**: L'entité longe les murs. Elle se dirige vers le mur le plus proche puis le suit dans le sens horaire.

**Fight**: L'entité cible le joueur. Il se déplace vers le joueur.

**Flight**: L'entité cible le joueur. Il se déplace à la position la plus éloignée du joueur.

**Watch**: L'entité cible le joueur en restant à la même distance du joueur.

### Shoot Behaviour

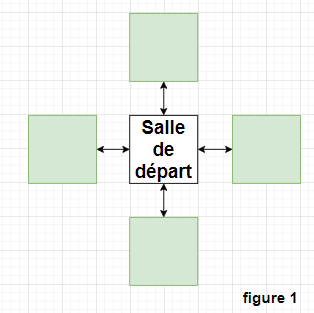
**Random**: L'entité tire de façon complètement aléatoire. Elle tire un projectile dans une direction puis change de direction.

**Rambo**: L'entité tire de façon complètement aléatoire dans un cône de visée. Le cône de visée est formé à partir de la position du joueur avec +- 45 degrés.

**Shooter**: L'entité cible le joueur. Il tir **sur** la position actuelle du joueur

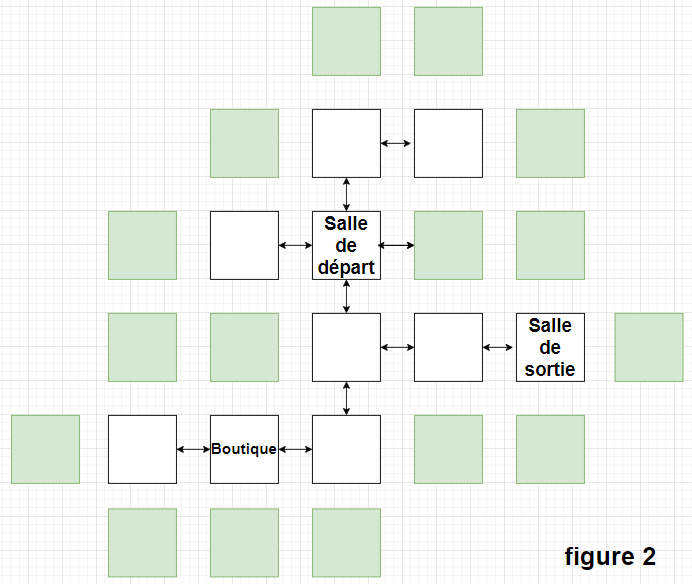
**Sharp Shooter**: L'entité cible le joueur. Elle tire à la position où **se trouve** le joueur. Elle prend en compte la vitesse du joueur, la vitesse du projectile et une évaluation du déplacement futur du joueur en fonction des trois derniers déplacements du joueur.

# GÉNÉRATION D’UN NIVEAU - (ALGORITHME)

**Étape 1:** Calculer le nombre de salle du niveau à générer (LOG(niveau;1,2)+5). Créer une matrice vide impaire de taille n, ou n = sqrt du nbr de salle arrondi au supérieur. On ajoute au centre de la matrice un objet **Room**  de type **Start Room.**

**Étape 2:** Créer un vecteur de tuple qui contient les positions possibles des prochaines salles. Calculer la borne inférieure et supérieure du nombre d’ennemis par salle.

**Étape 3:** Sélectionner au hasard une position dans le vecteur. Ajout dans la matrice à la position sélectionner, un objet **Room**  de type **Common Room.** Calcule le nombre aléatoire d’ennemis compris entre les bornes déterminées à l’étape 2. Créer la liste d’ennemis de la salle. Retire cette position du vecteur et ajoute les nouvelles potentiel position de salle. Répéter l'étape 3 i fois, ou i = au nombre total de salle calculé à l'étape 1.

Par exemple, dans la *figure 1*, on voit qu’il serait possible que la salle suivante se trouve en haut, en bas, à gauche ou à droite de la salle de départ (cases vertes). Dans la *figure 2*, on peut voir le résultat après 9 tours de boucle. Si une salle se retrouve à l’extérieur de la matrice, elle ne doit pas être ajoutée dans les salles possibles.

**Étape 3:** Choisir une salle au hasard de type **End Room** et **Boutique.** Effacer les ennemis dans ces salles.

# 

# SYSTÈME DE COLLISION - (DATA STRUCTURE)

Le système de collision permet de savoir si plusieurs entités entrent en contact puis traitent la collision en fonction des types d'entités qui sont entrées en contact. Tout d’abord, il faut détecter si une collision à lieu et pour ce faire, on à 3 options d'ordre de plus en plus petit et de plus en plus difficile à implémenter. Dans notre cas, nous utiliserons la plus facile et rapide (méthode 1) pour commencer et si nous avons le temps, nous essaierons de la remplacer par une des deux autres options. Après avoir détecté la collision, il faut la traiter. Chaque entité possède une méthode collision True(arg) qui prend pour arg l'entité dans laquelle elle est entrée en collision.

### OPTION 1: ordre n2

On crée une matrice carrée d'ordre n, où n est égal au nombre d'entités affichées à l'écran. À chaque *tick,* on met à jour les positions puis faisons une boucle dans la matrice pour vérifier si deux positions sont identiques. Si elles le sont, on à une collision.

### OPTION 2: ordre n x log(n)

On construit un KD tree qui segmente l’espace en plusieurs sous-espaces de même taille. Au moment de détecter les collisions, on teste la position des entités avec celle présente dans son sous espace. Si les coordonnées sont identiques alors il y à une collision.

### OPTION 3: ordre n(1)

La troisième option est de transformer l’espace du jeu de continu à discret. Puis dans une lookup table de n par m, ou n et m représentent la hauteur et largeur de l’espace discret créé (par exemple, 100x100). Pour chaque mouvement d’objet on ajoute une référence à celui-ci dans la lookup table. Si une des position d'arrivée d’un objet contient une référence à un autre objet, ces deux objets sont en collision.

# 

# 

# PATRONS DE CONCEPTION

# 

## PROTOTYPE (PROTOTYPE):

Prototype est un patron de conception qui crée de nouveaux objets à partir d’objets existants sans rendre le code dépendant de leur classe. Ce patron de conception sera utilisé lors de l’action de tir des personnages (joueurs et ennemis). Chaque entité possede en mémoire un objet Projectile qui , lorsqu’il décide de terre, est copié pour être “lancer”.

## STRATÉGIE (STRATEGY):

Le patron de conception stratégie vous propose de prendre une classe dotée d’un comportement spécifique mais qui l’exécute de différentes façons, et de décomposer ses algorithmes en classes séparées appelées stratégies. Ce patron de conception sera utilisé dans la gestion des **Behaviours** des ennemis. Le state machine qui contrôle des décisions des ennemis possede comme action differents **Behaviour** qui eux réponde au patron de Stratégie.

## ÉTAT (STATE MACHINE):

État est un patron de conception comportemental qui permet de modifier le comportement d’un objet lorsque son état interne change. L’objet donne l’impression qu’il change de classe. Le principe repose sur le fait qu’un programme possède un nombre fini d'états. Le programme se comporte différemment selon son état et peut en changer instantanément. En revanche, selon l’état dans lequel il se trouve, certains états ne lui sont pas accessibles. Ces règles de changement d’état sont appelées transitions. Elles sont également finies et prédéterminées.

Ce patron de conception sera utilisé comme “cerveau” de ennemi. Il va travailler main dans la main avec le patron de Stratégie pour donner au ennemies une structure de **Behaviour** uniforme.

# 

# INDEX DES COLLECTABLES

## BUTINS (6)

1. Coeur: Soigne 1 coeur vide
2. Gold: varie entre 1 et 5 unité
3. Fragment de relique Bob
4. Fragment de relique Morty
5. Fragment de relique Super Nana
6. Fragment de relique Finn

## OBJETS (14)

Offensif

1. Augmente la vitesse du projectile à rapide
2. Double le nombre de projectile par tir
3. Ajoute l’effet Flamme
4. Ajoute l’effet Glace
5. Ajoute l’effet Paralysie
6. Ajoute l’effet Confusion
7. Ajoute l’effet Peur
8. Ajoute l’effet Folie
9. Double le nombre de projectiles, augmente la vitesse et la cadence à rapide et retire les effets du projectile
10. Augmente la cadence de tir à instantanées

Défensif

1. Donne 1 coeur supplémentaire
2. Augmente la vitesse de déplacement de 25%
3. Augmente la vitesse de déplacement de 10%
4. Donne un bouclier qui ignore 1 projectile. recharge 30s.

## RELIQUES (4)

1. Bob l’éponge: Spatule
2. Adventure Time: Enchiridion
3. Rick et Morty: Portal fluid
4. Les Super Nanas: Agent chimique X

# 

# INDEX DES ENNEMIS

# 

Bob l’éponge

1. Carlos
2. Méduse
3. Femme de plankton
4. Bubble bass
5. Doodle Bob

Adventure Time

1. Roi des glaces
2. Citronnelle
3. Magic man
4. Golb
5. Donny (Jerk guy)

Rick et Morty

1. meeseeks
2. King jellybean
3. memory parasites
4. the Hooves
5. Mr. Nimbus

Les Super Nanas

1. Mojojojo
2. Princess Morbucks
3. Sedusa
4. Fuzzy Lumpkins
5. The gangreen gan

# 

# INDEX DES BOSS

**Man ray**

* Monde: Bob l’éponge
* Projectile:
  + Rapide
  + Lent
  + 5
  + Glace
* Main state:
  + Mouv: Flight
  + Shoot: Rambo
* Low Hp state
  + Mouv: Merry-Go-Round
  + Shoot: Shooter
* Hit target state:
  + Mouv: Flight
  + Shoot: Shooter
* Got Hit state:
  + Mouv: Random
  + Shoot: Random

**The Lich**

* Monde: Adventure Time
* Projectile:
  + Lent
  + Rapide
  + 2
  + Confusion
* Main state:
  + Mouv: Fight
  + Shoot: Random
* Low Hp state
  + Mouv: Flight
  + Shoot: Shooter
* Hit target state:
  + Mouv: Fight
  + Shoot: Random
* Got Hit state:
  + Mouv: Flight
  + Shoot: Rambo

**Evil morty**

* Monde: Rick et Morty
* Projectile:
  + Normal
  + lent
  + 1
  + Flamme, Confusion
* Main state:
  + Mouv: Random
  + Shoot: Shooter
* Low Hp state
  + Mouv: Flight
  + Shoot: Rambo
* Hit target state:
  + Mouv: Fight
  + Shoot: Sharp Shooter
* Got Hit state:
  + Mouv: Merry-Go-Round
  + Shoot: Sharp Shooter

**HIM**

* Monde: Les Super Nanas
* Projectile:
  + Rapide
  + Rapide
  + 5
  + aucun
* Main state:
  + Mouv: Fight
  + Shoot: Rambo
* Low Hp state
  + Mouv: Fight
  + Shoot: Sharp Shooter
* Hit target state:
  + Mouv: Fight
  + Shoot: Shooter
* Got Hit state:
  + Mouv: Watch
  + Shoot: Rambo

# RÉFÉRENCES

SITES WEB

Prototype. https://refactoring.guru/fr/design-patterns/prototype. Refactoring Guru.

Stratégie. https://refactoring.guru/fr/design-patterns/strategy. Refactoring Guru.

État. https://refactoring.guru/fr/design-patterns/etat. Refactoring Guru.

The Spriter Resource. https://www.spriters-resource.com/game\_boy\_advance/powerpuffgirlshimseek/. Cartoon Network..

The Spriter Resource. https://www.spriters-resource.com/pc\_computer/adventuretimeexplorethedungeonbecauseidontknow/. Cartoon Network.

The Spriter Resource. https://www.spriters-resource.com/custom\_edited/rickandmortycustoms/. Cartoon Network.

The Spriter Resource. https://www.spriters-resource.com/game\_boy\_advance/spongebobsquarepantsthemovie/. Nickelodeon

1. Scrum : Rencontre hebdomadaire entre une équipe de travail pour partager les avancements et déterminer les objectifs de travail pour la semaine à venir. [↑](#footnote-ref-0)
2. Front-End : Ce terme anglais signifie un type de développement qui représente l’aspect cosmétique ou visible par l’usager qui utilisera l’application. [↑](#footnote-ref-1)
3. Back-End : Ce terme anglais signifie que le développement effectué est fait du côté serveur ou pour créer des fonctionnalités qui ne seront pas utilisées directement par l’usager, mais qui servent au bon fonctionnement du site et de ses fonctionnalités. [↑](#footnote-ref-2)
4. The Spriter Resource. https://www.spriters-resource.com/game\_boy\_advance/powerpuffgirlshimseek/.(Nickelodeon) [↑](#footnote-ref-3)