M2104 : Rapport de Conception Orienté Objet

PROJET RUNNER

NÉDÉLEC Guillaume

--

MARCILLOUX Nicolas

Groupe B”

Année 2015-2016

Enseignant Responsable : Olivier Guibert

Sommaire

I - Choix de Réalisation du Projet

Niveau de vie :

- 10 points de vie à chaque début de partie.
- La vie est représentée par une jauge horizontale colorée (remplie à 100% au début de la partie).
- Le joueur perd entre 1 et 4 points de vie à chaque fois qu'il heurte un obstacle (en fonction du type d'obstacle).
- Le joueur récupère 3 points de vie à chaque bonus de santé attrapé.

Collision avec les obstacles :

- Il y a 4 types d'obstacles de forme/taille différentes.
- Chaque obstacle fait perdre un certain nombre de point(s) de vie selon son type (1, 2, 3 ou 4 points de vie).

Objets et obstacles :

- Les obstacles sont générés aléatoirement (type et délai d'apparition) tout en laissant une chance au joueur de ne pas en heurter (espace entre eux).
  (temps maximum entre les obstacles : 2 secondes)
- Les bonus sont aussi générés aléatoirement (type et délai d'apparition) mais moins régulièrement que les obstacles (entre 10 et 20 secondes).

Score :

- Le score s'affiche en bas à droite de la fenêtre.
- Il s'incrémente de deux manières parallèles :
  - avec la "distance parcourue" (variable incrémentée avec le temps)
  - avec la récupération de pièces générées à la manière des obstacles (+100 points par pièces)

Mode de déplacement de la balle :

- Pédestre (mode au sol, basique).
- En saut (après utilisation de la touche “UP”).

Algorithme de calcul du score :

Tant que le joueur est en vie (vie > 0)

*+10 points toutes les secondes + (10 * phase actuelle*difficulté)*

Si le joueur ramasse une pièce

*+100 points + 10 * phase actuelle*

Fin si

Fin tant que

*difficulté : facile = 1 ; normal = 2 ; difficile = 3

Vitesse en fonction du temps :

- La vitesse des obstacles/bonus/pièces sont les mêmes.
- Celle-ci augmente à chaque changement de phase par le simple calcul :
  Vitesse_Suivante = Vitesse_Initiale + (Phase*Difficulté)/10

Bonus :

Nom du bonus	Effet du bonus	Obtention du Bonus
Multiplicateur	Points x2 pendant 30 secondes	Lorsque le joueur obtient 10 pièces d'affilée sans subir de dégâts
Bouclier	Protège le joueur d’1, 2 ou 3 obstacles en fonction du niveau de difficulté (facile, normal, difficile)	Bonus à attraper
Double saut	Permet au joueur de sauter 2 fois d'affiler pendant 30, 20 ou 10 secondes en fonction du niveau de difficulté (facile, normal, difficile)	Bonus à attraper
Santé	Permet au joueur de regagner 3 points de vie	Bonus à attraper

➡ Les bonus actifs sont affichés en bas de la fenêtre

Meilleurs scores :

- À chaque fin de partie le score du joueur sera affiché à l'écran ainsi que les 10 meilleurs score. Si le score du joueur fait partie de ces scores alors il sera sauvegardé dans un fichier.
- Dans le menu, un bouton ***Best Scores*** permettra d'afficher à l'écran les 10 meilleurs scores réalisés (chargement du fichier de score).

Difficulté du jeu :

- Le joueur aura le choix (dans le menu) entre le mode Facile, Normal et Difficile.
- Le mode de difficulté influe sur :
    - La vitesse du jeu (augmente avec la difficulté)
    - le nombre de points obtenus (augmente avec la difficulté)
    - la fréquence d'apparition des Bonus (augmente avec la difficulté)
    - le temps d'effet des Bonus (diminue avec la difficulté)

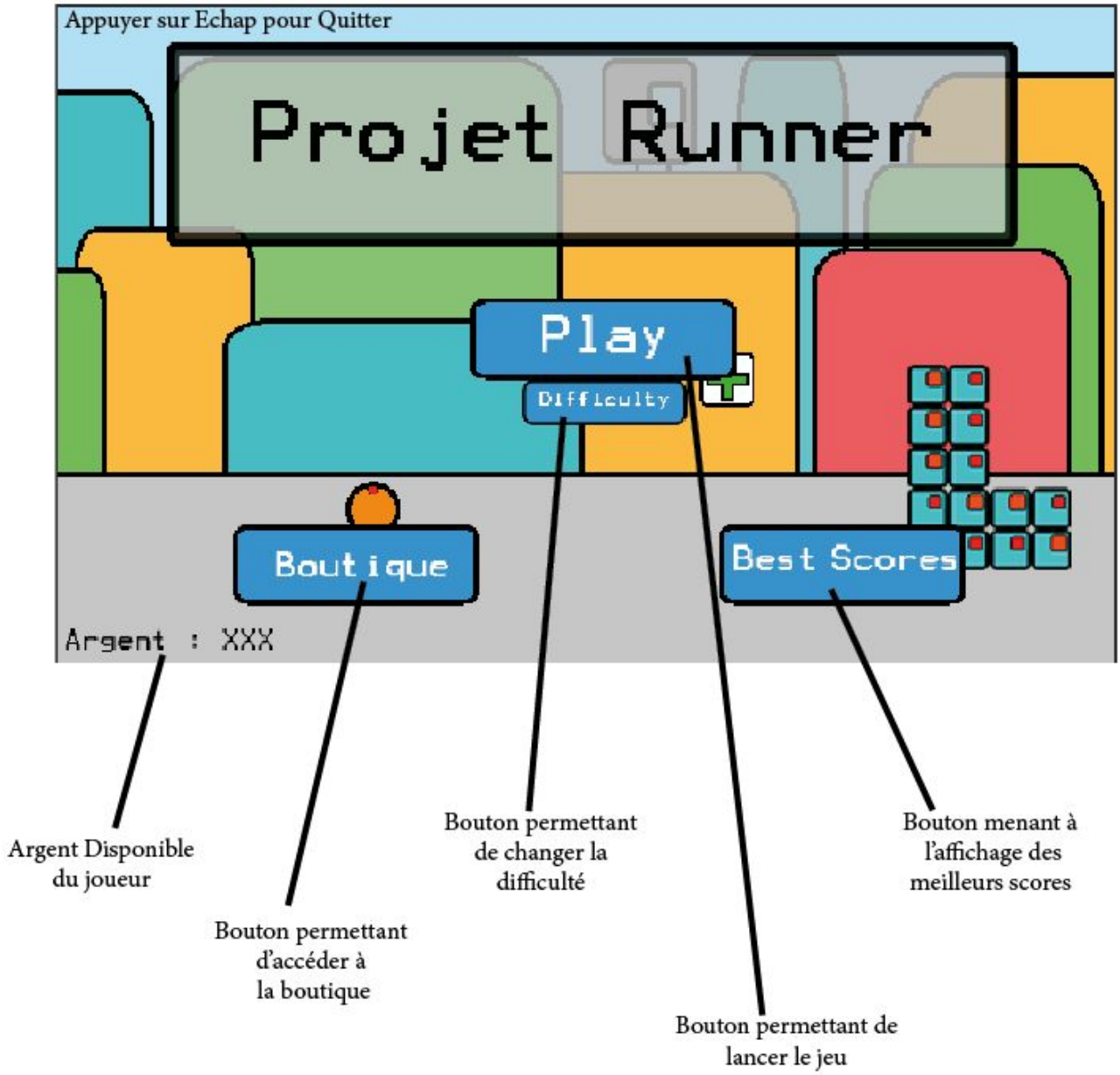
Porte-monnaie virtuel :

- À chaque fin de partie, le score est converti en "argent" (entier) qui sera sauvegardée dans un fichier.
- Cet argent permettra d'acheter des Bonus utilisables une seule fois chacun pour la partie suivante.
- L'argent disponible sera affichée en bas de la fenêtre dans le sous-menu ***Boutique*** ainsi que dans le menu principal.

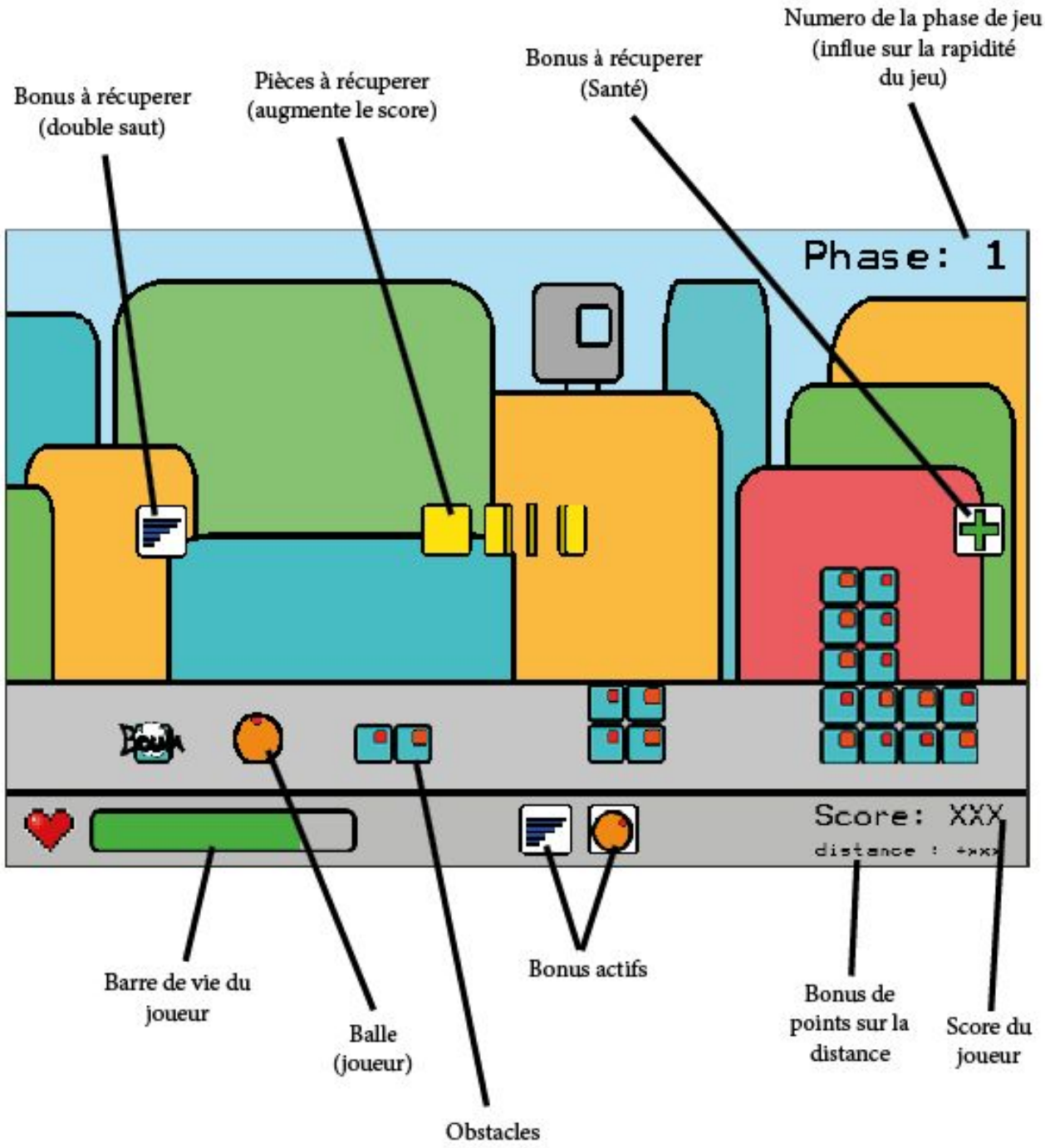
Effets Sonores :

- Présence d'une musique tout au long du jeu.
- Effets sonores temporaires lors d'une collision avec un obstacle, une pièce ou encore un bonus.

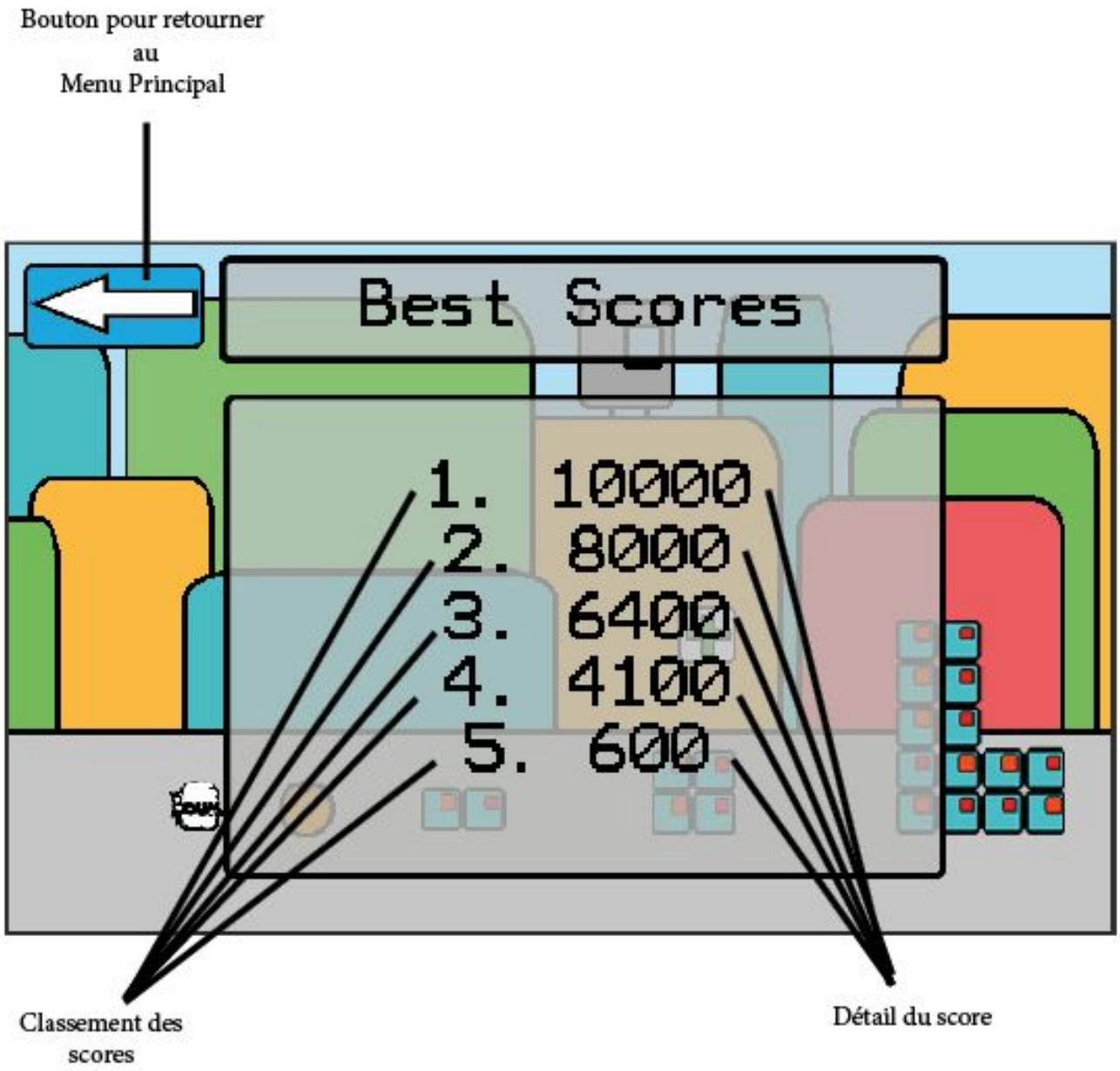
Maquette du Menu Principal



Maquette de la phase de jeu

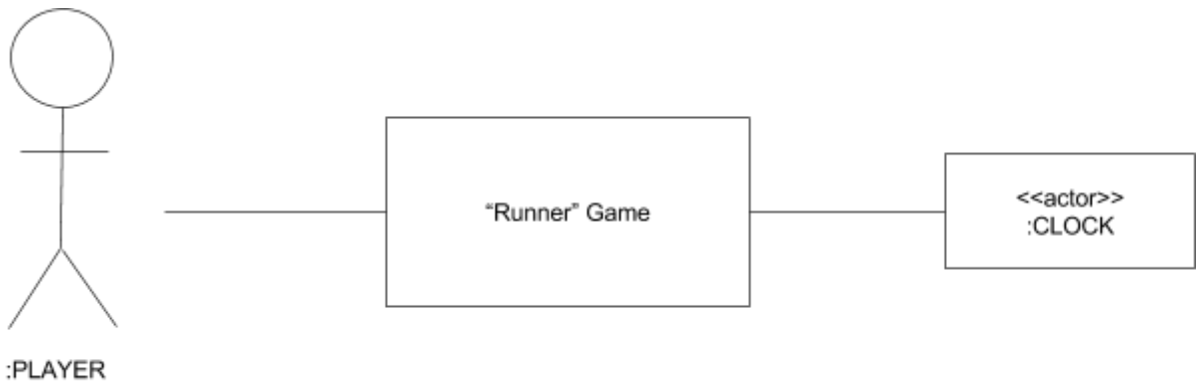


Maquette de l'écran des scores



II - Conception Orientée Objet du Projet

Diagramme de contexte Statique



Évènement externes

ARR_Start_Game
ARR_Quit_Game
ARR_Menu_Navigation
ARR_Pause
ARR_Back_To_Menu
ARR_Left_Move
ARR_Right_Move
ARR_Jump
ARR_Shop_Purchases

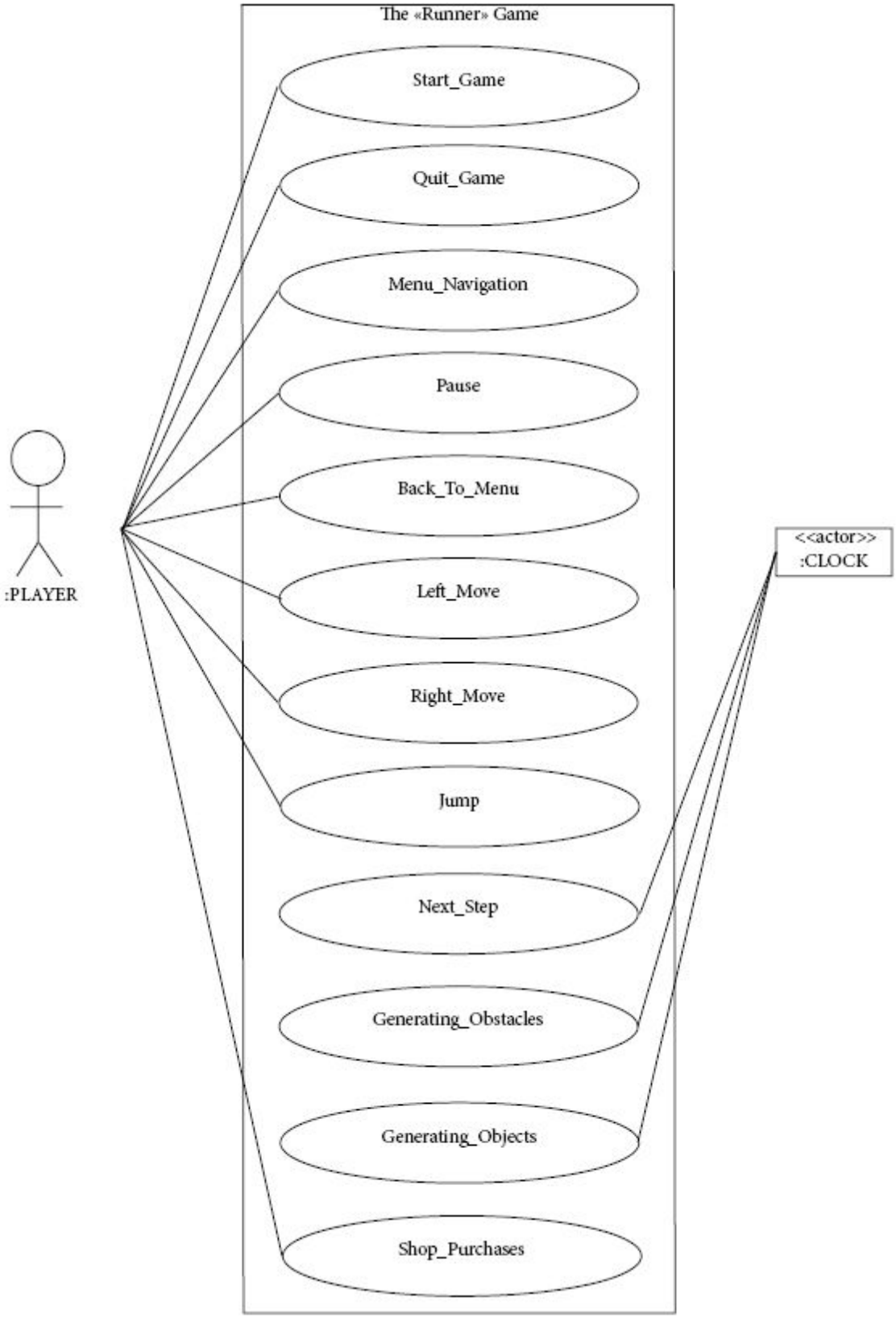
Évènement temporels

ARR_Next_Step
ARR_Generating_Obstacles
ARR_Generating_Objects

Évènement de résultats

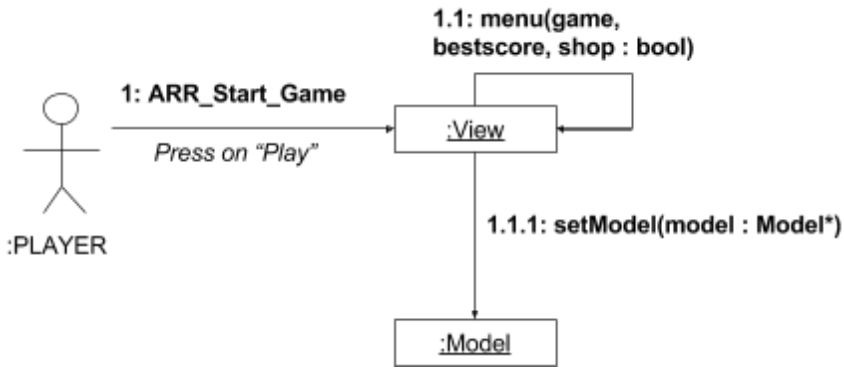
ENV_End_Game
ENV_Player_Damages
ENV_Score_Calculation
ENV_Bonus_Effects
ENV_Sounds_Effects

Diagramme de cas d'utilisation

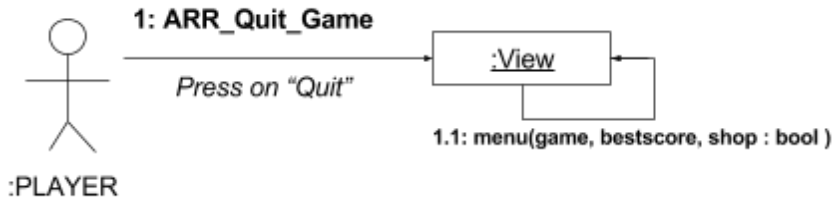


Diagrammes de Communications Nominal (DCN):

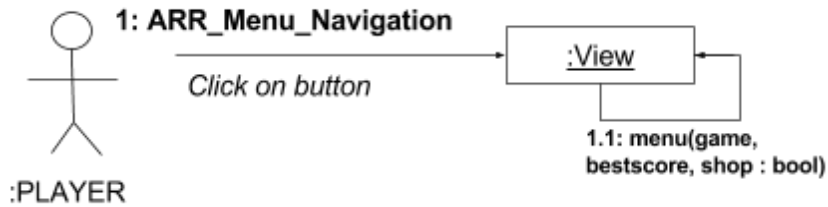
DCN de ARR_Start_Game :



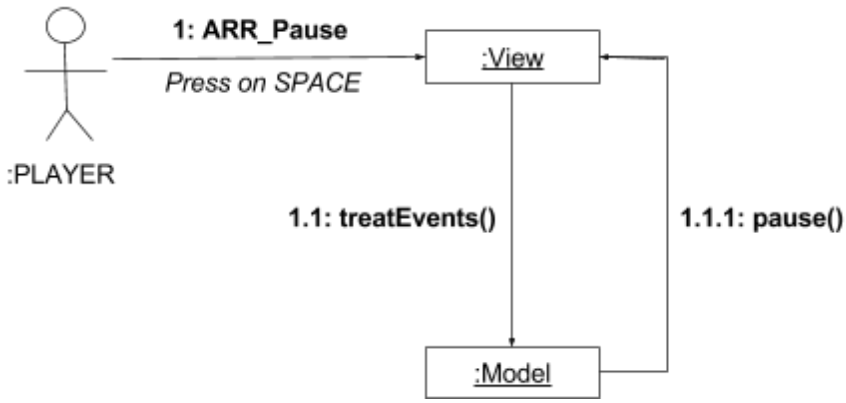
DCN de ARR_Quit_Game :



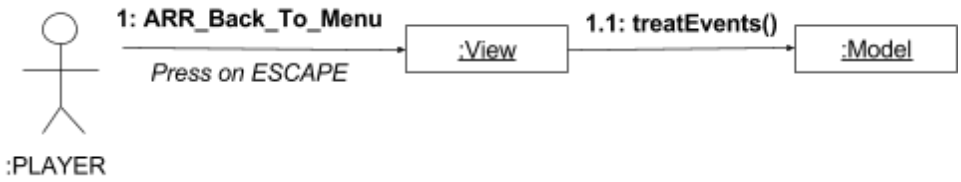
DCN de ARR_Menu_Navigation :



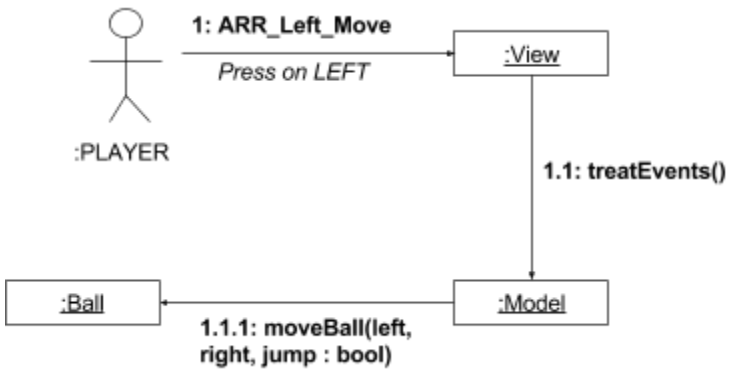
DCN de ARR_Pause :



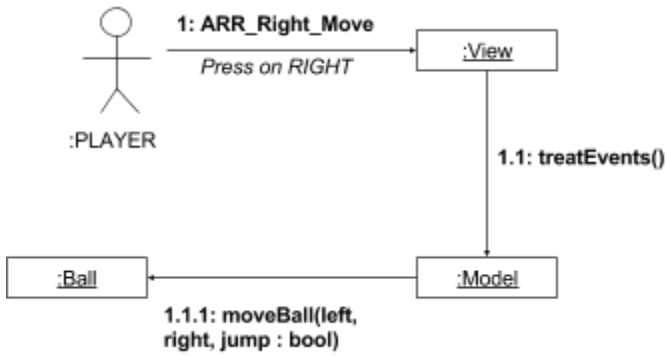
DCN de ARR_Back_To_Menu :



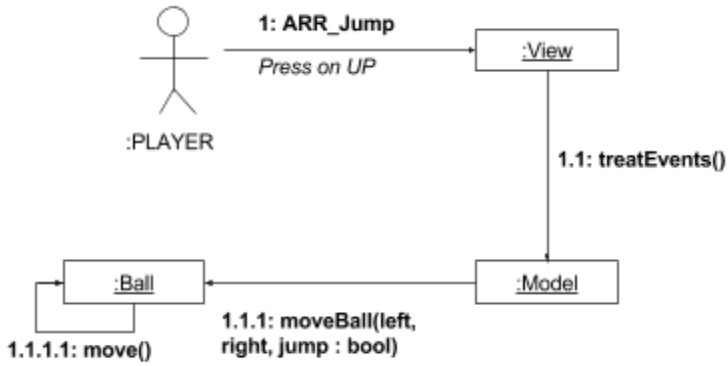
DCN de ARR_Left_Move :



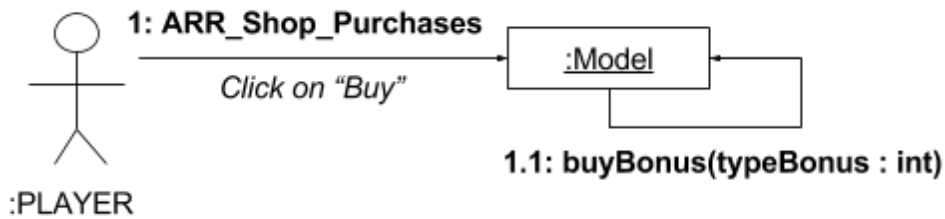
DCN de ARR_Right_Move :



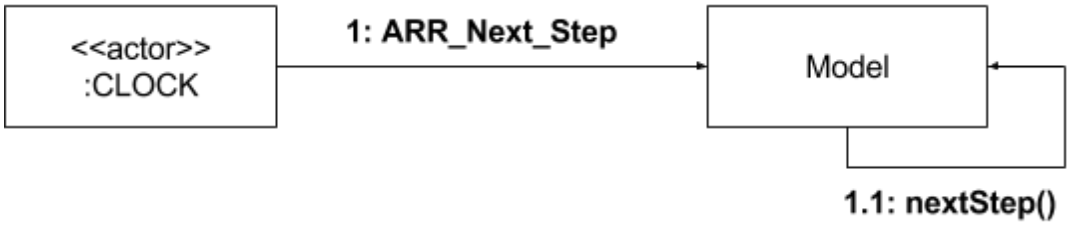
DCN de ARR_Jump :



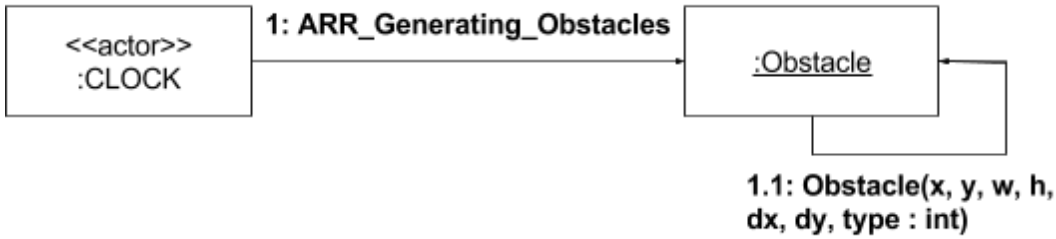
DCN de ARR_Shop_Purchases :



DCN de ARR_Next_Step :



DCN de ARR_Generating_Obstacles :



DCN de ARR_Generating_Objects :

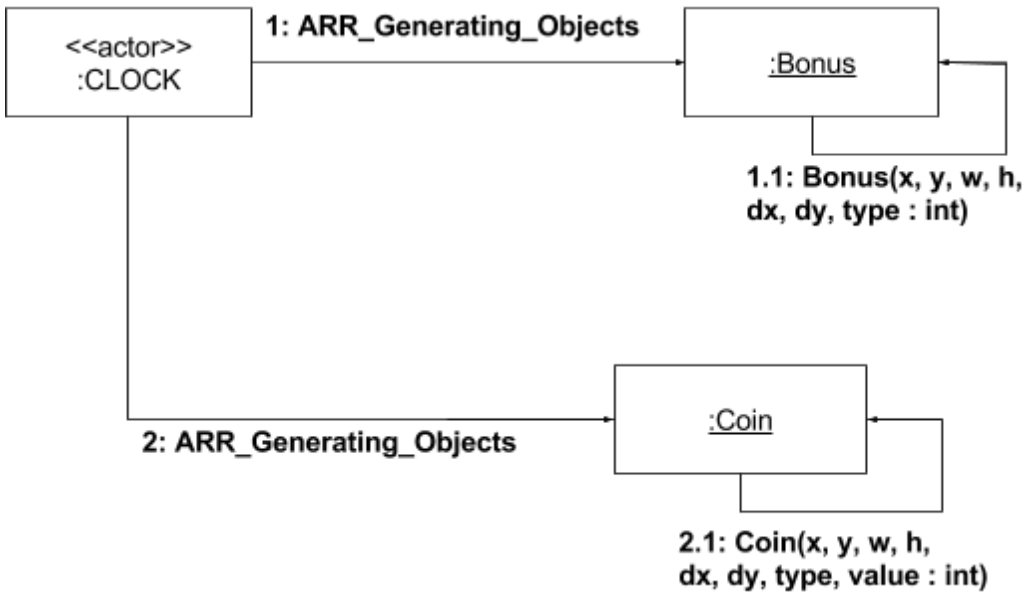
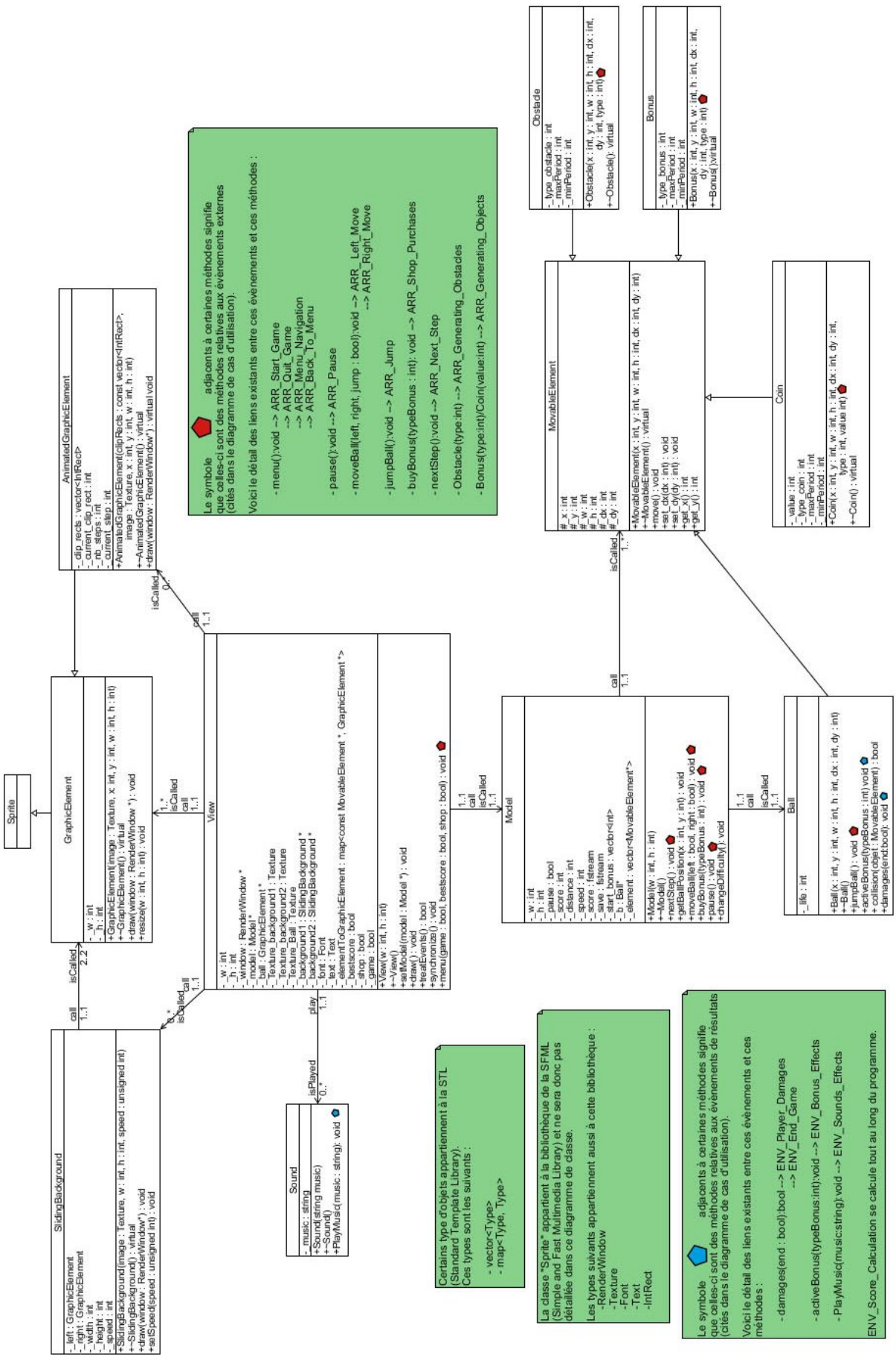


Diagramme de classe



Signature des méthodes

Class Model

Model(w : int, h : int)
~Model()
nextStep() : void
getBallPosition(x : int, y : int) : void
moveBall(left : bool, right : bool) : void
buyBonus(typeBonus : int) : void
pause() : void
changeDifficulty(): void

Class Ball

Ball(x : int, y : int, w : int, h : int, dx : int, dy : int)
~Ball()
jumpBall() : void
activeBonus(typeBonus : int) void
collision(objet : MovableElement) : bool
damages(end:bool): void

Class Movable Element

MovableElement(x : int, y : int, w : int, h : int, dx : int, dy : int)
~MovableElement() : virtual
move() : void
set_dx(dx : int) : void
set_dy(dy : int) : void
get_x() : int
get_y() : int

Class Obstacle

Obstacle(x : int, y : int, w : int, h : int, dx : int, dy : int, type : int)
~Obstacle(): virtual

Class Bonus

Bonus(x : int, y : int, w : int, h : int, dx : int, dy : int, type : int)
~Bonus():virtual

Class Coin

Coin(x : int, y : int, w : int, h : int, dx : int, dy : int, type : int, value int)
~Coin() : virtual

Class View

View(w : int, h : int)
~View()
setModel(model : Model *) : void
draw() : void
treatEvents() : bool
synchronize() : void
menu(game : bool, bestscore : bool, shop : bool) : void

Class GraphicElement

GraphicElement(image : Texture, x: int, y : int, w : int, h : int)
~GraphicElement() : virtual
draw(window : RenderWindow *) : void
resize(w : int, h : int) : void

Class SlidingBackground

SlidingBackground(image : Texture, w : int, h : int, speed : unsigned int)
~SlidingBackground() : virtual
draw(window : RenderWindow*) : void
setSpeed(speed : unsigned int) : void

Class AnimatedGraphicElement

AnimatedGraphicElement(clipRects : const vector<IntRect>, image : Texture, x : int, y : int, w : int, h : int)
~AnimatedGraphicElement() : virtual
draw(window : RenderWindow*) : virtual void

Class Sound

Sound(string music)
~Sound()
PlayMusic(music : string): void

III - Avancement du projet

Nous arrivons à la moitié du projet en termes de programmation. Nous possédons le fond en parallaxe qui défile ainsi qu'une balle pouvant se déplacer de droite à gauche.
La fonctionnalité de saut de la balle est en cours de développement tandis que nous commençons déjà à réfléchir sur la création des obstacles.

Le projet est encore loin d'être achevé, c'est pourquoi nous nous concentrerons principalement sur les fonctionnalités de base du jeu avant d'attaquer les parties complémentaires comme par exemple la boutique ou encore la musique.

IV - Annexe

Le dictionnaire de données