



RAPPORT DE PROJET

BOUNCING CAT : SITE DE JEUX EN LIGNE

Réalisé par

Clément BRES — Emmanuel BRUDY

Sacha LHOPITAL — Axel REZÉ

Sous la direction de

Madalina CROITORU

Année Universitaire 2014-2015

Remerciements

Ce rapport représente la conclusion du développement d'un projet qui fut une grande aventure. Il nous a permis de faire un pas de plus vers nos futures carrières d'informaticien. Ainsi il est l'occasion de remercier les personnes qui nous ont guidé dans ce projet.

Toute l'équipe souhaite remercier l'ensemble du personnel enseignant de l'IUT Informatique de Montpellier, car l'enseignement apporté par ces professeurs représentent les fondations de nos connaissances et donc de ce projet.

Nous remercions plus particulièrement notre tutrice de projet Madalina CROITORU qui nous a permis, en plus d'apprendre, de porter un nouveau regard sur le milieu informatique que nous avons côtoyé à ses côtés.

Sommaire

Introduction	1
1 Cahier des Charges	2
1.1 Introduction	2
1.1.1 Objectif	2
1.1.2 Cadre du projet	2
1.1.3 Destinataires	2
1.2 Description Generale	3
1.2.1 Perspective du produit	3
1.2.2 Fonctions du produit	3
1.2.3 Caractéristiques des utilisateurs	4
1.2.4 Environnement de l'application	4
1.2.5 Contraintes de conception et de fonctionnement	5
1.3 Exigences spécifiques	6
1.3.1 Exigences fonctionnelles des jeux	6
1.3.2 Exigences de l'interface	6
1.3.3 Exigences de performance	7
1.4 Fonctionnalités du Système	8
1.4.1 Accès à l'interface Web	8
1.4.2 Inscription utilisateur	8
1.4.3 Connexion utilisateur	8
1.4.4 Voir le classement	8
1.4.5 Jouer à un jeu	8
1.5 Exigences non fonctionnelles	9
1.5.1 Documentation du projet	9
1.5.2 Documentation utilisateur	9
2 Rapport Technique	10
2.1 Conception	10
2.1.1 Présentation du fonctionnement de Construct 2	10
2.1.2 Présentation du fonctionnement de JQuery Throwable	11
2.1.3 Arborescence des fichiers	12
2.1.4 Architecture du Système Principal	13
2.2 Resultats	14
2.2.1 Résultat Défoul'Cat	14
2.2.2 Résultat RainingCat	21
2.2.3 Résultat CatDefend	24
2.2.4 Résultat Développement du site	31
2.3 Tests	35
2.3.1 Tests concluants	35
2.3.2 Tests ayant permis une amélioration	35
2.4 Perspectives	36

3	Manuel d'Utilisation	37
3.1	Gestion des utilisateurs	38
3.2	Jouer	40
4	Rapport d'Activité	41
4.1	Présentation des outils utilisés	41
4.1.1	Trello	41
4.1.2	ShareLaTeX	42
4.1.3	FileZilla	42
4.1.4	Bootstrap	43
4.2	Cycle de développement	44
4.3	Planification	45
	Conclusion	46
A	Dossier d'Analyse	II

Table des figures

1.1	Schéma de la Base de Données	3
1.2	Diagramme de Cas d'Utilisation	4
1.3	Visuel de l'interface initialement prévu	7
2.1	Visuel de Construct2	10
2.2	Arborescence de l'Application	12
2.3	Architecture du Système Principal	13
2.4	Illustration de l'objectif à atteindre pour le jeu Defoul'Cat	14
2.5	Affichage esthétique Defoul'Cat	15
2.6	Defoul'Cat Version I	18
2.7	Defoul'Cat Version II	20
2.8	Paramètres du Chat CatDefend	25
2.9	Évènement : "Saut" CatDefend	25
2.10	Évènement : "Tir du laser" CatDefend	25
2.11	Évènement : "Déplacement du chien" CatDefend	26
2.12	Évènement : "Apparition des chiens" CatDefend	26
2.13	Évènement : "Miaoumeha" CatDefend	27
2.14	Évènement : "Bouclier" CatDefend	27
2.15	Évènement : "Bouclier" (Problème résolu) CatDefend	28
2.16	Évènement : "Laser et Score" CatDefend	28
2.17	Évènement : "Baisse du score" CatDefend	28
2.18	Évènement : "Mort du Héros" CatDefend	29
2.19	Exemple de Layout CatDefend	29
2.20	Évènement : "Son sur le Miaoumeha" CatDefend	30
2.21	Évènement : "Son sur le laser" CatDefend	30
2.22	Évènement : "Animation du Héros" CatDefend	30
2.23	Plan du site	32
2.24	score de Raining	33
3.1	Formulaire de connexion	37
3.2	Formulaire d'ajout d'un utilisateur	38
3.3	Interface de modification / suppression d'un utilisateur	38
3.4	Formulaire de mise à jour d'un utilisateur	39
3.5	Fiche du jeu DefoulCat avec lien de jeu	40
4.1	Impression d'écran du Trello au 75% du projet	42
4.2	Impression d'écran de cette page du rapport sur ShareLaTeX	42
4.3	Planning prévisionnel	45

Glossaire

Les termes définis dans ce glossaire sont identifiables dans le corps du texte au moyen d'un astérisque (). Il est accessible à tous.*

Bug : Est considéré comme un bug un défaut de conception d'un programme informatique à l'origine d'un dysfonctionnement.

Behavior : (Dans Construct 2) Un behavior permet de définir le comportement d'un sprite* dans le layer*, c'est-à-dire la façon dont il va interagir avec l'utilisateur.

Code source : Le code source est un ensemble d'instructions écrites dans un langage informatique intelligible par l'homme, permettant la lecture et l'affichage par une machine.

Drag and Drop : Le glisser-déposer est une méthode consistant à utiliser une souris, pour déplacer d'un endroit à un autre un élément graphique présent sur l'écran d'un ordinateur.

Event Sheet : (Dans Construct 2) Un « Event Sheet » est une page d'événement qui permet de créer des événements d'interaction entre les différents acteurs du jeu (entre l'utilisateur et le sprite* du héros, entre le sprite* du laser et celui des chiens,...)

Layout : (Dans Construct 2) Un layout est l'équivalent d'un niveau. Il possède sa propre page d'événement (Event Sheet*). Un jeu est composé de plusieurs layouts ayant chacun sa fonction (layout de menu, de jeu,...).

Layer : (Dans Construct 2) Un layer est une couche contenue dans un layout*. Il permet de créer différents « niveaux » dans le but d'afficher des éléments en dessous ou au-dessus d'autres éléments.

MVC : De son nom complet **model-view-controller** est un "patron" de développement qui permet de faciliter le développement et la compréhension du système principal*.

Navigateur : Un navigateur est un logiciel qui permet de consulter et d'exploiter les ressources du web et les ressources Internet dans son ensemble. Parmi les navigateurs les plus connus, on retrouve par exemple Firefox, Internet Explorer ou Google Chrome.

Plugin : En informatique, un plugin est un "paquet" qui complète un logiciel hôte pour lui apporter de nouvelles fonctionnalités.

Px : (Abréviation de Pixel) Il s'agit d'un point de couleur sur l'écran.

Sprites : Un sprite est un élément tangible (images, ...) ou non (clavier, souris,...) qui interagit avec l'utilisateur d'une ou plusieurs façons.

Système principal : Est désigné par "Le système principal", les différents jeux et la gestion des utilisateurs.

Variable Globale : (Dans Construct 2) Une variable globale est une variable qui est déclarée sur tout un programme et qui peut être utilisée sur plusieurs layouts*.

Introduction

Depuis le lancement d'internet, de nombreuses plate-formes de jeux en ligne sont apparues. Tous ces sites perdurent dans le temps car ils proposent sans cesse de nouveaux jeux, plus amusant que les précédents. Les joueurs utilisent généralement ce genre de site pour passer l'ennui.

Dans ce contexte, la commande consistait à réaliser une plate-forme du même type que celles présentes sur la toile.

Ce projet a ainsi pour but la conception et la réalisation d'un site internet regroupant plusieurs jeux avec pour thème principal les chats.

Le plan du rapport est le suivant : On travaillera d'abord sur les informations nécessaires au bon départ du projet avec le cahier des charges complet notamment en analysant l'existant. Ensuite nous détaillerons le fonctionnement précis des applications créées avec les outils utilisés, ceci dans le rapport technique. Puis nous détaillerons le manuel d'utilisation qui répondra à toutes les questions sur la gestion du site par son administrateur : créer un compte pour jouer et gérer les utilisateurs. Enfin un rapport d'activité rendra compte du déroulement du développement et de la gestion sur ce projet côté organisation.

Chapitre 1

Cahier des Charges

1.1 Introduction

1.1.1 Objectif

Cette partie présente les spécifications du projet. L'objectif de ce projet est de réaliser une plate-forme regroupant un certain nombre de jeux pour que chacun puisse s'amuser en ligne.

Ce document peut être utilisé pour poursuivre le projet actuel.

1.1.2 Cadre du projet

Comme **Bouncing Cat** a fait l'objet d'un certain temps de développement, il est possible qu'une partie du cahier des charges face écho au manuel d'utilisation.

Certaines informations techniques ont été incluses, d'autres non, les lecteurs peuvent se référer au glossaire disponible en début de document pour trouver les définitions des termes spécifiques.

1.1.3 Destinataires

Ce cahier des charges examine les capacités actuelles du projet et les fonctionnalités déjà implémentées. Il établit également les lignes directrices pour un futur développement.

Il est à noter que les stratégies évoquées ont servi de base aux tests effectués plus tard.

Pour ces deux raisons, le cahier des charges suivant peut-être lu par des développeurs, des testeurs et des utilisateurs de l'application.

1.2 Description Generale

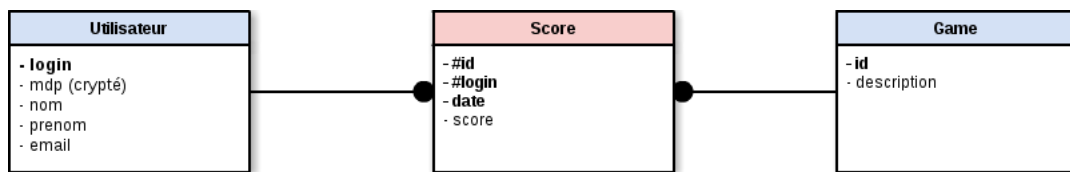
Cette section donnera un aperçu de l'ensemble du système. Son contexte sera développé pour montrer comment celui-ci interagit avec les utilisateurs et les fonctionnalités principales seront introduites. Enfin, les contraintes et les hypothèses du système seront également présentées.

1.2.1 Perspective du produit

Le site internet va devoir être capable de récupérer les données des différents jeux. Chaque jeu fonctionne de façon indépendante des autres. Pour gérer ces données, une base de données devra être utilisée pour stocker ou récupérer les scores des joueurs en fonction du jeu. Toutes les communications avec la base de données se feront via le navigateur* du joueur.

Le système de notre projet comporte deux acteurs et deux systèmes qui coopèrent. Le joueur et l'administrateur seront nos seuls acteurs. Le système principal* ainsi que la base de données sont les deux systèmes qui coopèrent. Chaque acteur accède au système par Internet et communique avec la base de données via l'interface web. Le schéma de la base de données se trouve ci-après.

FIGURE 1.1 – Schéma de la Base de Données



Dans la suite du cahier des charges, les acteurs seront désignées par respectivement le terme joueur et le terme administrateur. Le système de notre projet sera désigné par le terme application.

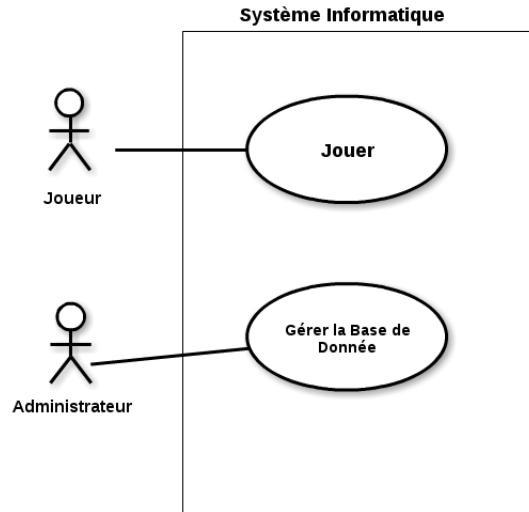
Après débat par l'ensemble de l'équipe, il a été décidé de développer trois jeux principaux dans l'application :

- **Defoul'Cat** : Le chat est seul sur l'espace de jeu. Le joueur peut envoyer le chat sur les bords de l'espace. Ce jeu ne présente pas de système de score. C'est un jeu pour se détendre.
- **RainingCat** : Une image de chat tombe du haut de l'écran par gravité. Le joueur doit l'attraper avec la souris. Le but est de ne pas laisser le chat toucher le sol. Au fil du temps, un autre chat apparaît, il y a donc deux chats que l'on ne doit pas laisser tomber. Et ainsi de suite jusqu'à ce que le joueur perde. Le temps de vie du joueur est alors enregistré comme score.
- **CatDefend** : Un chat est attiré par la gravité et à la possibilité de tirer en face de lui. Le joueur peut le mouvoir et déclencher des effets avec son clavier et sa souris. En face du chat, plusieurs ennemis (sous forme de vagues) se dirige vers le joueur pour le faire perdre. Celui-ci doit tirer sur ses adversaires.

1.2.2 Fonctions du produit

L'application ne doit offrir qu'une fonctionnalité essentielle pour le joueur : jouer aux jeux. Le système principal* doit interagir avec le joueur pour récupérer les scores qu'il effectue et les transmettre à la base de données. Il doit aussi gérer les cas d'erreurs possibles : le système principal* ne se synchronise avec la base de données que dans des situations cohérentes.

FIGURE 1.2 – Diagramme de Cas d'Utilisation



Comme le montre le diagramme ci-dessus, l'application doit avant tout permettre de jouer pour le joueur et de gérer la base de données des jeux pour l'administrateur.

D'autres objectifs secondaires sont présents comme par exemple la navigation sur le site ou la gestion des inscriptions et des comptes.

L'ensemble de l'analyse réalisée est disponible en Annexe A.

1.2.3 Caractéristiques des utilisateurs

Le joueur interagit avec l'application via une interface web. Cette interface doit être la plus simple d'utilisation possible et accessible.

L'administrateur interagit également avec l'application par l'interface. Il doit pouvoir facilement maintenir l'ensemble du système. Il est le seul capable de mettre à jour la base de données, par exemple si un utilisateur a perdu son mot de passe.

1.2.4 Environnement de l'application

Bouncing Cat est une application fonctionnant avec un navigateur* et une connexion internet. L'application a été conçue pour être utilisée uniquement sur un ordinateur. Elle fonctionne théoriquement sur tablette et smart-phone, mais les jeux ne sont pas assez fluide pour que l'expérience du joueur soit acceptable.

L'application a été testée sur Google Chrome, Internet Explorer, Opéra et Firefox.

Aucune compétences en informatique n'est nécessaire pour faire fonctionner l'application.

1.2.5 Contraintes de conception et de fonctionnement

Bouncing Cat est une application développée en *PHP* et en *SQL* avec une interface écrite en *HTML-CSS* et en *JavaScript*.

L'interface web de l'application impose de fortes contraintes au projet. En effet, il faut gérer les différents navigateurs qui existent. La connexion Internet est aussi une exigence pour l'application. Cette connexion est nécessaire à la bonne interaction avec la base de données et au bon fonctionnement des jeux. Sans connexion, l'application ne pourra pas fonctionner.

Une connexion Internet stable est requise pour que l'application soit efficace. Si la connexion est peu stable ou pas assez puissante, il se peut que le joueur rencontre des difficultés pour jouer. Accéder à l'application sur un navigateur* à jour est également fortement conseillé pour éviter tout problème d'affichage.

1.3 Exigences spécifiques

1.3.1 Exigences fonctionnelles des jeux

Un joueur n'a pas besoin de se connecter pour accéder aux jeux. L'administrateur par contre a besoin de se connecter pour gérer la base de données. Que l'utilisateur soit connecté ou non, une page par défaut doit s'afficher.

Exigences Communes à (presque) tous les jeux

- *Créer les jeux avec une bibliothèque JQuery* : c'est non seulement une facilité technique, mais également un réel choix. En effet, développer directement en JavaScript nous demanderait un apprentissage personnel trop long. De plus, le JQuery est une des bibliothèques les plus couramment utilisées en JavaScript. Ici, JQuery est clairement la première grosse difficulté technique que l'on peut prévoir. Il faudra l'utiliser proprement et comprendre les différentes méthodes utilisées.

Exigences Defoul'Cat

- *Faire rebondir le chat* : le chat doit pouvoir être lancé et rebondir en fonction de l'endroit où on le lance. Pour cela, il doit subir les lois de la gravité et de la physique.
- *Le chat doit rebondir dans un endroit précis* : Il s'agit ici d'implémenter un *cadre de jeu* qui permet de garder le chat dans une zone prédéfinie. Cette zone peut être l'écran lui-même ou une zone plus petite.

Exigences RainingCat

- *Augmentation de la difficulté* : un premier chat apparaît au début du jeu. Après un laps de temps prédéfini un autre chat doit apparaître dans ce même espace de jeu. Le laps de temps suivant, un troisième apparaît et ainsi de suite. Plus le nombre de chat augmente, plus la difficulté augmente.
- *Perdre la partie* : Quand un chat touche le sol, le joueur a perdu. L'évènement s'applique aussi bien au premier chat, qu'à tous les autres chats apparus.
- *Enregistrer son score* : Le score obtenu est enregistré dans la base de données grâce à l'interface. Mais dans le code source* du jeu, il faut gérer un score qui s'augmente dans le temps et s'affiche à côté de la zone de jeu. Plus le joueur joue longtemps sans perdre, plus le score augmente. Quand le joueur a perdu, le score s'arrête. Celui-ci ne peut pas diminuer.

Exigences CatDefend

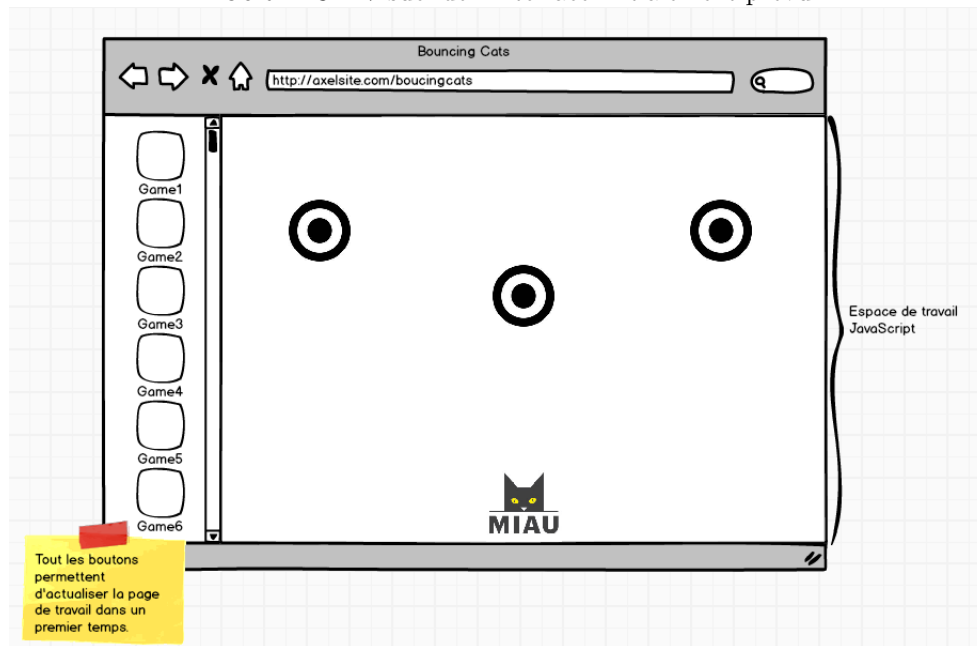
- *Gestion des ressources* : Est appelé "ressources" tous les éléments du jeu nécessaires à son fonctionnement : les ennemis sous forme de chiens par exemple, les sons, l'image de fond, etc.
- *Gestion des attaques & bonus* : Le chat possède un (ou plusieurs) bonus pour se défendre et pour attaquer. Le joueur utilise son clavier pour effectuer ces actions.
- *Perdre la partie* : Quand le chat sort de la zone de jeu ou quand il rentre en contact avec un ennemi, le joueur a perdu.
- *Enregistrer son score* : idem que pour *RainingCat*.

1.3.2 Exigences de l'interface

L'interface disponible sur le web doit permettre à l'utilisateur de naviguer d'un jeu à l'autre le plus facilement possible. L'administrateur va utiliser la même interface bien que son but ne soit pas identique.

L'interface ci-dessous représente l'idée initiale de son design. Depuis, pour des questions techniques que nous évoquerons plus tard, la zone de jeu a été définie sur une nouvelle page.

FIGURE 1.3 – Visuel de l'interface initialement prévu



1.3.3 Exigences de performance

Exigences du site internet

- *Simplicité d'utilisation* : que ce soit du côté administrateur ou du côté joueur, le site doit être le plus simple possible d'utilisation.
- *Facilité de navigation* : la facilité de navigation du site est étroitement liée à la simplicité d'utilisation.
- *Rapidité dans le temps de chargement* : les utilisateurs sont de plus en plus exigeants avec internet. Il faut donc adapter notre site internet pour que les réponses soient données le plus rapidement possible.

Exigences des jeux

- *Rapidité dans le temps de chargement* : de la même façon que pour le site internet, il est nécessaire que les jeux ne soient pas trop long au chargement. Le joueur doit pouvoir jouer de façon fluide à chaque jeu (dans la mesure où il respecte les contraintes de fonctionnement).

1.4 Fonctionnalités du Système

1.4.1 Accès à l'interface Web

L'accès à l'interface Web doit être possible depuis le plus grand nombre possible d'appareils : téléphones, tablettes, ordinateurs tous systèmes d'exploitation. Cependant il est à noter que le JavaScript n'offre pas les mêmes avantages suivant l'équipement informatique utilisé.

1.4.2 Inscription utilisateur

L'utilisateur doit pouvoir s'inscrire sur le site. Son inscription entraîne un enregistrement dans la Base de Données.

1.4.3 Connexion utilisateur

Une fois que l'utilisateur est inscrit, il peut se connecter avec son compte. Une fois qu'il est connecté, les scores réalisés sont enregistrés grâce à son identifiant. La connexion est permise en fonction des informations contenues dans la Base de Données.

1.4.4 Voir le classement

Tout utilisateur, connecté ou non, peut accéder à la liste des scores via l'interface Web. La Base de Données permet l'affichage des scores sur l'interface.

1.4.5 Jouer à un jeu

Tout utilisateur, connecté ou non, peut accéder aux informations des jeux via l'interface Web. Ils peuvent également jouer à tous les jeux. Dans ce dernier cas, le score est enregistré sous le pseudo "non connecté". L'enregistrement du score s'effectue dans la Base Données.

1.5 Exigences non fonctionnelles

1.5.1 Documentation du projet

Le projet doit être documenté de deux façons différentes, mais qui se complètent.

Dans un premier temps, le code de programmation du jeu doit être commenté de façon à pouvoir comprendre le fonctionnement de chaque jeu programmé, et le site web.

Dans un second temps, un document expliquant le fonctionnement de l'ensemble du projet, son but et comment il a été réalisé, doit être prévu.

1.5.2 Documentation utilisateur

Une documentation pour les joueurs doit être mise en place en passant par l'explication des règles de chaque jeu directement sur le site web.

Cependant, un autre document doit aussi être mis à disposition des administrateurs pour apporter détails sur le projet et réponses à ses questions.

Chapitre 2

Rapport Technique

Dans cette partie on détaillera le fonctionnement réel du site et de tous ces composants. Dans la partie précédente on a pu voir toutes les exigences y compris celles qui n'ont pas encore pu être développées. Si le développement du site reprend et que quelqu'un est amené à retravailler dessus, c'est cette partie que nous lui conseillons de lire en priorité.

2.1 Conception

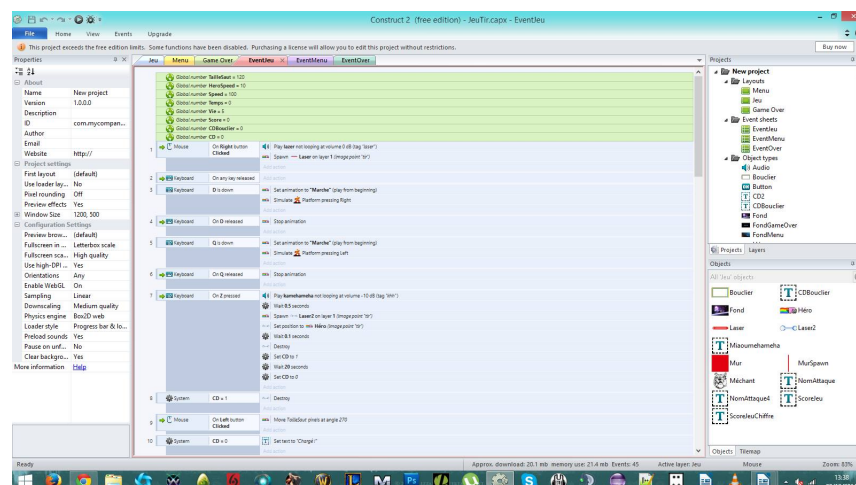
Pendant la réalisation du projet, certaines contraintes techniques rendaient nécessaire l'utilisation de plugin* ou logiciels pré-existant. Ces aides à la bonne réalisation du projet ont imposé un certain type d'apprentissage différent de l'apprentissage d'un langage classique.

Ainsi, CatDefend a été conçu avec le logiciel Construct 2 ; Defoul'Cat et RainingCat ont été développés sur la base du plugin* JQuery Throwable.

2.1.1 Présentation du fonctionnement de Construct 2

Construct 2 est un programme qui nous permet de créer des jeux en HTML5 facilement à l'aide d'un environnement de développement de type drag and drop*. La plupart des outils du programme peuvent être utilisés à partir de l'interface graphique.

FIGURE 2.1 – Visuel de Construct2



Construct 2 est une aide pour la création de jeux 2D et comprend beaucoup de ressources qui facilitent la tâche, comme une interface qui permet de paramétrer son jeu de façon simple, etc. De plus, il est très simple d'ajouter n'importe quel fichier médias externe (sons, images).

Le logiciel est basé sur la fonctionnalité HTML5 des *canvas*. Cette fonctionnalité se présente sous la forme de balises de tailles variables. On applique ensuite à ces balises des fonctions précises qui permettent de créer le jeu.

Pour plus de détails sur son fonctionnement, se référer à la section 2.2.3

2.1.2 Présentation du fonctionnement de JQuery Throwable

JQuery Throwable est un plugin* JavaScript : il permet donc nativement l'utilisation d'un certain nombre de fonctions déjà implantées comme la gestion de la gravité et des collisions.

JQuery Throwable est composé de trois grandes fonctions :

- *L'initialisation* qui génère le moteur physique au travers d'un monde virtuel et de différentes boîtes. Ce moteur utilise un autre plugin* sur lequel nous ne nous étendrons pas : Box2D.
- *\$.fn.throwable* qui est l'équivalent du programme principal. C'est à cet endroit que l'on indique les différents paramètres.
- Et une fonction toute particulière *throwable*, elle-même composée de sous-fonctions essentielles. Certaines de ces sous-fonctions ont été indispensables à la réalisation des différents jeux (cf. 2.2.1 et 2.2.2) :

Quelques exemples de ces sous-fonctions :

default :

```

1      defaults: {
2          infinitX: false,
3          gravity: {x: 0, y: 0},
4          bounce: 0,
5          containment: "window",
6          autostart:true,
7          shape: "box",
8          impulse:null,
9          fixed: false,
10         drag: true,
11         damping:0,
12         collisionDetection:false,
13         areaDetection: []
14     },

```

Celle-ci est très importante. C'est une initialisation nécessaire pour modifier éventuellement les paramètres de base. Ainsi les valeurs ci-dessus sont des valeurs par défaut.

applyOption :

applyOptions :function(o,i) est une fonction qui permet de fusionner aux paramètres par défaut, présentés ci-dessus, les caractéristiques de *o* dans un ordre *i*.

Concrètement, cette fonction permet de fusionner toutes les options entrées par l'utilisateur avec les options de base. Par exemple, en indiquant l'option "bounce : 1", l'option par défaut "bounce : 0" est remplacée par la précédente. Si l'option n'existe pas dans la variable *this.default*, elle sera créée.

setEnv :

Cette fonction très complète va enregistrer la taille du navigateur* de l'utilisateur à l'aide de la fonction **getBrowserDimensions()**.

Puis elle va ajuster la taille de l'espace pour rebondir en fonction du navigateur* et de la taille de l'écran. C'est également elle qui crée les *murs* invisibles qui permettent le rebond. Ces murs sont créés sous la forme de boîtes très fines avec Box2D. L'intérêt est de créer physiquement les murs pour pouvoir appliquer les collisions, l'amortissement et tous les concepts associés à la physique de Box2D.

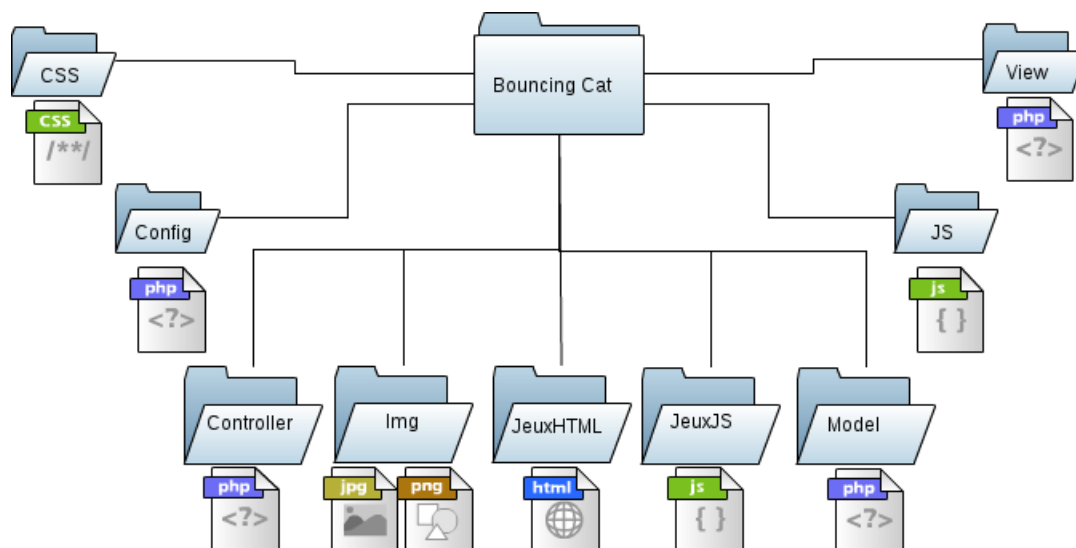
On pourrait ainsi définir des interactions avec ces murs quand la souris de l'utilisateur est à une certaine place, ou quand une collision a lieu.

Au départ, pour résoudre le problème de la gravité (évoqué dans la section 2.2.1), le plugin* **JGravity** devait être utilisé. Mais très vite il est apparu que JGravity est un excellent module pour gérer la gravité sur plusieurs éléments d'une même page, mais il est très limité quand aux possibilités de modification. Par exemple, impossible de définir une "zone" de gravité plus petite que le navigateur. Impossible également d'appliquer la gravité à un unique élément.

Dès les premiers essais des problèmes majeurs sont survenus et après quelques semaines de travail sur ce plugin*, il était clair que l'ensemble du module était sujet à modification pour notre utilisation. Pour ne pas perdre trop de temps et pouvoir implémenter rapidement les premiers jeux, l'utilisation d'un autre module JavaScript qui est plus facilement modifiable s'est imposé : JQuery Throwable.

2.1.3 Arborescence des fichiers

FIGURE 2.2 – Arborescence de l'Application



CSS contient l'ensemble des fichiers relatifs à l'utilisation de Bootstrap pour l'aspect visuel de l'application ainsi que les fichiers CSS des jeux.

Config, Controller, Model et View contiennent les différents fichiers nécessaires à l'interaction Base de Données - Système principal*. Ces fichiers sont structurés selon le modèle MVC* de programmation PHP.

Img contient toutes les images affichées dans les jeux.

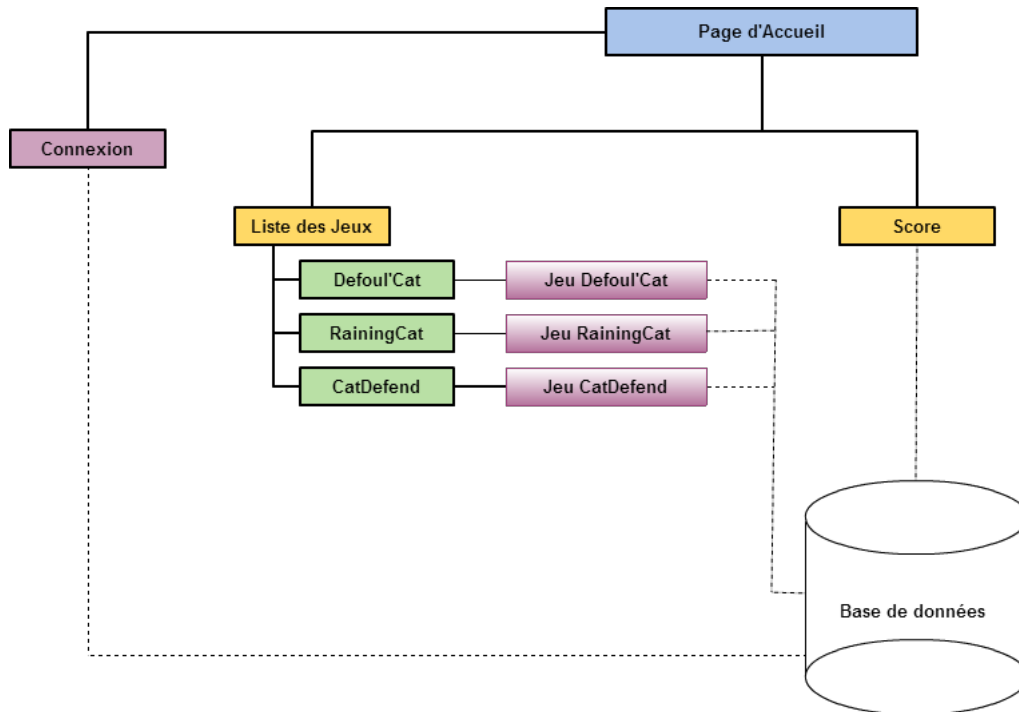
JeuxHTML contient un fichier de code HTML pour chaque jeu. Ce code permet l'affichage du jeu proprement dans une nouvelle page.

JeuxJS contient un fichier de code JavaScript pour chaque jeu.

JS est un répertoire particulier qui contient entre autres le code JavaScript de JQuery Throwable.

2.1.4 Architecture du Système Principal

FIGURE 2.3 – Architecture du Système Principal



N'importe quelle personne peut avoir accès à la liste des jeux sur la page d'accueil et peut donc y jouer. Les scores des personnes non-enregistrées sont stockés en mémoire dans la base de données avec un pseudo générique aux personnes déconnectées.

Les personnes non-inscrites peuvent s'inscrire et se connecter (ou seulement se connecter si elles sont déjà inscrites). De cette manière, elles peuvent jouer et ce sera leur pseudo qui sera mémorisé dans le score qu'ils feront dans le jeu.

2.2 Resultats

2.2.1 Résultat Défoul’Cat

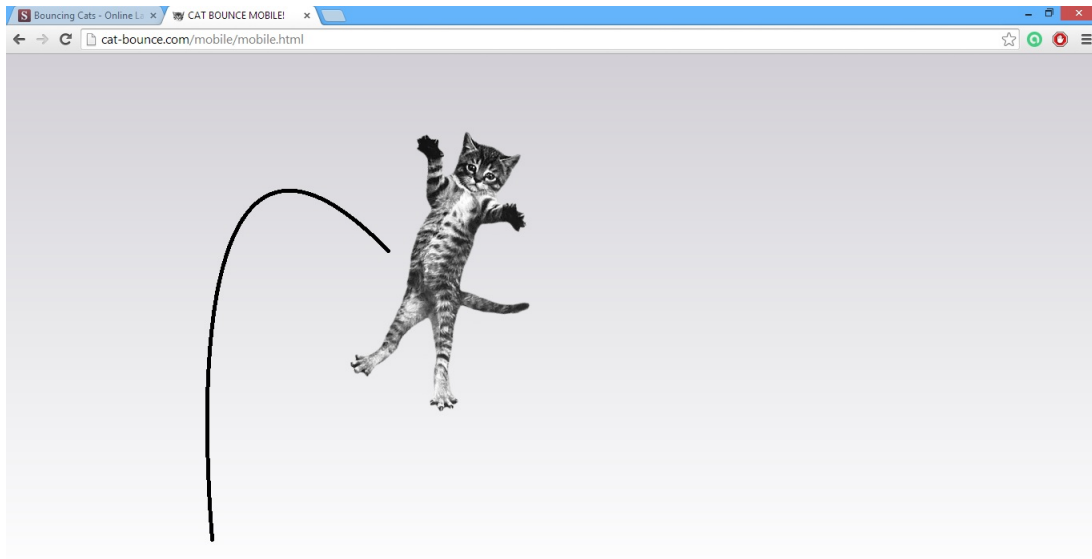
Introduction

Demande du tuteur : Réaliser un *Bouncing Cat* où un chat doit rebondir. On doit pouvoir le lancer, jouer avec et le tenir en l’air. Ce développement fait l’objet de deux fonctionnalités principales (cf. 1.3.1) :

Faire rebondir le chat

Le visuel suivant, tiré d’un autre site (cf. Références), donne un exemple du but à atteindre :

FIGURE 2.4 – Illustration de l’objectif à atteindre pour le jeu Defoul’Cat



Le chat doit rebondir dans un endroit précis

Nous avons au préalable, réalisé une rapide ébauche HTML du jeu dont voici le code source* :

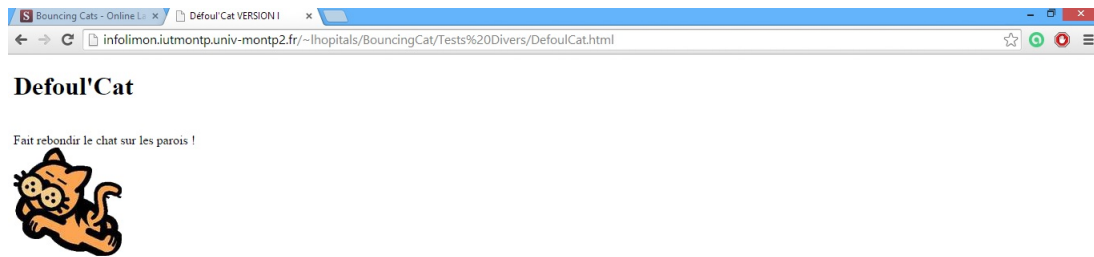
```

1  <body>
2
3      <h1> Defoul'Cat </h1>
4      <br>
5      Fait rebondir le chat sur les parois !
6      <br>
7
8
9      <div class="balls1"></div>
10     //correspond a l'image du chat. C'est a cette element que l'on appliquera le
        code jQuery
11
12     <script src="js/libs/jquery-1.9.0/jquery.min.js" type="text/javascript">
        </script>
13     //Import de la librairie jQuery
14
15     <script src="http://benahm.github.io/jquery.throwable/jascripts/
        jquery.throwable.js" type="text/javascript"></script>
16     //Import de la librairie throwable
17
18     <script src="DefoulCat.js"></script>
19     //Import du code Javascript
20
21 </body>

```

Ce qui nous donne une page *Defoul'Cat.html* affichant le résultat suivant :

FIGURE 2.5 – Affichage esthétique Defoul'Cat



A cet instant, dans le développement du projet, le chat est statique.

Mise en oeuvre

Gravité

Avec JQuery Throwable, on peut recréer un effet de gravité sur des éléments HTML.

On fait appel à ce plugin dans le code source du jeu *Defoul'Cat.js* :

```
1      $(".balls1").throwable({
2          gravity:{x:0,y:0}
3      });
```

Ce code interagit avec le code HTML (cf. 2.2.1) contenu dans *Defoul'Cat.html*. Dans le code de *Defoul'Cat.js*, on applique à l'élément *balls1* la fonction *Throwable* avec une gravité classique (l'élément est attiré vers le bas verticalement).

Dans le code de *Defoul'Cat.html*, c'est l'élément *balls1* qui établit la connexion avec le code JavaScript précédent.

Dans le code JQuery *Throwable*, l'option **gravity** permet d'appliquer une fonction **applyGravity : function()**. Celle-ci utilise des fonctions récurrentes de *Box2D* qui gère la masse virtuelle de l'objet et sa position en temps réel pour créer un vecteur qui permet le mouvement.

Rebond

Mais il ne faut pas uniquement appliquer de la gravité à l'élément. Il faut également qu'il puisse rebondir. Le code source* s'adapte donc comme suit :

```
1      $(".balls1").throwable({
2          gravity:{x:0,y:0},
3          bounce:0.5
4      });
```

JQuery *Throwable* récupère alors un nouveau paramètre **bounce** qui permet de faire varier la puissance du rebond. 0 correspond à aucun rebond, 1 correspond à un rebond d'une puissance maximale.

Tout comme le paramètre **gravity**, le paramètre **bounce** permet de modifier une fonction de JQuery *Throwable* : **createBox : function(world, x, y, width, height, fixed, categoryBits, maskBits, element) & createCircle : function(world, x, y, radius, fixed,categoryBits, maskBits, element)** Ces deux fonctions créent respectivement une boîte ou un cercle avec le plugin *Box2D*. Il faut utiliser uniquement l'une ou l'autre de ces deux fonctions selon la forme que l'on souhaite. *fixed* est un paramètre qui correspond à la densité de la boîte ou du cercle. En fait, la valeur du paramètre **bounce** va entrer en compte dans la création du monde, en collaboration avec la densité.

Intéraction avec le joueur

Maintenant que le chat subit la gravité et qu'il rebondit, il faut que l'utilisateur puisse le mouvoir avec sa propre souris :

```
1      $(".balls1").throwable({
2          drag: true,
3          gravity:{x:0,y:0},
4          bounce:0.5
5      });
```

On rajoute un paramètre **drag : true** qui sera transmis à JQuery *Throwable*. Dans celui-ci, la fonction **mouseDrag : function()** regroupe tous les événements concernant le mouvement possible de la souris de l'utilisateur quand il clique (ou laisse appuyé). Détaillons la fonction :

```

1      // Si on presse le bouton
2      if (this.isMouseDown && !this.mouseJoint) {
3
4          var body = this.getBodyAtMouse();
5
6          if (body) {
7
8              var md = new b2MouseJointDef();
9              md.body1 = world.m_groundBody;
10             md.body2 = body;
11             md.target.Set(this.mouse.x + window.scrollX, this.mouse.y +
                window.scrollY);
12             md.maxForce = 30000.0 * body.m_mass;
13             md.timeStep = this.defaults.timeStep;
14             this.mouseJoint = world.CreateJoint(md);
15             body.WakeUp();
16         }
17     }
18
19     // Si on relache le bouton
20     if (!this.isMouseDown) {
21
22         if (this.mouseJoint) {
23
24             world.DestroyJoint(this.mouseJoint);
25             this.mouseJoint = null;
26         }
27     }
28
29     // Si on bouge la souris
30     if (this.mouseJoint) {
31
32         var p2 = new b2Vec2(this.mouse.x + window.scrollX, this.mouse.y +
            window.scrollY);
33         this.mouseJoint.SetTarget(p2);
34     }
35 }

```

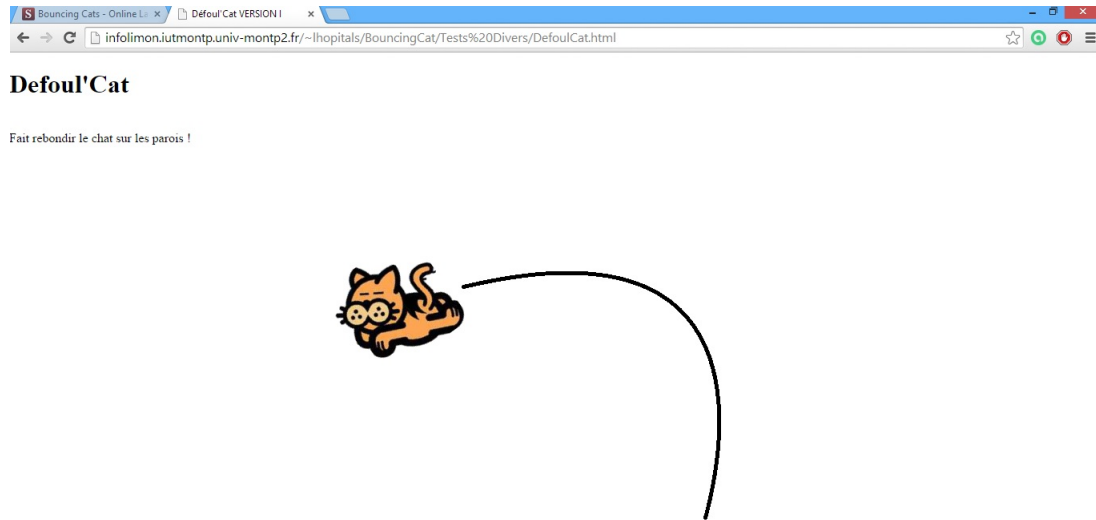
Si on presse le bouton de la souris, et qu'on n'est pas déjà le bouton enfoncé, la fonction enregistre la position exacte de la souris à ce moment dans une variable *body* avec la sous-fonction **getBodyAtMouse**. Puis si la position de la souris a été enregistrée correctement, on utilise des fonctions de Box2D pour appliquer une force très grande (ligne 12). Si la force n'est pas assez importante par rapport au poids de *balls1*, on ne pourra pas déplacer l'élément dans le monde virtuel créé par Box2D. Enfin, ligne 13, on crée une "articulation" entre la souris et *balls1*. **This.mouseJoint** est une variable qui récupère l'articulation. Ce qu'on nomme une *articulation*, est un ensemble de données qui indique qu'il existe un lien entre un élément et un autre. Dans ce cas, on crée un lien entre la souris et l'élément sur lequel elle clique.

Si la souris est en mouvement, tout en pressant le bouton, on utilise Box2D pour enregistrer un vecteur de déplacement et on modifie *this.mouseJoint* en fonctions de la nouvelle position de la souris.

Si on relâche le bouton, l'articulation est détruite.

Le résultat actuel est presque la version finale de *Defoul'Cat* : (le tracé noir représente le mouvement du chat).

FIGURE 2.6 – Defoul’Cat Version I



Définition de la zone de jeu

Il faut maintenant créer un **cadre de jeu** pour éviter que le chat ne rebondisse partout. Pour arriver à ce résultat, il a été nécessaire de modifier le code JavaScript ainsi que le code HTML :

Code JavaScript

```

1      $(".balls1").throwable({
2          containment: "parent",
3          drag: true,
4          gravity:{x:0,y:0},
5          bounce:0.5
6      });

```

Ici, ligne 2, il a été rajouté un paramètre **containment**. Celui-ci intervient dans JQuery Throwable et dans le HTML.

Dans le plugin*, **containment** est une condition de la fonction **getBrowserDimensions : function()**. Celle-ci permet d'obtenir les dimensions de la zone de jeu :

```

1      if (this.defaults.containment === "parent") {
2          var _this = this;
3          var parent = $(this.elements).parent();
4          var offset = parent.offset();
5
6          this.stage.X = offset.left;
7          this.stage.Y = offset.top;
8
9          this.stage.Width = parent.outerWidth() + offset.left;
10         this.stage.Height = parent.outerHeight() + offset.top;
11
12         });

```

Si le paramètre **containment** vaut *"parent"*, on récupère les propriétés CSS du parent de *balls1* grâce à la commande de la ligne 4. Plus précisément :

```
1    var parent = $(this.elements).parent();
```

`$(this.elements).parent()` pourrait se traduire dans ce cas par `$(.balls1).parent()`. On récupère dans la variable **parent** les propriétés CSS de l'élément parent de `balls1`. Cet élément parent peut être un autre élément contenant `balls1`, ou bien, en l'absence d'indication dans le code HTML, le navigateur*.

Dans la variable **offset**, ligne 5, on récupère précisément les dimensions du parent de `balls1`.

Puis, ligne 7 à 10, on définit les coordonnées de **this.stage** par les coordonnées de l'élément parent. **this.stage** est une variable essentielle au plugin* : c'est elle qui définit la *zone d'action* de JQuery Throwable grâce à des coordonnées :

```
1    this.stage = {X: window.screenX, Y: window.screenY, Width: window.innerWidth,
                  Height: window.innerHeight};
```

Au début de la fonction, elle est initialisée comme faisant tout l'écran (paramètres `window.screenX/Y` et `window.innerWidth/Height` qui retournent les coordonnées du début de l'écran, ainsi que sa largeur et de sa hauteur). Ainsi cette zone est visible dans l'espace.

Code HTML

```
1    <body>
2
3        <h1> Defoul'Cat </h1>
4        <br>
5        Fait rebondir le chat sur les parois !
6        <br>
7
8        <div class="parent">
9            // correspond a la paroi contre laquelle le chat rebondis
10
11            <div class="balls1"></div>
12            // correspond a l'image du chat. C'est a cette div que l'on appliquera le
13            code jquery
14        </div>
15
16        <script src="js/libs/jquery-1.9.0/jquery.min.js" type="text/javascript">
17            </script>
18        // Import de la librairie jquery
19
20        <script src="http://benahm.github.io/jquery.throwable/javascripts/
21            jquery.throwable.js" type="text/javascript"></script>
22        // Import de la librairie throwable
23
24        <script src="DefoulCat.js"></script>
25        // Import du code Javascript
26    </body>
```

La seule modification du code source réside à la ligne 8 : le chat est encadré par un cadre *parent* qui est défini par le CSS suivant :

```
1    .parent{
2        position: absolute;
3        left: 100px;
4        top: 150px;
5        width: 500px;
6        height: 500px;
7        border: 1px solid;
8        border-radius: 20px;
9    }
```

Ainsi, en appliquant ces trois modifications respectivement sur le code JavaScript, HTML et CSS, le chat ne peut pas quitter la zone *parent* :

FIGURE 2.7 – Defoul'Cat Version II



Pour aller plus loin

Il existe de nombreuses autres fonctionnalités facilement utilisables avec JQuery Throwable mais qui ne nécessitent pas d'être utilisées ici :

- Gérer une impulsion physique lors du premier lancement du jeu
- Gérer un amortissement qui peut ralentir ou accélérer les mouvements du chat
- Gérer la collision du chat avec un autre élément : le parent par exemple, ou tout simplement un autre chat.
- etc.

2.2.2 Résultat RainingCat

Dans le même esprit que *DefoulCat*, il a été demandé de réaliser un jeu où il est question de chats qui rebondissent. Cependant, il s'agit cette fois-ci d'un jeu non plus de détente, mais d'un jeu plus classique avec un but, un score et une fin possible. Techniquement, *RainingCat* possède le même code HTML que *DefoulCat* (cf. 2.2.1).

Demande du tuteur : Réaliser un jeu avec des chats. On doit pouvoir enregistrer un score. Les règles du jeu sont libre. Ce développement fait l'objet de trois fonctionnalités principales (cf. 1.3.1).

Augmentation de la difficulté

Perdre la partie

Enregistrer son score

Mise en oeuvre

Le but de ce jeu est d'empêcher les chats qui apparaissent à l'écran de toucher le sol. On utilise pour cela la souris, il faut lancer les chats en l'air et éviter qu'ils retombent, et ce le plus longtemps possible pour marquer un maximum de point.

En premier lieu, il faut naturellement appliquer la gravité au seul chat présent sur l'écran, tout comme *DefoulCat*. Ainsi il ne sera pas développé dans ce paragraphe les principes de gravité, de rebond, et d'interaction avec le joueur (cf. 2.2.1).

Un nouveau chat doit apparaître dans la zone de jeu

Le code suivant est une fonction JavaScript utilisant la bibliothèque JQuery (ce sera le cas pour tout les codes de ce jeu). Cette fonction s'occupe d'ajouter un chat en suivant ces étapes :

1. Apparition d'un nouveau chat dans le cadre.
2. Positionnement aléatoire du chat sur l'axe horizontal.
3. Application de la gravité au chat.

Il est à noter que ces opérations ne doivent pas être réalisés si le jeu est perdu.

```

1  function ajouterChat() {
2      if (!(aPerdu)) {
3          // verifie si le jeu n'est pas deja perdu
4          $(".parent").append("<div class='objectChat' id='Chat'></div>");
5          // ajoute dans la div parent, une div de chat
6          var leftPosChat = Math.floor(Math.random() * 450) + 1);
7          // definit un nombre aleatoire pour positionner aleatoirement le chat a l'
            horizontal
8          $("#Chat").css("left",leftPosChat);
9          // set la propriete left du css de la div chat nouvellement cree au nombre
            aleatoire
10         $("#Chat").throwable({
11             // ...
12         });
13     }
14 }
15 setInterval(ajouterChat, 5000);
16 // ajoute un chat toutes les 5s

```

La dernière ligne de ce code ne fait pas partie de la fonction d'ajout d'un chat, mais elle est absolument nécessaire au fonctionnement du jeu. Cette ligne permet d'appeler la fonction d'ajout d'un chat toutes les cinq secondes. Sans cette ligne, il n'y aurait jamais de nouveau chat à l'écran.

Gestion de la gravité pour chaque nouveau chat

Le code suivant fait partie de la fonction d'ajout d'un chat. Il a été passé sous silence, car certains de ses paramètres nécessitent une véritable explication.

Ce code se charge d'appliquer le plug-in* de gravité throwable au chat nouvellement créé. Ce sont à peu de choses près les mêmes lignes que celles expliquées dans la documentation de DefoulCat.

Cependant, contrairement à DefoulCat, le paramètre *areaDetection* est ici utilisé. Pour rappel, ce paramètre définit une zone (en px*) qui détecte les objets y entrant et en sortant. A chaque cas est associé un événement JavaScript auquel le code peut réagir. Ici, les valeurs définissent la zone du jeu. Il faudra donc récupérer l'événement "le chat sort de la zone" que l'on traduira par "le chat a touché le sol".

```

1      $('#Chat').throwable({
2      // Applique le plugin throwable au nouveau chat
3          containment: "parent",
4          drag:true,
5          gravity:{x:0,y:1},
6          impulse:{
7              f:70,
8              p:{x:0,y:1}
9          },
10         bounce:1,
11         damping:0,
12         autostart:true,
13         areaDetection:[[0,0,1000,636]],
14         collisionDetection: false
15     });

```

"Perdre" la partie

Comme dit plus haut, perdre la partie correspond à réagir à l'événement "le chat est sorti de la zone de jeu". Le déclencheur de cette réaction se trouve à la première ligne du code ci-dessous. Ensuite, si le jeu n'est pas déjà perdu (ce qui se traduit par : "si c'est le premier chat à toucher le sol") nous nous contentons d'afficher un message pour l'utilisateur et de qualifier la partie de perdue.

```

1      $(document).on("outarea",function (event,data) {
2          // Recupere l'evenement outarea lance par throwable pour determiner le game over
3          if (!(aPerdu)) {
4              // verifie si le jeu n'est pas deja perdu
5              var p = confirm("Perdu !");
6          }
7          aPerdu = true;
8          // qualifie le jeu de perdu dans une variable globale
9      });

```

Score

Le score fonctionne grâce au code suivant :

```

1      function score() {
2          // Cette fonction met a jour le score
3          if (!(aPerdu)) {
4              // verifie si le jeu n'est pas deja perdu
5              var score = parseInt($('#score').text())+1;
6              // recupere le score dans la variable "score" et l'augmente de 1
7              $('#score').text(score);
8              // inscrit le nouveau score dans l'element score du code html
9          }
10     }
11     setInterval(score,100); // appelle la fonction score() toutes les 100ms

```

La variable *aPerdu* qui est testée ligne 3 indique si le jeu est perdu ou pas, de façon à ce que le score ne varie plus une fois le jeu terminé.

Après rajout dans le code HTML d'un affichage du score, la ligne 4 du code JavaScript lit la valeur courante du score, transforme cette valeur textuelle en valeur numéraire, pour lui ajouter 1 et réécrit cette valeur dans le HTML (cf. la ligne 5 du code ci-dessous). Ainsi, la valeur affichée à l'écran de l'utilisateur est également la valeur sauvegardée pour les données de jeu.

```
1  <div id="scoreText">
2  // correspond au conteneur du mot score et a l'affichage de sa variable
3
4  <p>Score : </p><span id="score">0</span>
5  // correspond a la variable score et son affichage
6
7  </div>
```

Probleme du plugin de gravité pour ce jeu

L'implémentation de ce jeu a été possible grâce à l'utilisation du paramètre *areaDetection* du plugin* throwable. Pour ce faire, il nous a fallu renseigner les bonnes valeurs. Trouver ces valeurs fut un travail complexe, car le plugin* a une gestion étrange de la position des objets sur lesquels il applique la gravité.

Actuellement, le jeu présente un dysfonctionnement : le chat peut ne pas toucher le sol et déclencher la fin du jeu tout de même. Certes, ce cas de figure apparaît lorsque le chat est vraiment bas dans la zone, mais cela reste un problème. Dans les faits, ce comportement est souhaité, car il s'agit de la seule alternative à la condition d'arrêt du jeu "Le chat s'arrête au sol", condition d'arrêt qui rendait le jeu inintéressant en le simplifiant bien trop et qui est celle que l'on obtient si l'on ne comprend pas la logique de throwable.

La zone de jeu complète (l'espace où évoluent les chats) fait 750px* de large, et 750px* de haut. Ainsi, le paramètre 1000 de *areaDetection* couvre bien toute la largeur de la zone de jeu et il n'y a aucun risque qu'un chat sorte de cette zone par l'horizontal, et déclenche la fin du jeu par la même occasion.

Pour ce qui est du paramètre définissant la hauteur de la zone dont les chats ne doivent pas sortir, voici le calcul utilisé :

$$(80*80)+(80*80) = 12800, \text{ sqrt}(12800) = 113,137, 750 - 114 = 636$$

Il s'agit d'une application du théorème de Pythagore, utilisé afin de soustraire la diagonale de l'image du chat (qui se trouve être un carré de 80*80px*) à la hauteur de la zone de jeu complète.

Ainsi, le chat touche le sol quand un de ses coins entre en contact avec. Cela pose problème, car en prenant une telle valeur comme paramètre, si le chat n'est pas en diagonale quand il touche le sol, il le touchera plus "haut" dans l'écran. Seulement, si nous nous contentons de renseigner une taille de 80px* (soit la hauteur de l'image du chat), il faut attendre que le chat repose au sol pour perdre. Ce comportement est inacceptable, car le jeu devient alors si facile qu'il perd tout intérêt. De ce fait, nous avons décidé de garder la valeur théorique de 636px*. Ce problème est connu par l'équipe, mais il n'a pas été possible de le résoudre.

2.2.3 Résultat CatDefend

Après la création des deux précédents jeux en JavaScript, le développement d'un troisième jeu avec les mêmes principes de gravité et de physique semblait beaucoup plus simple.

Demande du tuteur : Réaliser un jeu avec des chats. Les règles du jeu sont libres. Ce développement fait l'objet de quatre fonctionnalités principales (cf. 1.3.1).

Perdre la partie

Enregistrer son score

Gestion des ressources

Gestion des attaques & bonus

Concernant ces deux derniers points, divers problèmes sont apparus lors d'une première phase d'analyse, notamment pour gérer la collision et la destruction des éléments générés aléatoirement (les ennemis qualifiés ici de "chiens"). D'autant plus que gérer la collision avec le plugin JQuery Throwable était sujet à plusieurs bugs majeurs.

Une solution a été trouvée, mais elle était très complexe : maintenir en temps réel un tableau à deux entrées qui mémorise les coordonnées du point central de chaque élément. Le but étant, avec l'aide de calculs complexes, de détecter la collision des objets. Mais le principal problème était le suivant : si un nombre trop important d'éléments étaient générés, le tableau prendrait des dimensions énormes et le jeu risquait de connaître des problèmes de performance trop importants et impacter les sensations du joueur.

Il fut alors décidé d'utiliser une autre solution pour créer le jeu : le logiciel Construct 2.

Mise en Œuvre

La gravité et la direction du chat

Avant d'aborder le jeu, il est nécessaire de comprendre des points clés de Construct 2. Le logiciel se compose de « layers* » qui représentent différentes couches sur lesquelles on peut faire apparaître des objets nommés « Sprites* ».

Dans Cat's Defend, il existe différents Sprites*. Le premier est celui représentant le chat que nous contrôlons, puis il y a ceux des lasers tirés par le chat, ainsi que les chiens qui attaquent et même les limites des bords de l'écran jouable. Ils sont donc au coeur du jeu.

Un sprite* est donc un élément visuel pouvant être de l'ordre du décor ou d'objet tangible dans le jeu. On lui attribut un "Behavior*" (Comportement) qui permet par la suite de paramétrer ses actions à partir de la page d'évènement.

Le chat possède donc un comportement de « Platformer ». Ce comportement signifie que notre chat est soumis à une certaine gravité dont les paramètres sont spécifiés dans un onglet du logiciel.

FIGURE 2.8 – Paramètres du Chat CatDefend

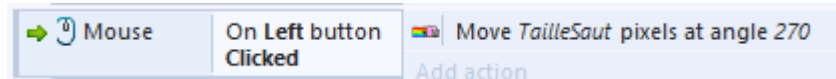
Behaviors	
Platform	
Max speed	500
Accelerati...	1500
Decelerati...	1500
Jump stre...	650
Gravity	100
Max fall s...	150
Default co...	Yes
Initial state	Enabled
Add / edit	Behaviors

Afin de rendre le chat contrôlable, la vitesse maximale de chute (Max fall Speed ci-dessus) du chat a été réduite, afin que le chat mette plus de temps à descendre pour avoir le temps de le contrôler. De base elle était à 200.

Pour la direction, deux solutions furent utilisées. La première fut la configuration des touches 'Q' et 'D' du clavier du joueur afin qu'il puisse faire avancer ou reculer le chat sur l'écran. Par rapport à la figure 2.8, la vitesse maximale est de 500. Donc plus le joueur appuie longtemps sur la touche, plus sa vitesse augmente jusqu'à un maximum de 500.

La seconde solution fut au niveau du saut possible pour le chat. En effet, le comportement «Platformer» classique n'autorise le saut que si le chat est sur une plate-forme. Or, dans le jeu, le chat est dans les airs. Il a donc été implémenté l'évènement suivant :

FIGURE 2.9 – Évènement : "Saut" CatDefend



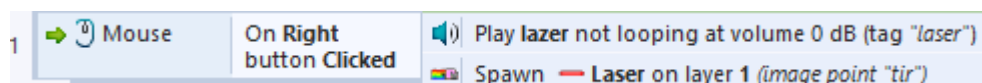
Cet évènement permet, lors d'un clic gauche, de faire monter le chat de *TailleSaut* px* de haut (variable initialisée à 120). Donc sa augmente de 120px. La fonction utilise un angle pour connaître la direction du chat. L'angle est de 270 degrés afin que l'image du chat puisse remonter.

Chat, laser et ennemis

En plus des Sprites* tangibles, il en existe aussi des non-tangible. En effet, il est également possible d'ajouter des "objets" que l'on ne peut pas voir à l'écran (comme le clavier par exemple). Leur utilité est très avérée : Construct 2 possède une page nommée *Event Sheet** contenant en son sein des événements et des réactions associées choisies par le développeur.

On souhaite que le chat puisse tirer des lasers avec le bouton droit de la souris. Ces fameux lasers servent à détruire les ennemis. On crée donc un évènement du type : "Si le joueur clique droit avec sa souris, un laser apparaît." (cf. 2.10).

FIGURE 2.10 – Évènement : "Tir du laser" CatDefend



Les lasers possèdent un comportement « Bullet » afin de pouvoir se diriger dans une certaine direction lors de leur apparition. Il fut décidé de leur attribuer une vitesse élevée afin que les lasers puissent parcourir plus vite la distance les séparant des chiens attaquant. Pour les chiens, un sprite* spécial fut créé dans le but de pouvoir les faire apparaître à divers points paramétrés sur lui-même.

On attribua aux chiens le comportement « CustomMovement » (Mouvements paramétrables) afin de pouvoir les faire aller de leur point d'apparition vers le chat à l'écran. L'évènement permettant le déplacement des chiens fut créé de la sorte :

FIGURE 2.11 – Évènement : "Déplacement du chien" CatDefend

20	System	Every tick	Set Bullet speed to 1500
			Set CustomMovement Enabled
			Set CustomMovement Horizontal speed to Speed
			Set CustomMovement angle of motion to 180

Ligne 1, on spécifie que Every tick (en permanence) la vitesse des lasers est de 1500. Ligne 2, on active le déplacement des chiens. Ensuite ligne 3 on indique que la vitesse du mouvement horizontal vaut *Speed* (initialisé à 100). Enfin ligne 4, on définit l'angle qui permet au chien d'aller vers le chat (donc ici 180 correspond à l'horizontal).

Au début, les chiens apparaissaient tous en même temps, créant des lignes de chiens continues et posant des soucis de jeu. En effet, si une rangée de chiens passait derrière le chat, celui-ci ne pouvait plus la détruire sans mourir. La solution fut la suivante :

FIGURE 2.12 – Évènement : "Apparition des chiens" CatDefend

15	System	Every 3 seconds	Spawn Méchant on layer 1 (image point "Spawn1")
			Add action
16	System	Every 5 seconds	Spawn Méchant on layer 1 (image point 0)
			Add action
17	System	Every 6 seconds	Spawn Méchant on layer 1 (image point "Spawn2")
			Add action
18	System	Every 2 seconds	Spawn Méchant on layer 1 (image point "Spawn3")
			Add action
19	System	Every 4 seconds	Spawn Méchant on layer 1 (image point "Spawn4")
			Add action

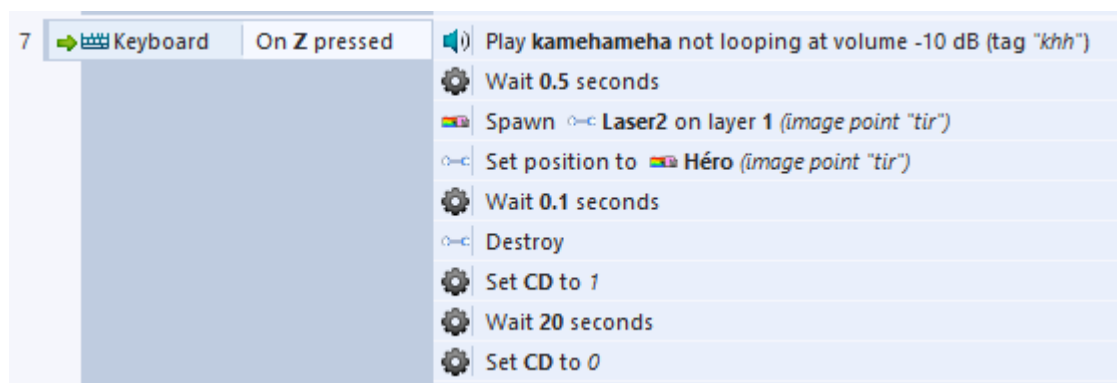
Les apparitions de chiens furent paramétrés afin qu'ils apparaissent à différents endroits à des moments précis, créant une impression d'aléatoire et évitant de rester bloquer si jamais un chien venait à passer derrière le chat.

Compétences

Le chat n'est pas un simple chat. En effet, il peut, en plus des lasers, utiliser deux compétences qui possèdent un temps de recharge de 20 secondes chacune.

La première compétence se nomme « Miaoumeameha ». Elle permet de tirer un laser pouvant détruire plusieurs cibles (les chiens). Mais ce laser possède une caractéristique très importante : un temps de recharge. Celui-ci est géré de la manière suivante :

FIGURE 2.13 – Évènement : "Miaoumeha" CatDefend

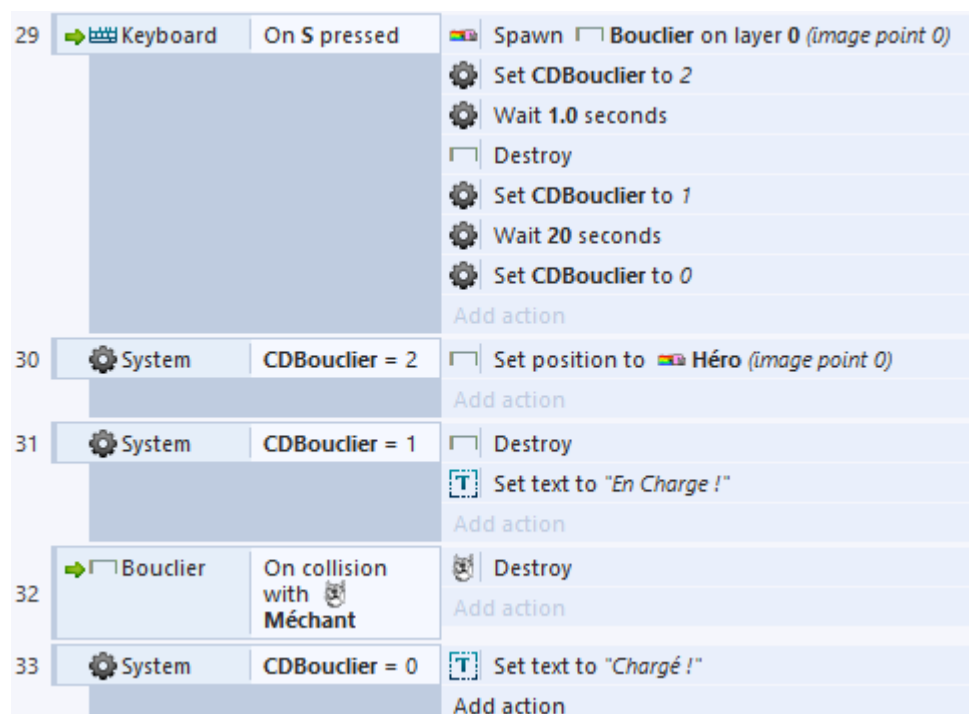


On a donc déclaré une variable globale* qui nous sert de compteur (CD). Elle est de base à 0 (ligne 7) et varie à 1 lorsque la compétence est activée. Après l'apparition du Sprite* *Laser*, un délai de 20s (ligne 8) est lancé avant de remettre la variable à 0 (ligne 9). Si le joueur appui à nouveau sur la touche alors que le temps de recharge n'est pas encore terminé, il ne se passe rien.

En haut de la page de jeu, la compétence est marquée « En charge ! » ou « Chargée ! » en fonction de l'état de chargement de la compétence.

Un second sort a été mis en place : le bouclier. Le bouclier a pour effet de créer une zone autour de notre personnage (figure 2.14, ligne 1 & ligne 8) afin de le protéger durant une courte durée des ennemis et de détruire ceux-ci au contact (ligne 11). On rappelle que si le chat rentre en contact avec le chien, la partie est perdue.

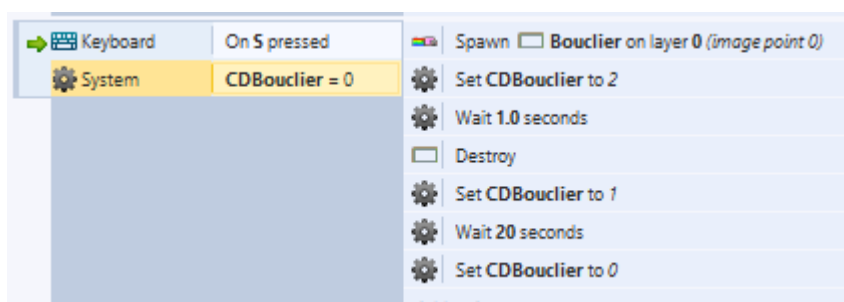
FIGURE 2.14 – Évènement : "Bouclier" CatDefend



Mais un problème persistait. En effet, le bouclier lorsqu'il apparaît modifie directement la variable qui affecte son temps de recharge, permettant ainsi au joueur d'utiliser le bouclier sans tenir compte du temps de recharge du bouclier.

Une légère modification a été implémentée afin de résoudre le problème. En effet, une simple condition supplémentaire a été ajoutée (il faut que $CDBouclier = 0$) afin de permettre la relance du sort uniquement au bout de 20s.

FIGURE 2.15 – Évènement : "Bouclier" (Problème résolu) CatDefend



Le Score et la sortie de l'écran

Il a été décidé pour la partie score de compter le nombre de cible touchées par un laser en incrémentant le score de 1 et de rendre très punitif le passage d'un chien de l'autre côté de l'écran en réduisant de 5 le score pour chaque chien qui a traversé. Le but était donc d'inviter le joueur à tirer sur les chiens (cf. figure 2.16).

FIGURE 2.16 – Évènement : "Laser et Score" CatDefend



Construct 2 permet de gérer les scores d'une façon quelque peu atypique. En effet, on ne peut pas stocker les scores dans une variable récupérable pour l'afficher sur le site (comme Raining Cat). Le logiciel ne permet que le stockage de ce score sur un site externe nommé <http://www.scirra.com/arcade>. On peut donc stocker ou afficher le score, mais pas dans la base de données de l'application.

Afin de compter le nombre de chiens étant passés derrière le chat, un sprite* fut créé en dehors de l'écran et sert à détecter la collision des chiens avec celui-ci. En effet, au contact, le chien est détruit et le score diminue. Cela permet aussi d'éviter de gérer un grand nombre de sprites* passés en dehors de l'écran en continuant leur trajectoire, et donc d'éviter des problèmes de ralentissement sur le jeu (cf. figure 2.17).

FIGURE 2.17 – Évènement : "Baisse du score" CatDefend



La sortie d'écran est aussi gérée pour le chat. En effet, le chat ne peut pas sortir de l'écran car sinon il perd la partie. Au début, il y avait quelques soucis avec la sortie de l'écran car les sprites* la gérant étaient placés beaucoup trop près des bords et il arrivait souvent de mourir en essayant de toucher les cibles les plus en haut ou les plus en bas (cf. figure 2.18)

FIGURE 2.18 – Évènement : "Mort du Héros" CatDefend



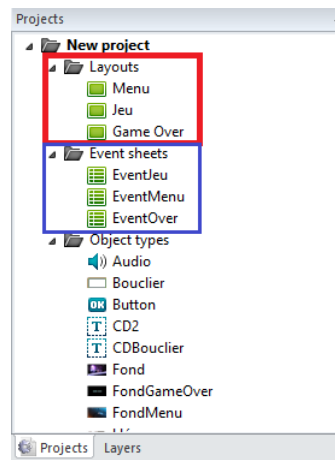
Layout, menu et game over

Dans construct 2, un « layout* » est l'équivalent d'un niveau de jeu. A chaque « Layout » est attribué son « Event Sheet* », c'est à dire une page d'évènement qui lui est propre. Les Layouts* peuvent être communiquant entre eux grâce à des fonctions du logiciel. C'est donc grâce à cela que nous avons pu créer un menu et une page de "game over".

Au début, nous n'avions pas utilisé le principe de layout* mais celui des layers*, qui sont des calques dans lesquels nous pouvons mettre différentes choses (par exemple, un layer* sert à afficher les informations, l'autre à afficher le Héros et les ennemis,...). Le principe étant le suivant : quand je clique sur jouer, le layer* menu devient invisible et s'affiche celui du jeu. Quand le Héros meurt, le layer* de jeu devient invisible et s'affiche celui de game over. Or cela posait beaucoup de soucis car, bien que cachés, les objets se trouvaient quelque part sur l'écran de jeu, et il arrivait souvent que le jeu redémarre car on avait cliqué sur le bouton invisible qui servait à jouer. Il arrivait aussi que notre score soit modifié lorsque qu'une partie était perdue car même si on masquait le layer* du jeu, il continuait de fonctionner en arrière plan et donc à décrémenter le score.

Nous avons donc mis en place un système de layout* avec chacun un nom correspondant à sa fonction : LayoutMenu, LayoutJeu, LayoutGameOver de façon à pouvoir jongler entre eux. Le premier s'affichant étant celui du menu, permettant l'accès au jeu. Lorsqu'une partie est perdue, on arrive sur le layout* de game over qui nous affiche notre score et nous propose de recommencer une nouvelle partie.

FIGURE 2.19 – Exemple de Layout CatDefend



Animation, sons et musique

FIGURE 2.20 – Évènement : "Son sur le Miaoumeheha" CatDefend

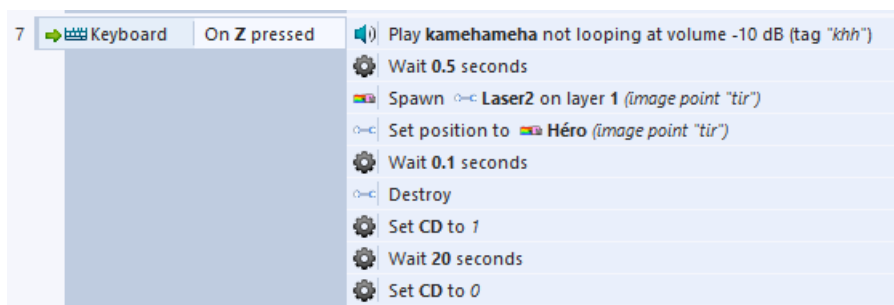


FIGURE 2.21 – Évènement : "Son sur le laser" CatDefend



Afin de rendre le jeu graphiquement plus joli et plus amusant, des animations ont été réalisées et des sons implémentés en fonction des actions réalisées. En effet, lorsqu'un tir de laser est lancé, un bruit de laser se déclenche. Le bruit est préchargé avant le lancement de chaque partie afin qu'il puisse être utilisé. De même, lorsque le miaoumeheha est lancé, un bruitage est actionné, et à la fin de celui-ci, la compétence est lancée

FIGURE 2.22 – Évènement : "Animation du Héros" CatDefend



Pour les animations, notre chat, lors de son déplacement avant ou arrière, déclenche une animation de déplacement qui lui fait bouger les pattes, la tête et l'arc-en-ciel. Cette animation rend le personnage moins statique. L'animation a été réalisée image par image de façon à la rendre la plus fluide possible.

Il a été essayé de faire une autre animation, lors du lancement d'un laser. Mais cela n'a pas bien marché. En effet, lors du déplacement, l'animation de tir de laser faisait bugger le jeu et accusait souvent des rollback (retour en arrière) du personnage et des bugs de l'animation et de la zone de tir. Bref, l'idée fut abandonnée, de même que celle sur le Miahoumeheha, posant les mêmes soucis.

Au niveau de la musique, une bande son n'a pas été implémenté à cause de la limitation de la version gratuite, toutefois, un test sonore est disponible lorsque l'on appuie sur la touche « maj » qui active alors une musique.

2.2.4 Résultat Développement du site

Introduction

Demande du tuteur : Réaliser un site sur lequel on pourra jouer à des mini-jeux. On doit pouvoir s'identifier et se connecter pour enregistrer son score. Ce développement fait l'objet de trois fonctionnalités principales (cf. 1.3.1 & 1.4).

Base de données

Utilisation du site

Enregistrer son score

Mise en oeuvre

Gestion de la base de données

Nous avons fait le choix d'une connexion à la base données via PDO car c'est la seule méthode que nous ayons vu et que nous n'avions pas le temps d'en voir d'autre, de plus cela permet la préparation de requêtes.

L'ensemble des fichiers contenus dans Model (cf. Figure 2.2) interagissent avec la base de données. Par exemple lors de la création d'un utilisateur, le controller reçoit les paramètres et les passe au modèle pour qu'il les enregistre dans la base de donnée comme suit :

```
1      public static function insert($data) { //créer un utilisateur
2      // $data est un tableau indexé par les champs de la table utilisateur
3      try {
4          // Preparation de la requete
5          $req = self::$pdo->prepare('INSERT INTO utilisateur (login, nom, prenom, email,mdp
              ) VALUES (:login, :nom, :prenom, :email, :mdp)');
6          // execution de la requete
7          return $req->execute($data);
8      } catch (PDOException $e) {
9          echo $e->getMessage();
10         die('Erreur lors de l\'insertion d\'un utilisateur dans la BDD');
11     }
12 }
```

Le site

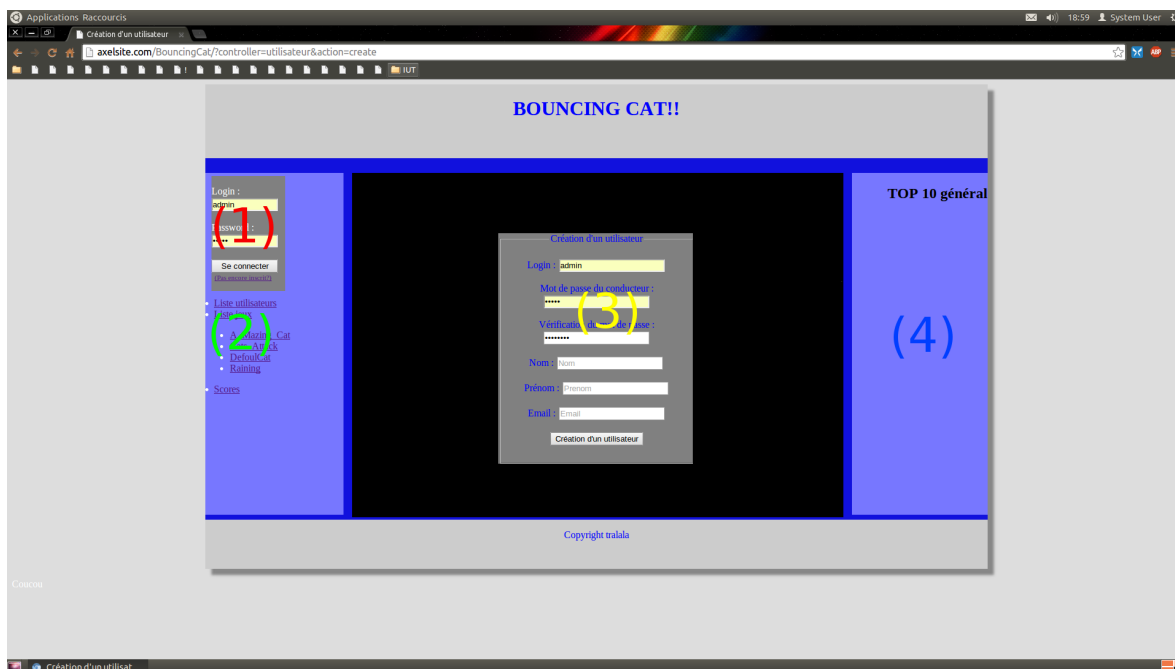
Le site fonctionne sur la base d'une "vue principale" qui est chargée et dans laquelle sont appelées les différentes "sous"-vues dont on a besoin :

```

1  <?php require VIEW_PATH.'utilisateur'.DS.'viewConnectUtilisateur.php'; //chargement de
    la vue de connexion (1)
2  ?>
3  <ol> <?php if (Session::is_user('admin')) echo "<li><a href='./?&action=all&
    controller=utilisateur'>Liste utilisateurs</a></li>";?>
4      <li><a href="./?&action=all&controller=game">Liste jeux</a>
5          <ul><?php jeux($jeux) //affichage de la liste des jeux sous forme de
            liens (2)
6          ?></ul>
7      </li>
8  </ol>
9  <div id="jeu">
10 <?php require VIEW_PATH.$controller.DS.'view'.ucfirst($view).ucfirst($controller).''.php
    '; //chargement de la vue demandee (3)
11 ?>
12 </div>
13 <aside id="classement">
14 <?php
15     $action="classement";
16     require ROOT . DS . 'controller' . DS . 'ControllerGame.php'; //chargement
        de la vue classement (4)
17     ?>
18 </aside>

```

FIGURE 2.23 – Plan du site



Par exemple pour la création d'un utilisateur :

- (1) La vue de connexion
- (2) Le menu avec la liste des jeux
- (3) La vue d'inscription
- (4) La vue classement (vide au moment de l'impression d'écran)

Nous avons choisi d'ouvrir les jeux dans une nouvelle fenêtre au lieu de les charger comme "sous-vue" car les positionnements, relatifs, absolus et fixes étaient trop difficiles à manipuler avec le plugin JGravity : le cadre dans lequel le chat devait rebondir était impossible à gérer. Plus tard, après un passage au plugin JQuery Throwable, l'idée de joueur sur une nouvelle page a été conservée car elle offre plus de possibilités au joueur.

Le score

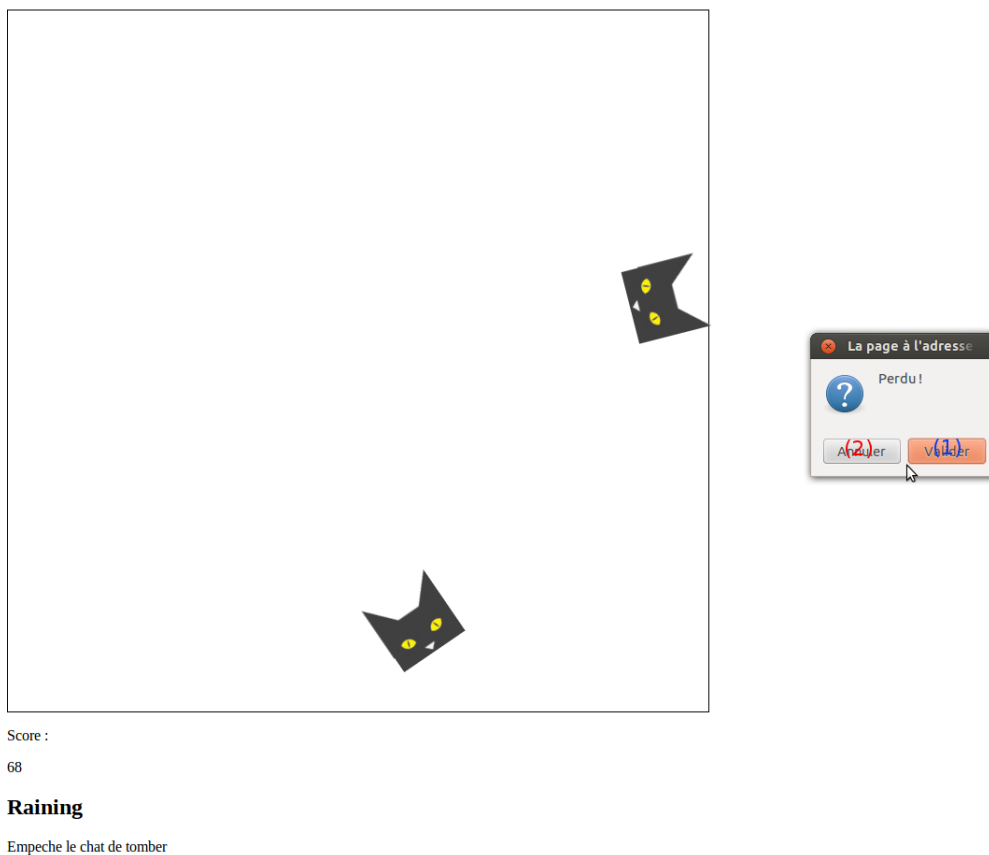
La fonctionnalité la plus difficile à implémenter a été la récupération de la variable *score* du jeu **Raining Cat** pour l'enregistrement dans la base de données. Finalement c'est le code JavaScript du jeu qu'il a fallu modifier pour obtenir le résultat attendu. Ce dernier définissait un événement "Si l'élément est en dehors de la zone prévu", une alerte est déclenchée pour arrêter le score. Il a été modifié comme suit :

```

1      $(document).on("outarea",function (event,data) {
2      if (!(aPerdu)) { // verifie si le jeu n'est pas deja perdu
3          var p = confirm("Perdu !"); //ouvre une boite de dialogue indiquant au joueur qu'
              il a perdu;
4
5          if (p){ //si il fait ok (1)
6              var k = ""+parseInt($('#score').text());
7              document.location.href="http://axelsite.com/BouncingCat/?action=created&
              controller=score&id=Raining&score="+k;
8          }
9          //sinon il reste sur la page et son score ne sera jamais enregistre (2)
10     }

```

FIGURE 2.24 – score de Raining



Toujours avec le même évènement, en plus de définir une alerte, on enregistre le score.

On définit une variable `p` (ligne 3) qui prend la valeur "vrai" si le joueur a cliqué sur "ok" (1) lors de l'alerte, faux sinon. Puis ligne 5 à 8, si `p` est vrai, on récupère le score dans une variable `k` et on re-dirige le joueur sur le site principal avec la fonction `document.location.href`.

A la rédaction de ce rapport, le score concernant CatDefend n'a pas été implémenté dans la Base de Données.

2.3 Tests

Afin d'améliorer l'application, plusieurs séries de tests ont été réalisées à intervalle régulier. Pour chaque nouvelle série, les tests ont été classés dans deux catégories distinctes : les tests dit *concluants*, c'est à dire qui ne nécessitent pas de modification particulière ; et les tests *ayant permis une amélioration du système*.

D'autres tests pourront être effectués entre le rendu de ce rapport et la présentation orale du projet.

2.3.1 Tests concluants

- Les champs du formulaire d'inscription sont tous nécessaires.
- On ne peut pas se connecter si l'identifiant ou le mot de passe est erroné.
- Dans chaque jeu, le chat ne doit pas quitter la zone d'action prévue.

2.3.2 Tests ayant permis une amélioration

- Quand l'administrateur (et seulement lui) se connecte, il obtient la liste des utilisateurs de la base de données. S'il se déconnecte en étant sur cette liste, il y a un retour à la page d'accueil.
- Le score de tous les utilisateurs doit s'enregistrer. Si l'utilisateur n'est pas connecté, on lui attribue le pseudo "Non connecté".
- N'importe quel utilisateur peut visualiser les scores de chaque jeu.
- Pour chaque taille d'écran ou de navigateur différente, le site s'adapte en fonction de cette nouvelle taille. Le jeu par contre ne change pas de dimension pour éviter les bugs*.
- Quand l'administrateur crée un compte, celui-ci doit rester connecté avec son compte.
- Le formulaire de modification d'un compte se préremplit avec les informations du compte, sauf pour le mot de passe.
- Le champ mail du formulaire de création d'un compte doit être valide.
- Quand l'administrateur clique sur "supprimer un utilisateur", on lui demande confirmation avant de réaliser la suppression.

2.4 Perspectives

Concernant l'ensemble du projet, de nombreux points sont améliorables.

Tout d'abord, et ce, malgré une volonté de bien faire, l'interface graphique est nettement améliorable tant du point de vue du design que du point de vue structurel. Étant étudiants en IUT Informatique, nous souhaitons favoriser le côté technique de ce projet plutôt que l'interface. Nous sommes néanmoins pleinement conscients que l'interface représente un facteur clé dans l'appréciation de l'utilisateur de notre application. De ce dernier critère dépend son succès sur la toile.

Ensuite, après réflexions avec notre tuteur, il peut apparaître avantageux d'insérer quelques publicités à certains endroits du site. Cet aspect "rentable" est loin d'être notre motivation première. Néanmoins, cette idée démontre à quel point ce système peut s'ancrer dans un contexte réaliste. On peut tout à fait imaginer notre application comme faisant partie intégrante d'une plate-forme de ce genre déjà existante.

Troisièmement, notre formation n'étant pas terminée, notre code peut être améliorable. Certaines fonctions sont sans doute mal utilisées et mal optimisées.

Enfin, ce projet peut être repris librement, soit par l'ensemble de l'équipe, soit par l'un de ces membres pour y implémenter de nouveaux jeux ou de nouvelles fonctionnalités. C'est sans aucun doute ce type de contenu qui fait la richesse de l'application. Par manque de temps, nous n'avons développé que trois jeux, mais l'équipe possède encore un grand nombre d'idées plus réalisables les unes que les autres pour améliorer Defoul'Cat, RainingCat ou CatDefend. Notre projet sera plus riche et plus complet à mesure que le temps passera.

Chapitre 3

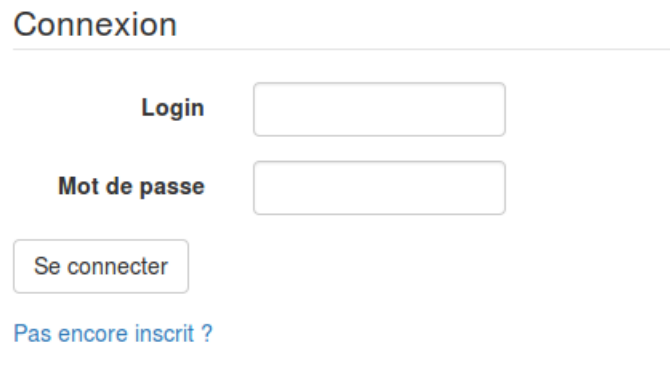
Manuel d'Utilisation

Nous essayerons dans ce court manuel d'expliciter la plupart des fonctionnalités mises à disposition de l'administrateur (pour accéder au site : cf. Références). Ce manuel a pour but de faciliter la prise de main du système même si l'administrateur change au fil du temps.

Pour effectuer les étapes suivantes, l'administrateur doit impérativement se connecter : Dans l'espace prévue à cet effet (cf. 3.1) mettre dans la case login et mot de passe "admin". Puis cliquez sur "Se Connecter".

Une fois connecté, vous pouvez librement accéder aux fonctionnalités décrites ci-dessous.

FIGURE 3.1 – Formulaire de connexion



Connexion

Login

Mot de passe

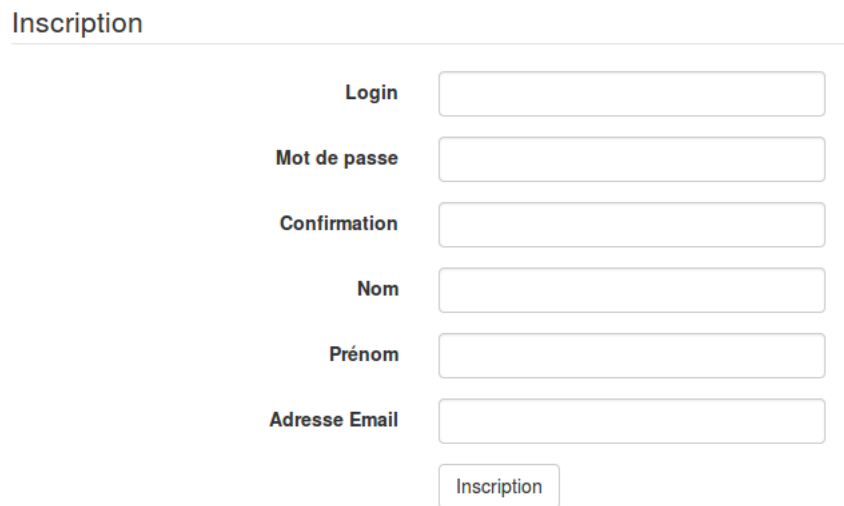
[Pas encore inscrit ?](#)

3.1 Gestion des utilisateurs

- **Ajout d'utilisateur :**

1. Une fois connecté, au centre du site est apparue la liste des utilisateurs. Cliquez sur "Créer un nouvel utilisateur" à la suite de cette liste.
2. Complétez le formulaire ainsi apparu : attention tous les champs sont obligatoires, y compris le mot de passe.
3. Cliquez sur "Créer l'utilisateur".

FIGURE 3.2 – Formulaire d'ajout d'un utilisateur



Le formulaire d'inscription est intitulé "Inscription" et est structuré comme suit :

- Champs de saisie : Login, Mot de passe, Confirmation, Nom, Prénom, Adresse Email.
- Bouton : Inscription.

- **Supprimer un utilisateur :**

1. Une fois connecté, au centre du site est apparue la liste des utilisateurs. Cliquez sur le lien "Détail" de l'utilisateur à supprimer.
2. Cliquez sur "Supprimer" en dessous de la description de l'utilisateur.

FIGURE 3.3 – Interface de modification / suppression d'un utilisateur

Sous le login Noren se trouve l'utilisateur Axel Rezé dont l'email est axel.reze@live.fr. Il a déjà joué à :

- Raining Voir

Mettre à jour, Supprimer

Retour à la page principale.

- **Modifier un utilisateur :**

1. Une fois connecté, au centre du site est apparue la liste des utilisateurs. Cliquez sur le lien "Détail" de l'utilisateur à mettre à jour.
2. Cliquez sur "Mettre à jour" en dessous de la description de l'utilisateur.
3. Modifiez le formulaire ainsi apparu : les anciennes informations sont encore présentes. Attention le login de l'utilisateur n'est pas modifiable.

4. Cliquer sur "Mise à jour".

FIGURE 3.4 – Formulaire de mise à jour d'un utilisateur

Mise à jour

Login	<input type="text" value="Noren"/>
Mot de passe	<input type="password"/>
Confirmation	<input type="password"/>
Nom	<input type="text" value="Rezé"/>
Prénom	<input type="text" value="Axel"/>
Adresse Email	<input type="text" value="axel.reze@live.fr"/>
	<input type="button" value="Mise à jour"/>

N

3.2 Jouer

- **Jouer à l'un des jeux proposés**
 1. Cliquez sur le jeu souhaité (par exemple "Defoul'Cat") sous le formulaire de connexion.
 2. Cliquez sur "Jouer". Une nouvelle page s'est ouverte pour vous permettre de jouer.

FIGURE 3.5 – Fiche du jeu DefoulCat avec lien de jeu

DefoulCat

Aide le chaton à s'amuser et à sauter partout ! Fait rebondir le chat dans le ciel avec ta souris !

[Jouer](#)

Chapitre 4

Rapport d'Activité

4.1 Présentation des outils utilisés

Dans la rubrique Références disponible avec les Annexes se trouvent les liens disponibles pour accéder aux différents outils cités ci-dessous.

4.1.1 Trello

Trello est un outil de gestion de projet en ligne. Il définit un espace auquel tous les membres du projet ont accès. Dans cet espace se trouvent plusieurs listes de tâches définies par les utilisateurs. Il est possible de renseigner ces tâches avec une grande précision, d'y affecter des membres, de leur donner des thématiques, d'en discuter. Il est également possible de bouger ces tâches de liste en liste.

Nous avons décidé d'utiliser Trello car toute l'équipe connaissait déjà ce site, et qu'il est très simple d'utilisation pour obtenir un tel résultat de gestion de projet. Nous avons organisé notre espace de travail sur ce site en quatre listes :

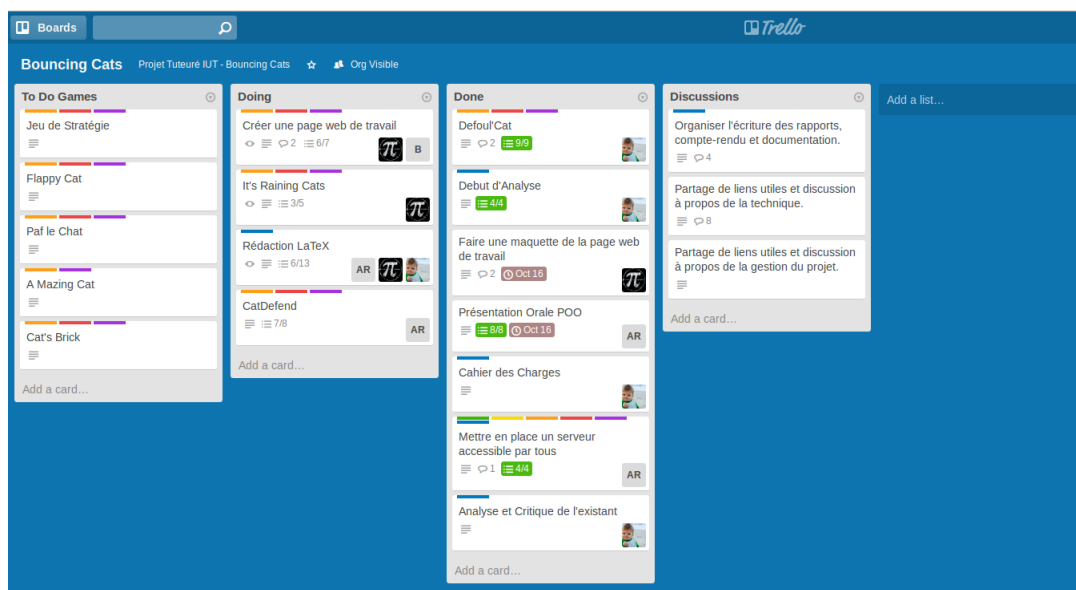
To Do : Dans cette liste se trouvaient toutes les tâches à faire pour le prochain rendez-vous avec le tuteur. C'est là que les membres de l'équipe allaient chercher du travail s'ils étaient inoccupés.

Doing : Cette liste montrait à tout les membres du projet qui était en train de travailler sur quoi.

Done : Cette liste répertoriait toutes les tâches terminées, c'est à dire fonctionnelle et testée.

Discuss : Cette liste de "fausses" tâches était rangée par thématiques. Nous utilisions ces tâches pour garder une trace de nos discussions et pour discuter en différé.

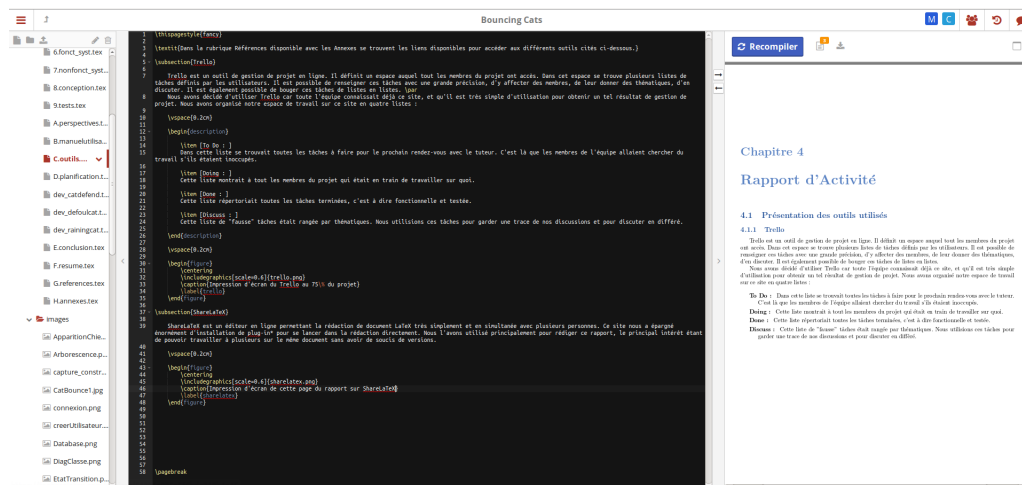
FIGURE 4.1 – Impression d'écran du Trello au 75% du projet



4.1.2 ShareLaTeX

ShareLaTeX est un éditeur en ligne permettant la rédaction de document LaTeX très simplement et en simultanée avec plusieurs personnes. Ce site nous a épargné énormément d'installations de plug-in* pour se lancer dans la rédaction directement. Nous l'avons utilisé principalement pour rédiger ce rapport, le principal intérêt étant de pouvoir travailler à plusieurs sur le même document sans avoir de soucis de versions.

FIGURE 4.2 – Impression d'écran de cette page du rapport sur ShareLaTeX



4.1.3 FileZilla

FileZilla est un client FTP. IL nous a permis de mettre en ligne nos fichiers sur le serveur web afin de visionner notre travail. Il s'agit d'un logiciel open source sous licence GNU qui est plus que largement utilisé.

4.1.4 Bootstrap

Bootstrap est un framework CSS. Il fournit des feuilles de styles prédéfinis et modifiables à l'aide de templates téléchargeables sur différents sites ressources. Nous l'avons utilisé pour faire évoluer le design du site vers quelque chose de plus neutre, le tout sans se compliquer la tâche à écrire des lignes et des lignes. De plus certains membres du groupe devaient pour leurs stages apprendre comment se servir de Bootstrap, ce qui fut déterminant dans notre choix.

4.2 Cycle de développement

Les cycles de développement du projet étaient rythmés par les rendez-vous avec notre tuteur pour discuter du projet. Nous nous imposions d'avoir des choses nouvelles à lui montrer à chaque rendez-vous et ce fonctionnement nous a permis une révision itérative de certaines tâches en fonction de l'évolution des besoins et de nos compétences techniques.

De plus, le planning prévisionnel était d'une grande importance pour les membres de l'équipe, car certains membres ne pouvaient pas se permettre de travailler plus sans couper court à d'autres projets personnels ou à leur travail. Suivre ce planning du mieux possible a amené la création de gros cycle de développement durant un mois venant encadrer les rendez-vous presque hebdomadaires avec le tuteur.

Enfin, le groupe de travail sur ce projet étant un groupe d'affinités, de nombreuses réunions de travail se tenaient de façon informelle aux interours, aux pauses déjeuner et aux autres moments libres de la journée.

4.3 Planification

FIGURE 4.3 – Planning prévisionnel

	Octobre	Novembre	Décembre	Janvier
Axel R.	Analyse	Création de ressources graphiques et création de cat's defend	Fin de création de cat's defend et rédaction du rapport	Rédaction du rapport et création du Prezi de présentation
Clément B.	Apprentissage Javascript/JQuery	Apprentissage de throwable et création de raining cat	Finalisation raining cat et création du bootstrap pour le site	Rédaction du rapport et aide à la création du Prezi de présentation
Emmanuel B.	Documentation et analyse	Création du HTML/CSS et début d'implémentation PHP/MySQL	Implémentation PHP et MySQL	Finalisation du PHP/MySQL et rédaction du rapport
Sacha L.	Apprentissage Javascript/JQuery	Apprentissage de throwable et création de defoul cat	Fin de création de defoul cat et rédaction du rapport	Rédaction du rapport

Ci-dessus le planning prévisionnel tel qu'il a été conçu la première semaine du projet. Évidemment, le projet ne s'est pas déroulé exactement comme prévu, aussi nous allons voir ici les différences entre ce document et ce qu'il s'est réellement passé.

Le premier écart avec les prévisions est arrivé relativement tard. Sacha L. avait fini son jeu plus vite que prévu. En raison du caractère inachevé du travail des autres membres de l'équipe, elle ne pouvait pas se lancer dans la rédaction du rapport. Elle aida donc les autres membres en leur expliquant certains fonctionnements du plug-in* de gravité throwable et en cherchant des outils adaptés à la future écriture de ce rapport.

Un deuxième écart avec les prévisions est apparu courant décembre. Clément B. avait pris du retard sur son jeu, ce qui le força à implémenter Bootstrap sur le site très rapidement durant les vacances de décembre. Ceci peut expliquer le design très sommaire du site.

Le dernier écart majeur de ce planning est dû à l'enregistrement des scores. Emmanuel B. devait s'occuper de récupérer les scores des différents jeux pour les inscrire dans la base de données. Il pouvait compter sur l'aide des développeurs des différents jeux pour mener à bien cette tâche, mais récupérer le score d'un jeu en JavaScript avec du PHP lui a posé plus de problèmes que prévu. Il a donc pris plus de temps pour développer cette fonction essentielle du site, au détriment du rapport qu'il aura moins travaillé.

Conclusion

Nous avons choisi ce projet pour plusieurs raisons.

D'abord, parce qu'à l'inverse d'autres sujets, celui-ci proposait d'explorer un langage informatique très peu connu par l'ensemble de l'équipe. La liste des compétences que nous avons apprises en autodidacte fait que nous possédons un plus par rapport à nos camarades. Ensuite parce que ce projet a permis d'aboutir à un résultat concret, fonctionnel, que nous pouvons conserver durablement dans le temps et utiliser.

Chacun d'entre nous est fier d'avoir pu participer à ce projet. Le fruit de notre collaboration ne fut pas qu'un enrichissement technique, il fut également un enrichissement humain.

Références

- **Bouncing Cat** [consultation pendant tout le projet] :
<http://www.axelsite.com/BouncingCat/>

Site personnel : Le site Bouncing Cat est accessible sur le site internet d'un des collaborateurs du projet. Il regroupe l'ensemble des jeux ainsi que l'interface de navigation.

- **Cat Bounce** [consultation 10/2014] :
<http://cat-bounce.com/>

Site personnel : Ce site réalisé en Flash permet de jouer à Cat Bounce, un jeu où des chats rebondissent. Ce site a servi d'inspiration pour le projet.

- **GitHub de benahm** [consultation pendant tout le projet] :
<https://github.com/benahm/jquery.throwable>

Site associatif : La plate-forme GitHub permet l'hébergement et la gestion de développement de certains logiciels, plugin*, etc.

Le plugin* JQuery Throwable ainsi que toute sa documentation ont été trouvés sur ce site, comme un projet de l'utilisateur benahm. On peut également y trouver des commentaires techniques, des démonstrations d'utilisations, etc.

- **ShareLaTeX** [consultation pendant tout le projet] :
<https://fr.sharelatex.com/>

Site commercial : Ce site permet de coder du LaTeX (langage de programmation) et de le compiler directement en ligne, sans avoir installé quoi que ce soit sur l'ordinateur. Il regroupe également une série de ressources et d'aides concernant le langage. Enfin il est possible par ce site de travailler à plusieurs sur un même document, ce qui fut très pratique lors de la rédaction de ce rapport.

- **Ensemble de forums** [consultation pendant tout le projet] :
<http://www.commentcamarche.net/forum/>
<http://stackoverflow.com/>
<http://openclassrooms.com/forum/>

Sites associatifs : Ces forums ont servi à l'ensemble de l'équipe pour résoudre un certain nombre de problèmes rencontrés. Les termes recherchés ont été par exemple : "Ajout entête LaTeX", "Récupération donnée JavaScript PHP" ou encore "Supprimer un élément CSS avec JQuery".

Annexe A

Dossier d'Analyse

Diagramme de Sequence :

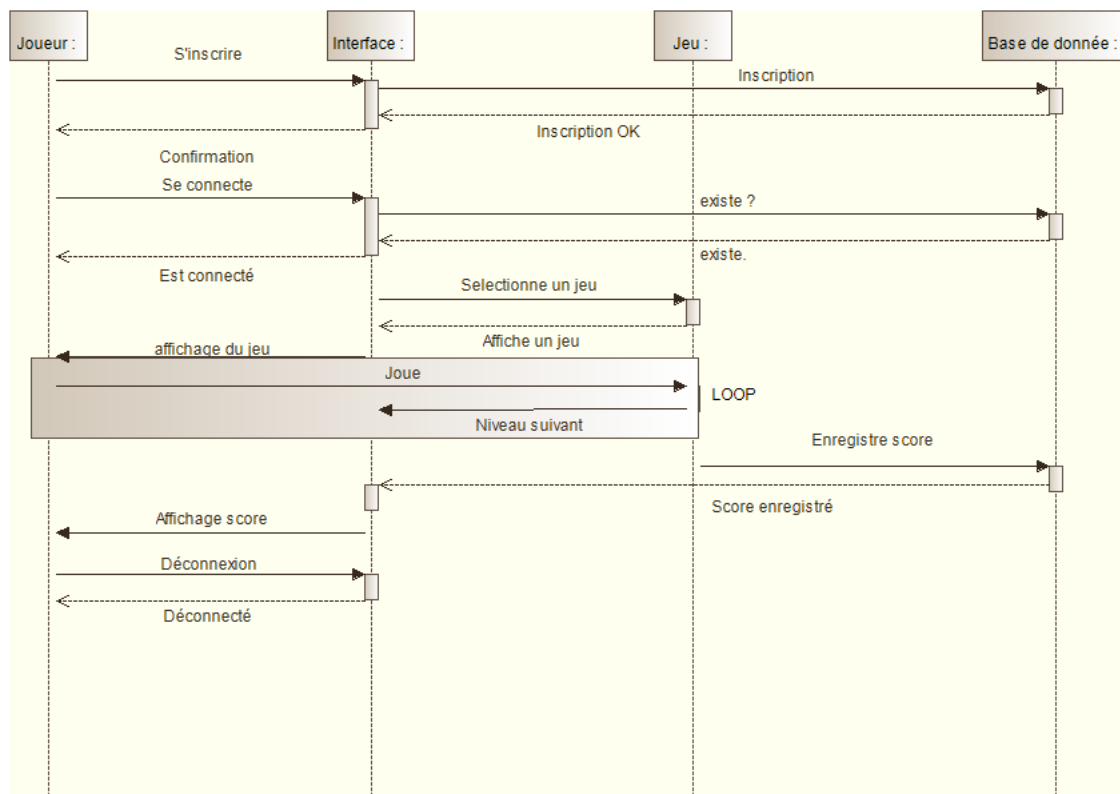


Diagramme d'Etat - Transition :

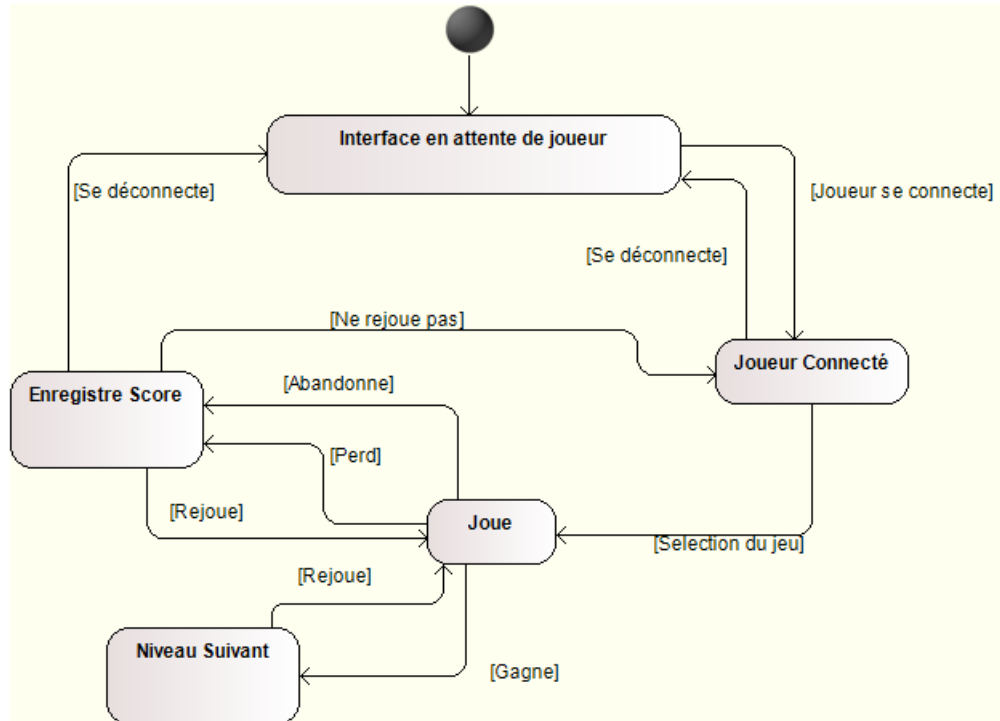
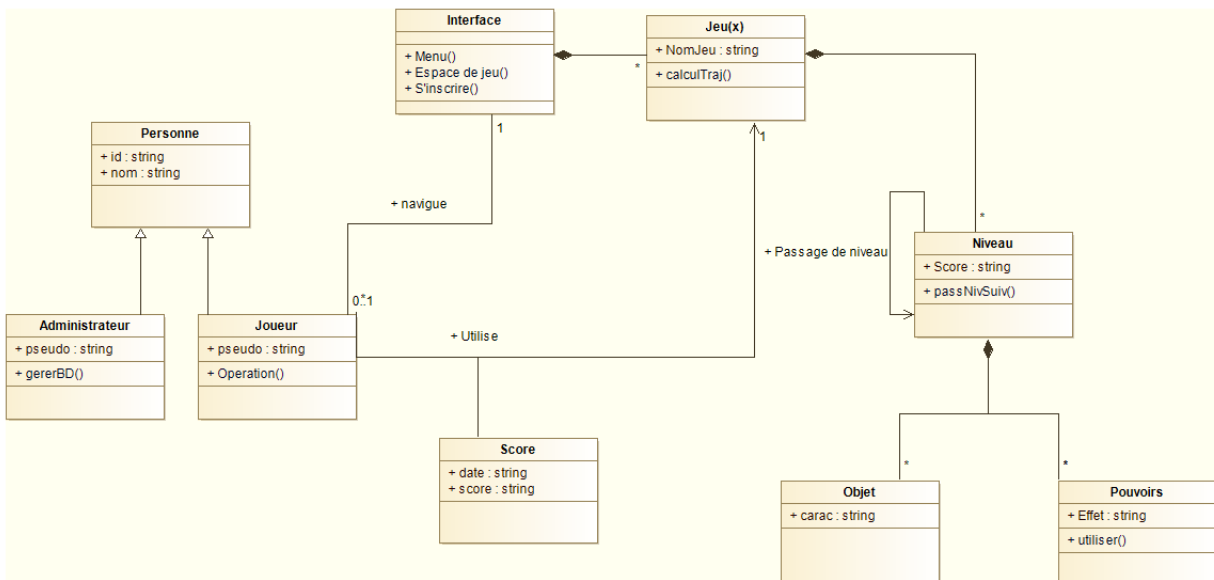


Diagramme de Classe :



Résumé

Le projet Bouncing Cat est un site internet dont le but est de permettre aux utilisateurs de jouer à plusieurs jeux facilement. Le site doit permettre un accès libre pour tout utilisateur et doit comprendre des fonctionnalités adaptées aux deux types d'utilisateurs identifiés : l'administrateur du site et le joueur. Le site Web a été développé en PHP avec BootStrap. Il utilise la bibliothèque JQuery (JavaScript) et une base de données MySQL.

Mots Clés

BootStrap, JQuery, Jeux, MySQL, PHP, Site internet

The Bouncing Cat project is a website whose purpose is to allow users to play at some different games easily. The website has to allow a free acces for every users and has to include fonctions for both types of identified users : the website administrator and the player. The website have been developped in PHP with BootStrap. It use the library JQuery (JavaScript) and MySQL database.

Key Words

BootStrap, Games, JQuery, MySQL, PHP, Website