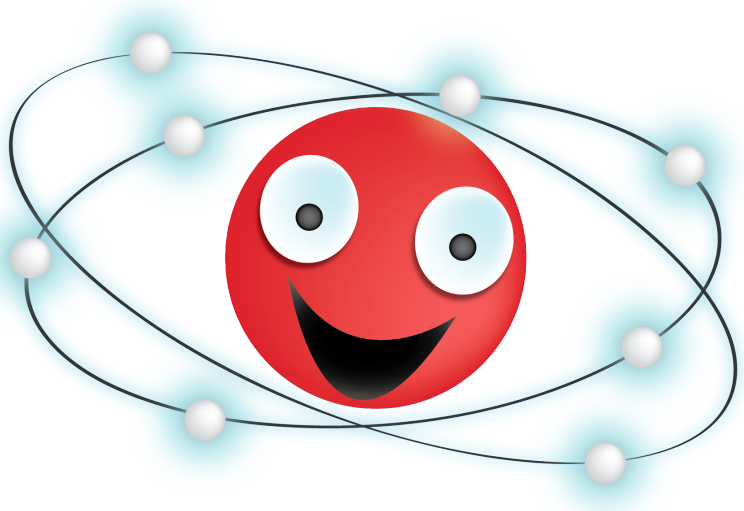
Chemicall

Rapport de projet

Serious Game

Kevin Bollini, Nathalie Glad, Thomas Hinsinger
Geoffrey Mélia, Ailin Moreno



Master informatique

Année universitaire 2012/2013

Table des matières

Table des figures

Remerciements

- Damien Djaouti, pour ses conseils avisés en matière de Game Design.
- Abdelkader Gouaich pour ses cours et pipeline de développement.
- Nastassia Jacquet pour son aide précieuse en matière de sound design et de bruitage.
- Frédéric Tari, pour avoir partagé ses connaissances et son affinité pour la musique.
- Camille Seilles, pour ses réalisations graphiques de notre avatar oxygène sous ses différentes formes.

Première partie

Rapport de projet

Chapitre 1

Introduction

1.1 Présentation du jeu

Dans le cadre du Master 2 informatique, nous avons été amenés à concevoir notre serious game et l'avons axé sur le thème de la chimie.
Notre projet est un jeu de plate-forme 2D à la manière d'un puzzle game, aux graphismes ludiques permettant d'aborder les notions élémentaires de chimie, comme les éléments de base ou l'existence de certaines interactions.

Le joueur incarne un atome d'oxygène qui doit délivrer d'autres atomes, lui permettant par là même d'augmenter ses capacités grâce aux interactions chimiques qu'il pourra alors réaliser. Dans sa quête, il rencontrera des entités ayant un électron en moins ou en plus et qui lui feront perdre ou gagner des électrons : les ions. Il devra sortir du niveau en étant stable afin de gagner un nouvel atome. Dès lors, le joueur apprendra à utiliser ce nouvel atome avec celui d'oxygène afin de créer une molécule ayant des propriétés particulières. Il pourra alors, appeler un de ces atomes gagnés, lors des prochains niveaux afin de modifier son état physique et franchir les obstacles.

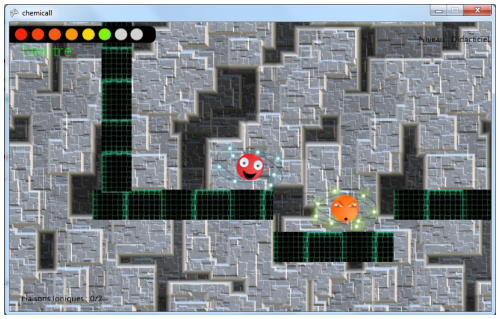


FIGURE 1.1 – Aperçu de l'application

1.2 Contexte

Ce jeu a été pensé et conçu pour les enfants âgés de 10 à 14 ans dont la scolarité amène à apprendre les premières notions de chimie. Le choix de ces notions dans le contenu pédagogique de notre jeu a donc été réalisé en fonction des connaissances chimiques attendues d'un enfant de cet âge.
Nous avons dans un premier temps étudié les notions importantes à mettre en valeur dans notre jeu puis, dans un second temps, la manière dont ces notions pouvaient être intégrées au jeu afin de suivre une visée pédagogique.

Éléments de chimie

L'atome est un élément essentiel de la chimie et en compose la base : nous avons choisi de l'utiliser comme l'élément central du jeu. Notre objectif étant de faire apprendre :

- les notions d'interactions entre atomes ainsi que l'état physique résultant.
- Les relations entre électrons et atomes
- Les différents états de l'atome en fonction de son nombre d'électrons.

Au vue des programmes scolaires de chimie, le premier élément complexe, ou molécule, appris est l'eau. Nous avons donc penser utiliser les atomes le composant : l'oxygène ou l'hydrogène. Étant donné que l'hydrogène n'a que deux électrons et que pour notre idée de jeu, ceci était trop faible, nous avons décidé de garder l'atome oxygène.
De plus, l'oxygène est un élément simple qui s'associe facilement avec d'autres atomes, ce qui était intéressant pour notre gameplay. Nous pouvons facilement créer les molécules simples comme le dioxygène et l'eau.

Intégration

Après avoir réfléchi aux éléments et aux notions de chimie que nous souhaitions intégrer, nous nous sommes intéressés à la manière la plus adéquate pour fondre dans le jeu ces différents éléments.
L'atome étant l'élément central du serious game, nous avons décidé de l'utiliser comme personnage principal. Afin de proposer un challenge au joueur, nous voulions intégrer des personnages ennemis ou alliés dans le jeu. Étant donné que nous souhaitions introduire la notion d'électrons, l'idée d'anion et de cations nous apparu rapidement comme la solution la plus adéquate.
Pour l'association avec d'autres atomes, nous avons pensé à plusieurs possibilités telles que les intégrer directement dans le niveau comme un pouvoir disponible par exemple. Mais ceci ne nous apportait pas de solutions permettant de proposer un challenge intéressant pour finir le niveau. Nous avons donc décidé de l'intégrer à la fin de la réussite d'un niveau et de pouvoir expliquer l'intérêt d'utiliser cet atome.

Chapitre 2

Analyse et conception

2.1 Gestion du projet

Pour le développement de ce projet, nous avons décidé d'utiliser une méthode AGILE. Nous avons procédé par série de sprints, visant à développer des versions prototypes du jeu puis à les enrichir par itérations successives.

Version du Prototype	Contenu et Ajouts majeurs	Durée (Jours)
1	Atome évoluant en 2D sans gravité ni collisions Création du projet et du dépôt	3
2	Atome soumis à une gravité Ajout du moteur Physique	2
3	Atome évoluant sur des plateformes physiques, pouvant effectuer des sauts Ajout des collisions entre entités	3
4	Entités du jeu animés Modification de la classe gérant les sprites pour les animer	2
5	Ajout des entités Anion Et Cation Restructuration des classes du projet	2
6	Atome pouvant échanger des électrons avec les autres Atomes Modification complète des collisions	2
7	Sonorisation du prototype Ajout librairie FMOD	2
8	Niveau pouvant être parcouru via scrolling Ajout d'une caméra et d 'une interface utilisateur	3
9	Ajout d'un menu de jeu Création d'un Game Manager	2
10	Premier prototype avec graphisme définitifs Création des assets finaux et d'un Level Manager	5
11	Prototype avec niveaux pouvant être terminés Ajout d'éléments graphiques à l'interface et création de la sortie de niveau	4

2.2 Outils utilisés

Outils techniques

- Integrated Development Environment : Visual C# 2010
- Gestionnaire de version : Subversion
- Communication : Mails, outils de vidéoconférence, éditeur de documents collaboratif

Langages :

Pour le développement du projet nous avons décidé d'utiliser le langage de programmation C#, après une étude préalable. En effet ce choix n'est pas du au hasard. La principale raison de celui-ci est l'opportunité de découvrir une technologie non traitée au cours de notre formation, dans le cadre d'un projet conséquent. D'autre part, ce langage semblait le plus approprié au projet car permettant un développement assez rapide de jeu grâce au Framework XNA.

Frameworks :

Le jeu utilise trois Frameworks, chacun influant sur un des moteurs du jeu. le choix de l'utilisation de ces composants s'est fait pour permettre l'accélération et la qualité du développement.

XNA est un ensemble d'outils facilitant le développement de jeu sur les terminaux Microsoft. (Xbox360, Windows Phone, Windows)
Notre choix d'utiliser cette technologie fait suite à notre envie d'utilisation d'outils qui ne nous sont pas connus. D'autre part, XNA est très bien documenté et complètement orienté sur le développement de jeux vidéos grands public.

Farseer Physics est un moteur physique libre basé sur le réputé moteur BOX2D. Ce moteur est spécialement conçu pour être utilisé avec XNA, et permet l'implémentation d'interactions physiques réaliste dans le jeu. Ce moteur permet en outre une gestion complète des collisions entre les entités.
Notre choix s'est porté sur ce moteur pour diverses raisons. Tout d'abord, il spécialement conçu pour fonctionner avec XNA. De plus il possède une documentation complète et une communauté active et est le seul portage C# de BOX2D à être constamment maintenu.

FMOD est un moteur sonore permettant une gestion complète de la sonorisation d'un jeu. Initialement conçu pour le langage C, celui-ci possède un binding officiel pour le C#, permettant l'utilisation de la librairie C.
Notre choix s'est porté sur cette technologie car c'est le framework qui nous a été enseigné dans l'UE "Sons et Musique", UE pour lequel une partie de notre groupe réalise la sonorisation du jeu.

2.3 Analyse

2.3.1 Diagramme de classes

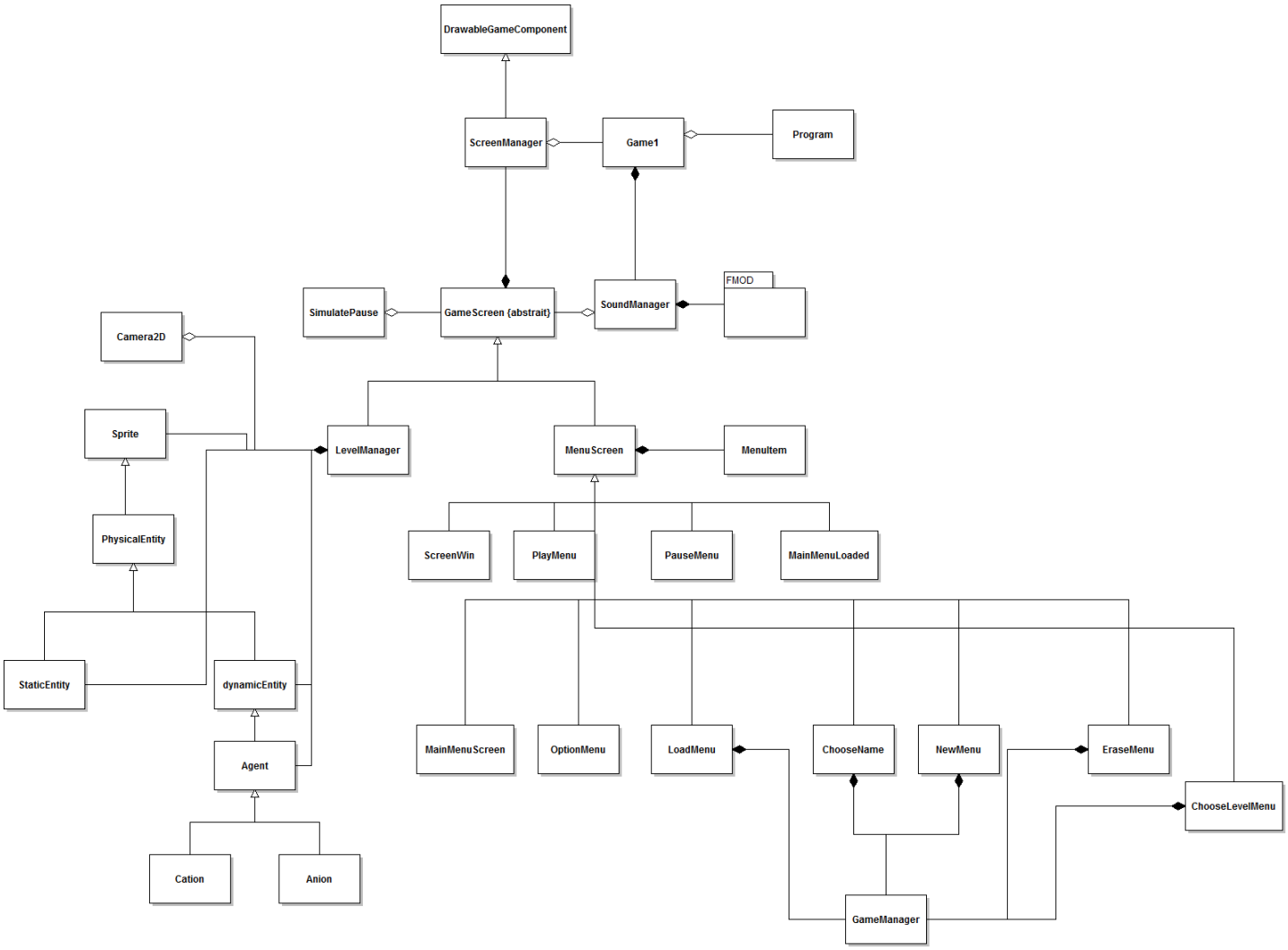


FIGURE 2.1 – Pseudo-Diagramme de classe

Chapitre 3

Réalisation

Dans ce chapitre nous détaillerons les points essentiels de notre réalisation en illustrant une partie des composantes développées pour le jeu.

3.1 Game Engine

La première chose que nous avons réalisé est le moteur du jeu, celui-ci utilise le framework XNA pour les logiques de base (Boucle principale. dessin de la scène etc.) et le moteur physique Farseer. Ainsi, nous avons donc développé des "briques" de bases, utilisant ou non le moteur physique, permettant la construction des niveaux : nous appellerons ces briques les "Entités" ;

3.1.1 Entités

On peut distinguer deux types d'entités, les Sprites et les entités physiques. La différence entre ces deux composants est triviale, l'une n'est pas soumise au contrainte du monde physique auquel il appartiens, à contrario de l'autre.

Sprite

Cette entité est à la base de tout le système, elle permet l'affichage à l'écran à une position donnée d'un élément graphique. Cet élément peut être animé, de la taille de son asset graphique ou non (permettant la création de texture). Cette classe d'entité est mère de toute entité ayant une texture à afficher, que ce soit un élément d'interface, un acteur physique ou simplement un arrière-plan.

Physique

A la différence d'une simple sprite, une entité physique possède un corps tangible, de forme ronde ou polygonale, soumis aux interactions du monde. Cette classe d'entité est donc la base de tout élément interactif du jeu, de la plateforme à l'agent en passant par les éventuels interrupteurs et autres portes. On distingue tout de même deux types d'entités physiques, les entités statiques et celles dynamiques. Il faut noter qu'une entité physique est aussi une sprite, son corps étant dans la majorité des cas un rectangle de la taille de son asset graphique.

Entités Statiques : Elles permettent la création de toutes les entités qui ne bougeront pas dans le jeu (plateformes, portes etc.) L'intérêt de ces entités est qu'elles ne sont soumises à aucune force extérieure (gravité, frottements, etc).

Entités Dynamiques : Elles permettent la création de tout ce qui va être soumis à l'environnement du jeu, les acteurs par exemple. C'est cette différence qui permet d'obtenir une gravité et des mouvements réalistes des entités du jeu. Ces entités peuvent être par exemple des blocs que l'on peut pousser ou un mur que l'on peut briser.

Agents : ils sont une sous classe des entités dynamiques, ils se différencient des entités dynamiques par le fait qu'ils peuvent être animés par un comportement ou un mouvement sur un parcours par exemple.

Character : Il permet le contrôle d'un personnage, il se différencie des agents par le fait que son comportement est dicté par le joueur.

3.1.2 Feedback

Les éléments du feedback donnent au joueur un élément d'immersion dans le jeu. Cette importante constatation nous à amener au développement de quelques éléments de retour comme le menu, les infobulles, la modification dynamique d'assets graphiques ou du son.

Menu : Les menus sont une guide au joueur pour initialiser le jeu, sauvegarder une partie, éliminer une partie, faire une sélection de niveaux, etc. Nous avons dans un premier temps rassemblé les éléments communs à tous les menus dans une classe parente pour ensuite faire chaque menu avec ses options. Nous comptons avec plus de 10 écrans de menu (écran de pause et menu de réglages inclus) pour garantir au joueur une navigation et une utilisation intuitives.

Infobulles : Les infobulles sont des éléments « ingame » que permettent un retour pédagogique au fur et à mesure du déroulement du jeu. Ces éléments donnent des informations au joueur sur, par exemple, les conséquences de toucher un anion/cation. L'idée est de notamment donner au joueur des explications qui peuvent être testées lors du jeu et ainsi doter ce dernier des principes sérieux.

Son : Nous nous sommes également occupés d'inclure dans notre projet des fichiers de son pour la sélection des options de menu, saut et collision des personnages. Ceci permet une immersion tout en contribuant à garantir que le joueur ne s'ennuie pas.

3.2 Game Manager

3.2.1 Utilité

Lors de la conception de notre Serious Game, nous avions trouvé une fonctionnalité intéressante à développer : la sauvegarde et le chargement d'une partie lors de l'exécution de notre jeu. En effet, il est assez déroutant de jouer à une partie et de quitter celle-ci puis, lors du relancement du jeu, de ne plus pouvoir reprendre la partie là où elle en était.

Il était donc important pour nous que cette fonctionnalité soit présente dans notre Serious Game.

3.2.2 Conception

Game Manager

Voici comment se présente ce Game Manager :

```
class GameManager
{
    private static GameManager instance;

    public String fichier;
    public string nameUser;
    public bool[] tNiveau;
    public int[] tScore;
    public bool[] tAtome;

    public static GameManager getInstance();
    private GameManager();
    public void LoadGameManager(string fichier);
    public void SaveGameManager(string fichier, string name);
    public void EraseGameManager(string fichier);
    public void SaveLevel(int numeroNiveau, bool niveau, int score, bool atome);
    public string LoadNameSave(string fichier);
}
```

FIGURE 3.1 – Game Manager

Paramètres :

- GameManager instance : l'instance unique du GameManager s'utilisant dans notre Serious Game
- String fichier : l'adresse de notre fichier de sauvegarde
- string nameUser : le pseudo du joueur
- bool[] tNiveau : un tableau de boolean où sont stockés tous les niveaux réussis au moins une fois par le joueur
- int[] tScore : un tableau d'entiers où sont stockés les scores de chaque niveaux réussis par le joueur
- bool[] tAtome : un tableau de boolean où sont stockés les atomes gagnés par le joueur lorsqu'il termine un niveau en étant stable

public static GameManager getInstance() : fonction permettant de récupérer l'instance unique de GameManager en utilisant le pattern Singleton. Cela nous permet d'être certain qu'au cours de l'exécution de notre Serious Game une seule instance de GameManager est modifiée.

private GameManager() : constructeur d'un GameManager

public void LoadGameManager(string fichier) fonction permettant de charger un GameManager à partir d'un fichier de sauvegarde et donc de charger la sauvegarde du joueur.

public void SaveGameManager(string fichier, string name) : fonction permettant de sauvegarder un GameManager. Cette fonction est appelée à chaque fois que le joueur termine un niveau, et elle se fait donc automatiquement sans que le joueur ait besoin de sauvegarder lui même sa partie.

public void EraseGameManager(string fichier) : fonction permettant de supprimer un GameManager. Cette fonction est utile lorsque le joueur veut créer une partie et que le nombre maximal de sauvegardes est atteint (comme dans Pokemon). Il doit donc supprimer une partie enregistrée pour pouvoir créer sa partie.

public void SaveLevel(int numeroNiveau, bool niveau, int score, bool atome) : cette fonction sauvegarde un niveau dans le GameManager. Elle prend le soin de vérifier si le joueur a déjà réussi à finir ce niveau et si oui ne met à jour le score que si celui-ci est meilleur.

public string LoadNameSave(string fichier) : fonction permettant de trouver le pseudo du joueur à partir du fichier de sauvegarde.

Fichier de sauvegarde

Le fichier de sauvegarde est un fichier texte. Pour que cela soit possible il faut que le fichier soit de cette forme là uniquement :

```
THOMAS
2
True
False
2
0
True
False
```

FIGURE 3.2 – Fichier de sauvegarde

- **Thomas :** le pseudo du joueur
- **2 :** le nombre de niveau disponible dans le Serious Game
- **True**
  False : les niveaux terminés par le joueur
- **2**
  0 : les scores obtenus pour chaque niveau du joueur
- **True**
  False : les atomes obtenus par le joueur

3.3 Level Manager

3.3.1 Utilité

Une autre 'brique' importante dans le moteur de notre Serious Game est le LevelManager. Chemicall est en effet en jeu divisé en plusieurs niveaux comme cela se fait couramment dans le monde du jeu vidéo. Cette décomposition permet de proposer facilement un gameplay évolutif, tant du point du vue du contenu que de celui de la difficulté. Le LevelManager va donc permettre de gérer les différents niveaux du jeu au sein du programme. Il est intimement lié au GameManager puisque chaque joueur, en fonction de son avancement dans le jeu, n'aura par exemple pas accès aux même niveaux ni aux même pouvoirs. C'est aussi lui qui gère les éléments propres à un niveau, leur mise-à-jour ainsi que leur éventuel affichage dans l'interface graphique ou leur transmission au GameManager (sauvegarde des scores, niveaux débloqués, etc).

3.3.2 Fonctionnement et Conception

Description d'un Level

Le LevelManager, comme son nom l'indique, gère les différents niveaux de jeu. Voyons donc dans un premier temps la manière dont sont construits ces Levels.
Nous souhaitions pouvoir créer et utiliser nos niveaux de manière indépendante. Nous avons ainsi décidé d'utiliser un système bien adapté aux stockage de données et à leur manipulation par différents systèmes : le XML. Cela nous permet de bien différencier la description d'un Level de son utilisation dans le jeu, ainsi que de facilement étendre la définition ou le contenu d'un Level. Il a tout d'abord fallu identifier les éléments propres à un Level afin d'en définir la structure. Il est apparu qu'en plus du numéro et du nom du niveau, une simple liste des entités présentes dans le niveau suffisait à décrire ce dernier.

```
public class LevelDescription
{
    public int num { get; set; } //numéro du level, servant aussi d'identifiant
    public String name { get; set; } // le nom du niveau
    public List<EntityDescription> listEntities { get; set; } // liste des entités présentes dans le niveau

    public LevelDescription()...
}
```

FIGURE 3.3 – Description d'un niveau

Une entité étant elle caractérisée par son type, sa position initiale et sa dimension :

```
public class EntityDescription
{
    public int typeID { get; set; } // indique le type d'objet (anion, avatar, etc.)
    public Point initPosXY { get; set; } // position initiale de l'objet sur la map
    public Point dimXY { get; set; } //et ses dimensions.
    public String name { get; set; } // optionnel : nom du fichier image permettant de dessiner l'entité

    public EntityDescription()...
}
```

FIGURE 3.4 – Description d'une entité

Ces descriptions, bien que très simplistes, suffisent à définir l'intégralité du contenu d'un niveau. Cette approche permet par ailleurs d'enrichir aisément le contenu d'un niveau :

- en quantité : en remplissant simplement la liste des entités
- en variabilité : en utilisant un nouveau type d'entité

Cette définition semble donc robuste aux changements, comme nous avons d'ailleurs pu le constater tout au long du projet.

Fichiers XML

A partir de ces définitions, il est alors simple de créer autant de niveaux que nous le souhaitons en créant des fichiers XML respectant cette syntaxe. En voici un exemple volontairement épuré.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <XnaContent xmlns:data="chemicall.data":
3    <Asset Type="data:LevelDescription">
4      <num>1</num>
5      <name>Start Level</name>
6      <nbIons>2</nbIons>
7      <listEntities>
8        <Item>
9          <typeID>1</typeID>
10         <initPosXY>80 0</initPosXY>
11         <dimXY>117 75</dimXY>
12         <name></name>
13       </Item>
14     <Item>
15         <typeID>100</typeID>
16         <initPosXY>300 300</initPosXY>
17         <dimXY>300 100</dimXY>
18         <name>ground</name>
19       </Item>
20     </listEntities>
21   </Asset>
22 </XnaContent>
```

FIGURE 3.5 – Définition d'un niveau en XML

Chargement et exploitation des données

Une fois notre banque de niveaux créée, il faut alors pouvoir charger les données des fichiers XML afin de les exploiter dans l'application. C'est là un des rôles principaux de notre LevelManager. Il est temps de regarder d'un plus près son fonctionnement (Figure 3.6)

```
public static LevelManager getInstance()...

private LevelManager()...

/// <summary>
/// Reset toutes les infos relatives au LevelManager.
/// Fonction à appeler lorsque l'on charge un nouveau niveau
/// </summary>
private void resetLevelManager()...

/// <summary>
/// Permet de charger et de convertir les données issues du fichier XML de définition d'un
/// niveau.
/// </summary>
private void LoadXMLLevelDescription(String XMLLevelFile)...

/// <summary>
/// Permet de charger et de lancer le niveau numLevel
/// </summary>
/// <param name="numLevel"></param>
public void LoadLevel(int numLevel)...

/// <summary>
/// Termine le niveau en cours et transmet les informations au GameManager
/// </summary>
/// <param name="etat"></param>
public void EndLevel()...

/// <summary>
/// Allows the game to run logic such as updating the world,
/// checking for collisions, gathering input, and playing audio.
/// </summary>
/// <param name="gameTime">Provides a snapshot of timing values.</param>
public override void Update(GameTime gameTime, bool otherScreenHasFocus, bool coveredByOtherScreen)..

/// <summary>
/// This is called when the game should draw itself.
/// </summary>
/// <param name="gameTime">Provides a snapshot of timing values.</param>
public override void Draw(GameTime gameTime)...
```

FIGURE 3.6 – Level Manager

Notons tout d'abord qu'il ne peut y avoir qu'une instance unique du LevelManager. Il ne serait pas utile d'en avoir plus et cela risquerait de créer des conflits.

Méthodes importantes :

- **LoadLevel(int numLevel) :** Appelée lorsque l'on souhaite charger (ou recharger) un niveau. Elle se charge d'appeler la méthode resetLevelManager() pour réinitialiser les données puis charge les informations spécifiques au niveau souhaité grâce à la seconde méthode LoadXMLLevelDescription.

- **LoadXMLLevelDescription(String XMLLevelFile) :** Permet de de lire le fichier XML de définition du niveau, et convertit les données lues en données utilisables par l'application en suivant les descriptions LevelDescription et EntityDescription détaillées en 3.3.2.

- **EndLevel() :** Se charge d'envoyer l'état des informations importantes à la sauvegarde de données au GameManager, en utilisant notamment sa fonction SaveLevel (cf 3.2.2).

Chapitre 4

Résultats

4.1 Assets Graphiques

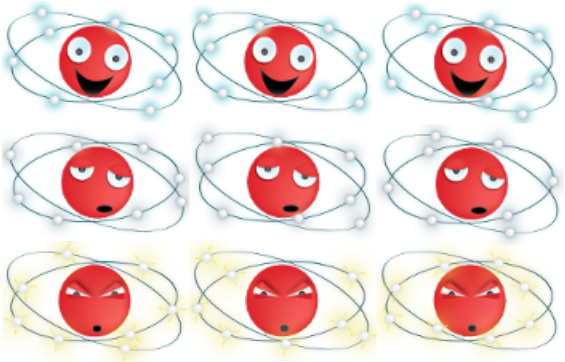


FIGURE 4.1 – SpriteSheet de l'oxygen

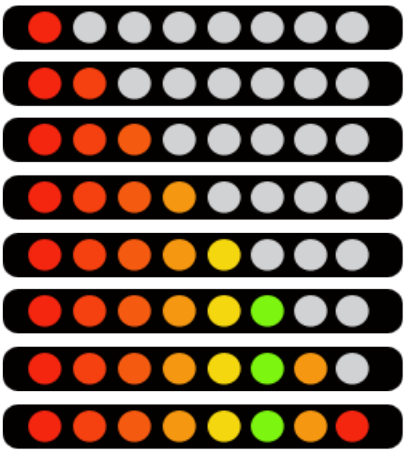


FIGURE 4.2 – Interface de décompte des électrons

4.2 Prototype

4.2.1 Menus

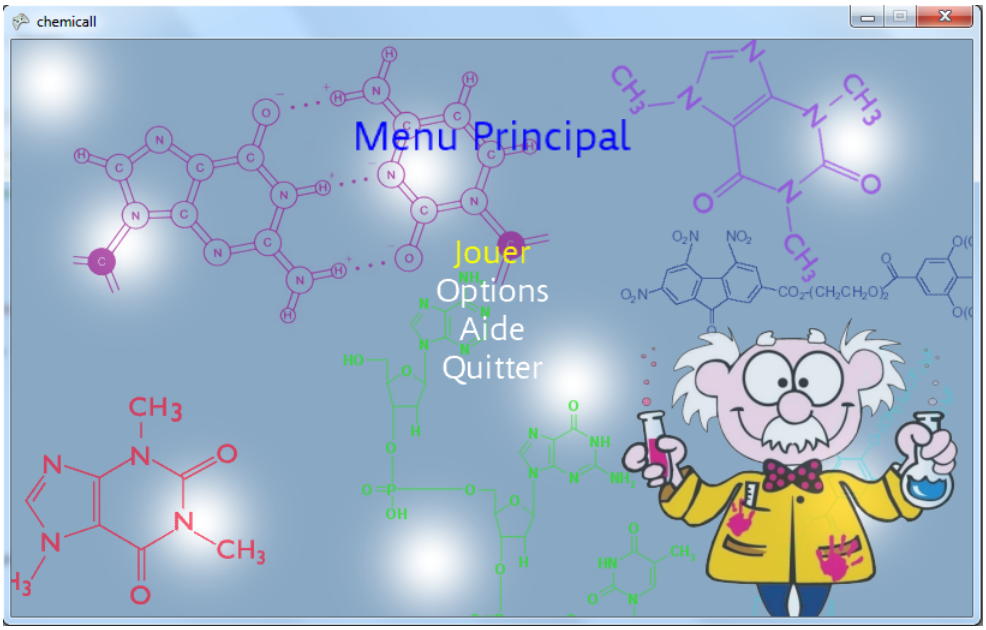


FIGURE 4.3 – Menu du jeu

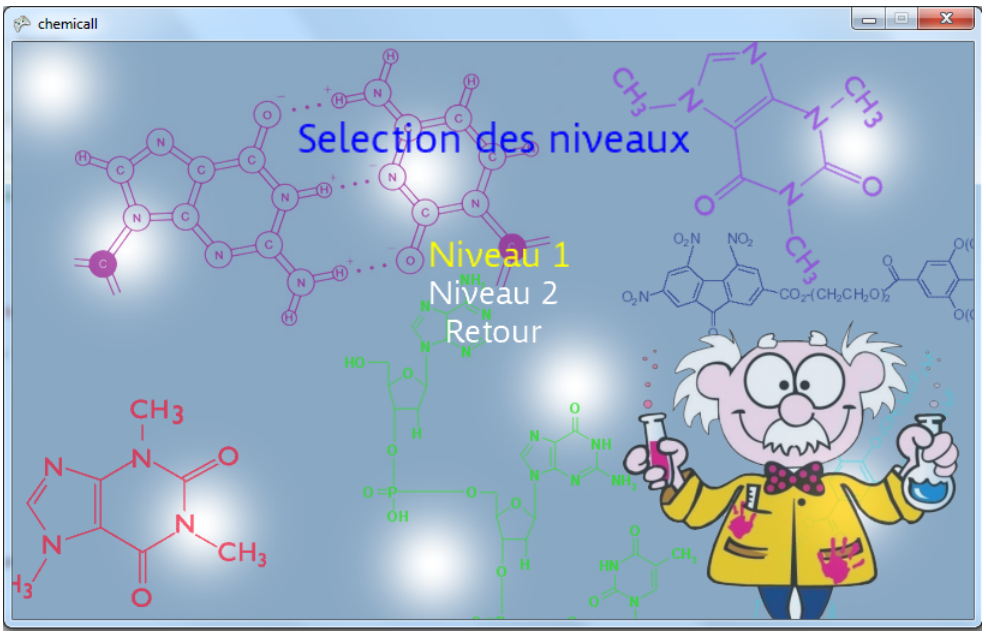


FIGURE 4.4 – Sélecteur de niveaux

4.2.2 Ingame

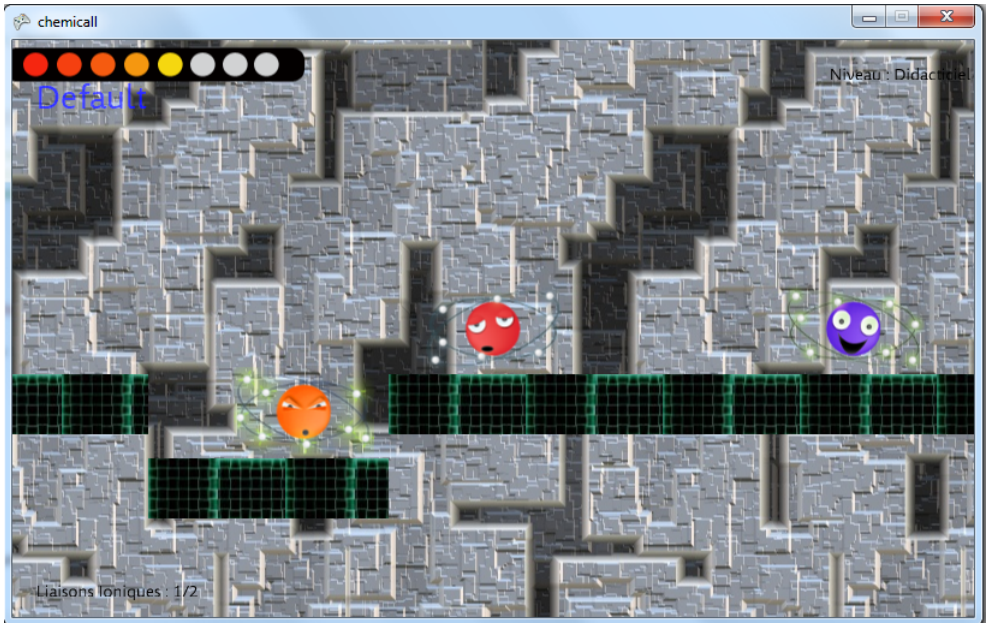


FIGURE 4.5 – Capture ingame du jeu

FIGURE 4.6 – Capture ingame du jeu avec feedback

Chapitre 5

Discussion

5.1 Difficultés rencontrées

Le projet étant un serious game basé sur les propriétés fondamentales de la physique et de la chimie, la première difficulté a été d'imaginer un gameplay autour de ces propriétés,en conciliant plaisir de jeu et respect des lois physico-chimiques.

5.1.1 Synchronisation

Pour un projet d'une ampleur comme celle de Chemicall, un groupe de travail de cinq personnes travaillant toutes à distance impose nécessairement des exigences fortes quant à la synchronisation et aux méthodes de travail. La méthode AGILE utilisée a pour concept la répartition du travail par affectation de tâches. Or toutes les tâches ne peuvent pas être indépendantes et il arrivait donc fréquemment qu'il faille attendre qu'une tâche affectée à une autre personne soit terminée pour que la notre puisse fonctionner. Nous avons bien sur réparti les tâches de manière à ce que cette situation soit la moins fréquente possible tout en tentant de répartir le travail de manière équilibré et en fonction des capacités et souhaits de chacun.
C'est donc un gros travail d'équipe qu'il fallut faire, et la communication au sein du groupe a été très importante pour garantir un avancement constant du travail sans tâche bloquante retardée.

5.1.2 Technique

Une des difficultés majeures du projet fût de réussir à importer toutes les librairies externes et faire les fonctionner correctement entre elles. La librairie sonore Fmod notamment n'est pas destinée pour le C# et a demandé plus de temps et de recherche avant de fonctionner. D'autre part, étant tous des programmeur et en aucun cas des graphistes, la création des assets graphiques à été assez chrono-phage et très laborieuse. Enfin, toute les technologies utilisées étant nouvelles pour nous, une des grandes difficulté à été le temps de formation de chacun pour réussir à correctement tout manipuler.

5.2 Impact Sérieux

5.2.1 Elements graphiques

Nous allons détaillé la manière dont les éléments ont été pensé afin de réaliser une visée pédagogique.
Les éléments graphiques ont une place importante dans la compréhension du jeu. Comme détaillé auparavant, les différents éléments graphiques choisis permettent d'intégrer des éléments de feedback à une dimension "sérieuse".

Concernant l'affichage du joueur, celui-ci commence stable donc nous avons choisi une image représentant le bien-être avec un sourire. Dès lors que le joueur n'est plus stable ,plusieurs éléments graphiques lui permettent de comprendre l'état du jeu :

- Un affichage texte s'affiche, permettant d'informer le joueur. Le mot "exces" ou " default" s'affiche si le joueur a reçu ou perdu , respectivement, un électron.
  L'affichage de texte permet de faire comprendre rapidement au joueur, dans quelle situation il se trouve et si il est en excès ou en manque d'un électron.
- La barre du nombre d'électrons se modifie. Si ce dernier lui en manque ou en possède trop, il passera alors dans le rouge permettant d'indiquer au joueur que cette situation n'est pas la bonne. En effet, plusieurs recherches ont montrés que

le rouge était associé au mal ou à l'erreur et à l'inverse, le vert comme étant juste ou correcteur, c'est pourquoi nous avons choisi trois couleurs : le rouge, le vert et le jaune. Le jaune étant uniquement la transition entre le vert et le rouge. Ceci est uniquement psychologique. Il permet au joueur d'avoir un feedback rapide sur sa situation.
- L'affichage de l'atome du joueur change. Il possède alors le même affichage qu'un atome trop/peu chargé comme les éléments du jeu qu'il peut rencontrer. Afin de modéliser visuellement, cette échange d'électrons, l'anion/cation touché devient stable et prends donc l'affichage adéquat et le joueur se retrouve alors dans un état d'excès ou de manque et cela se visualise par l'affichage de l'agent touché.

5.2.2 Visée pédagogique

Nous allons nous intéresser à ce que le joueur apprend au cours du jeu.

Anion / Cation

A la suite de l'apprentissage des règles du jeu, le joueur comprend qu'il peut gagner ou perdre un électron comme dans la chimie réelle. Pour cela, il doit rencontrer des autres atomes, ayant un électron en plus ou moins, s'appelant respectivement anion et cation. Il s'agit d'une information implicite, le joueur ne se souviendra peut-être pas du nom mais de l'existence de ces deux états de l'atome possible.

Electrons

Grâce à la barre d'électron situé en haut à droite, le joueur apprend qu'un atome possède des électrons. Le nombre a été fixé à six, représentant les six électrons de la seconde couche de l'atome. Cependant, le but pédagogique visé est de faire comprendre au joueur les interactions possibles entres les atomes en échangeant des électrons. Ainsi, le joueur peut comprendre l'influence d'un élément sur son personnage et de comprendre que le personnage principal, représenté ici, ne doit pas être autre que stable.

Associations d'atomes et leurs états physiques

A la suite d'un niveau réussi, le joueur découvre un nouvel atome qui peut être associé à celui de son joueur afin de réaliser une molécule ayant une nouvelle propriété physique. Le joueur apprend différentes informations :
- Le nom d'un nouvel atome.
- L'association possible avec son atome ainsi que la formule chimique.
- La propriété physique associé à cette association.

Afin d'illustrer nos propos, en voici un exemple :
A la suite du premier niveau, nous gagnons l'atome hydrogène. Une fenêtre s'affiche et nous explique alors que l'hydrogène possède deux électrons et qu'il peut en partager un avec nous. Étant donné qu'il nous reste la possibilité de prendre encore deux électrons, on peut alors associer deux hydrogène avec l'atome oxygène afin de créer une molécule : l'eau.
On nous indique la formule chimique de cette molécule : H_2O
De plus, on apprend alors l'état physique de l'eau : liquide. On pourra alors passer à travers une grille.
A travers cet exemple, on retrouve les différents éléments cités dans la section auparavant.

Afin de faciliter la compréhension pour l'utilisateur, nous avons décidé de réaliser des assets graphiques.
Nous avons réalisé les spriteSheet (Figure 5.1) de l'atome d'oxygène mais également ceux des anions et cations permettant d'avoir des animations. La première ligne représente l'animation stable, son expression renvoit à une certaine joie. La seconde ligne represente l'animation dès lors qu'il possèdera un ou des électrons en plus,il semble fatigué. La dernière ligne représente l'animation dès lors qu'il lui manque un ou plusieurs éléctrons. Son visage est fermé, il est excité.
Il était important que la représentation de ces animations soit compréhensible par tous, c'est pourquoi, nous avons utilisé des expressions très différentes et significatives.

Concernant l'interface, nous avons réalisé un spriteSheet (Figure 4.2) modélisant le nombre d'électrons que possède le joueur. Comme nous l'expliquerons plus tard, la connotation du rouge et du vert n'est pas anodine. Les dessins ronds rappellent la forme d'un électron.

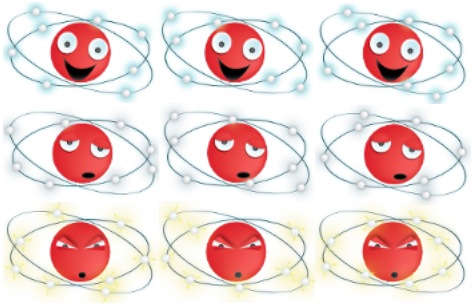


FIGURE 5.1 – SpriteSheet de l'oxygène

Chapitre 6

Conclusion

6.1 Conclusion

Nous sommes assez fier quand au résultat de notre prototype. Nous avons réussit à faire pratiquement tout ce que nous voulions faire. Notre avons ensemble longuement réfléchit à la conception de se prototype dans l'objectif non pas de rendre seulement un prototype mais un Serious Game complet et "commercialisable".

En effet, tout à été construit dans cette objectif là. Nous avons créer l'ensemble des entités que nous souhaitions mettre en place dans notre Serious Game. Nous avons qu'un prototype à ce jour, mais le LevelManager nous permet aisément de créer un niveau complet et bien construit. Il est donc facile par la suite de créer tous les niveaux nécessaire pour avoir un Serious Game complet et plaisant à jouer.

Tout à été pensé pour faciliter la création par la suite de notre Serious Game, le LevelManager pour la création de niveaux, le GameManager pour la sauvegarde et chargement de parties, SoundManager pour la créations des différents sons, ScreenManager pour la création des fenêtres de notre jeu, etc...

Nous avons préférés construire quelque chose de bien fait, bien réfléchit que beaucoup de chose faite n'importe comment.

6.2 Perspectives

Transformations : Il manque à ce jour certaine choses que nous aimerions faire par la suite. Comme par exemple l'association de notre personnage principal (l'Oxygène) avec un atome gagné. En effet, si le jouer termine un niveau et gagne l'atome d'Hydrogène, il doit pouvoir s'associer avec celui-ci pour se transformer en molécule H2O, l'eau. Et donc passer au travers de certaines zones du niveau, comme par exemple une grille. Évidemment, tout est en place pour pouvoir créer ces choses.

Le menu : Nous avons un menu correct qui nous convient plus l'instant, mais il pourrait être amélioré et optimisé. En effet nous ne sommes pas des graphistes, nous ne sommes pas très doué dans ce domaine là. Ainsi notre background est très simple. De plus il manque quelques points à rajouté dans le menu (comme par exemple, la description détaillé des molécules H2O, CO2, etc... pour augmenter le côté sérieux de notre application), ou encore la possibilité de changer les raccourcis par défaut de notre application des les options, et d'autres choses encore.

Les funnyfacts : Nous avons mis en place la création d'infobulles permettant au joueur de comprendre ce qu'il se passe autour de lui. Nous avions dans l'idée de rajouter des funnyfacts lors d'un chargement d'un niveau pour apprendre des choses amusants sur la chimie au joueur. Cela augmenterait encore plus le côté sérieux de notre application.

Les récompenses : Nous voulions aussi mettre en place un système de badges pour différencier les joueurs. En effet en plus d'avoir un score par niveau, il aurait la possibilité de gagner des badges. Par exemple, si le joueur arrive à stabiliser tous les anions et cations du niveau en plus de rester stable, il gagnerait le badge Chimiste accompli. Cela rajoute un sentiment de satisfaction au joueur qui est intéressant à mettre en place.

Chapitre 7

Références

Sources web
- https ://fr.wikibooks.org
- http ://www.demonixis.net
- http ://fr.wikipedia.org

Deuxième partie

Annexes

Annexe A

Diagrame de Classes

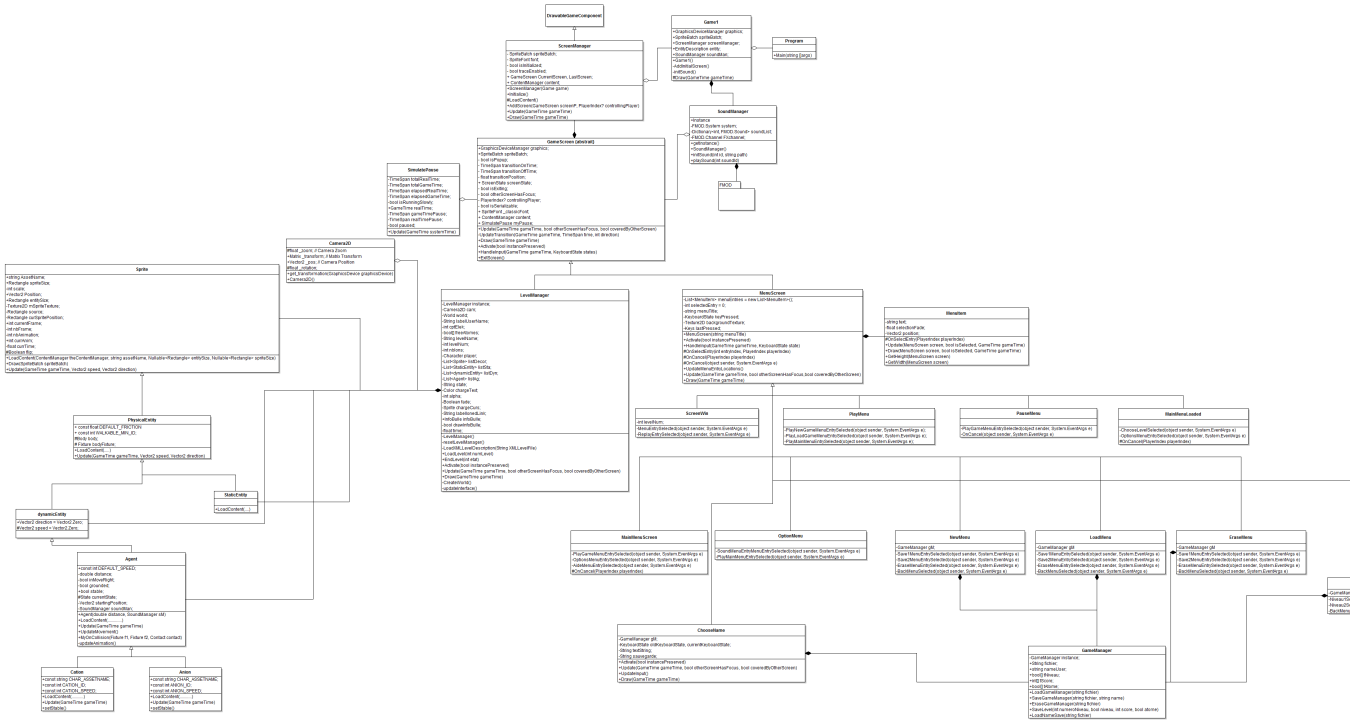


FIGURE A.1 – Aperçu de l'application

Annexe B

Game Design Document

Serious Game

Game Design Document
Chemicall

Nathalie Glad
Thomas Hinsinger
Geoffrey Melia

Ailin Moreno
Kevin Bollini

5 Décembre 2012

Table des matières

1 Pitch du jeu

Notre petit ami l'atome d'Oxygène se retrouve bloqué dans un monde imaginaire. Vous devez l'aider à se sortir de ce monde pour retrouver ses amis atomes. Mais plein d'obstacles se dressent sur son chemin. Il devra franchir ces obstacles à l'aide de ses amis Hydrogène, Carbone (et les autres !). Mais pour se faire il devra au préalable les libérer au cours de son épopée fantastique. Libérez les tous et affrontez le boss final pour enfin le sortir de sa prison.

2 Coeur(s) de gameplay

2.1 Mono-Core

- Jeu de plateforme 2D/casse tête.
- Traverser un niveau tout en évitant les obstacles (trou, mur, etc...), et les ennemis (anions, cations) pour finir le niveau en restant stable.
- Si le joueur termine le niveau en restant stable, il gagne l'atome correspondant au niveau. Il pourra par la suite utiliser cet atome pour se transformer en molécule. On appel cela une compétence : si l'atome d'Oxygène gagne l'atome d'Hydrogène, il peut alors se transformer en molécule H20 et se transformer en eau, pour traverser certains obstacles (une grille par exemple).
- Le joueur doit donc combiner toutes ses compétences pour terminer le niveau tout en restant stable.

3 Contrôles

- Déplacer (Gauche et Droite)
- Sauter
- Utiliser abilitée
- Changer abilitée

4 Camera

- Scrolling Horizontal et Vertical

5 Acteurs

5.1 Joueur

- Le joueur représentant l'atome initial, l'Oxygène.
  Voici à quoi il pourrait ressembler :

FIGURE 1 – Atome d'Oxygène stable

FIGURE 2 – Atome d'Oxygène en surcharge

FIGURE 3 – Atome d'Oxygène en sous charge

5.2 Ennemis

– Cations : possèdent un électron en moins.

FIGURE 4 – Un Cation stable

FIGURE 5 – Un Cation en sous-charge

– Anions : possèdent un électron en trop.

FIGURE 6 – Un Anion stable

FIGURE 7 – Un Anion en sur-charge

5.3 Éléments

– Interrupteurs : Altèrent la température du niveau.
– Atomes : peuvent être appelés par le joueur pour utiliser leurs abilitées.
– Pièges : trous que le joueur doit éviter, obstacles à éviter, etc...

6 Feedback

6.1 ingame

Affichage du nombre d'électrons possédant le joueur avec un code couleur
- < 6 : orange->rouge (0 = niveau perdu)
- $= 6$: vert (0 = niveau perdu)
- > 6 : orange->rouge (0 = niveau perdu)

FIGURE 8 – La barre de vie du joueur lorsqu'il est stable

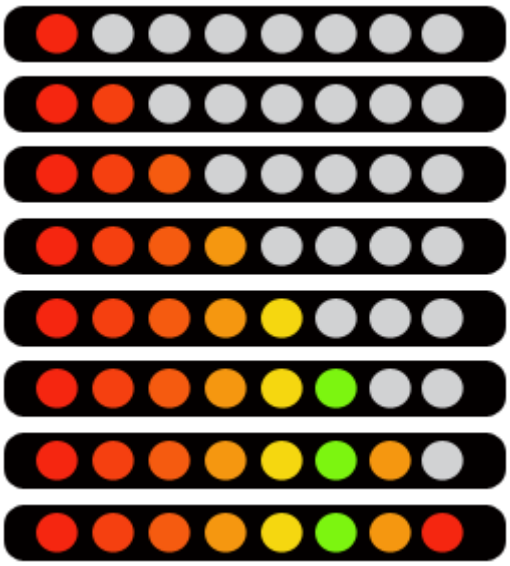


FIGURE 9 – Toutes les barres de vie possible pour le joueur

6.2 Menu

Affichage des niveaux réalisés et des atomes gagnés.
Possibilités à tout moment du jeu, d'afficher les atomes possédés par le joueur ainsi que l'explication de leurs associations (2 parties : une théorique pour l'aspect éducatif, l'autre pratique pour le gameplay)

6.3 Sons

- Musique d'ambiance à l'intérieur du niveau changeant en fonction du nombre d'électrons possédés par l'atome d'Oxygène
- Bruitage lorsque l'atome d'Oxygène touche un anion ou un cation

7 Assistance

- Affichage du nombre d'électrons que possède le joueur
- Affichage lors de la réussite d'un niveau, de l'atome gagné
- Explication de l'association entre l'atome gagné et celui d'oxygène
- Explication de la molécule pouvant être formée, ainsi que son état et son intérêt

8 Boucle OCR

8.1 Boucle Principale

- Objectif : Réussir le niveau sans "mourir" (en gardant au minimum 1 électron).
- Challenge : Finir le niveau avec exactement six électrons.
- Récompense : Nouvel atome gagné.

8.2 Boucle "Quête-Annexe"

8.2.1 Quête-Annexe 1

- Objectif : Équilibrer tous les atomes, cations et anions, en leur retirant ou ajoutant des électrons.
- Challenge : Stabiliser chaque atomes en évitant de se déséquilibrer fatalement soi-même.
- Récompense : Badge Chimiste accompli.

8.2.2 Quête-Annexe 2

- Objectif : Gagner tous les atomes.
- Challenge : Finir chaque niveaux avec exactement six électrons.
- Récompense : Badge Tableau périodique des éléments.

9 Règle du jeu

Vous incarnez un atome d'Oxygène. Vous possédez un nombre précis d'électrons à votre départ. Vous pouvez perdre ou gagner des électrons. Le but du jeu est de finir le niveau avec au moins six électrons. Pour cela, vous avez des alliés, nommé Anions, ils possèdent un électron en trop, ils pourront alors vous le donner. Cependant une fois l'échange effectué, l'anion devient alors stable, et il ne peut vous en donner d'autres. Vous aurez aussi des ennemis, des cations, il leur manque un électron et donc à leur contact, il vous prendront un électron sur la totalité que vous possédez (game over si ce total tombe à 0). Lorsque que vous démarrez le jeu, vous avez la possibilité de choisir parmi les premiers niveaux disponibles. Chaque niveau indique l'atome pour lequel vous jouez. Si vous finissez le niveau avec au moins six électrons, vous gagnerez cette atome. Une explication vous sera donnée sur le fonctionnement de cet atome avec le votre (Oxygène O) et les possibilités qui s'offrent à vous Dans les prochains niveaux, vous pouvez alors appeler le (ou les) atome(s) gagné(s) afin de vous transformer en une molécule avec un état physique particulier. Il vous aidera à passer les obstacles ou à vous défendre.

10 Menu

- Jouer
- Règles du jeu
- Options (Son, Contrôles)
- Récapitulatif de toutes les propriétés physiques/atomiques découvertes
- Quitter le jeu

Annexe C

Etude Comparative

Serious Game

Etude Comparative
Chemicall

Nathalie Glad
Thomas Hinsinger
Geoffrey Melia

Ailin Moreno
Kevin Bollini

5 Décembre 2012

Table des matières

1 Introduction

Ce document à pour but de montrer la recherche que nous avons fait au cours de notre projet sur le Serious Game. En effet avant de nous lancer dans la conception de notre jeu, et d'appliquer nos idées, nous voulions d'abord comme nous l'avait conseillé et demandé Damien Djaoutti. Il aura donc dans ce qui suit tout ce que nous avons pu trouver sur les Serious Game en tout genre traitant la thématique de notre jeu : la Chimie.

2 Super Kimy

FIGURE 1 – Super Kimy

2.1 Explication du jeu

Ce Serious Game est un jeu en ligne (de type Flash donc). Il a pour but d'apprendre la chimie en résolvant des problèmes concrets comme la raréfaction de l'eau potable, la lutte contre le cancer, l'augmentation des gaz à effet de serre. Pas moins de 15 problèmes de ce type existe dans ce Serious Game. Chaque problème se pose sous forme de mini-jeux.

L'héros de ce Serious Game s'appel Kimmy et à l'aide de ses amis Kimics avec l'aide des innovations de la chimie contre Nefastos bien décidé à anéantir la terre de ses habitants.

2.2 Public

Le public visé de cette application est les enfants de 3 à 11 ans, les plus petits essayant juste de s'amuser comme ils le peuvent tandis que les plus grands essayeront de comprendre le message que ce Serious Game essaye de faire passer

3 Foldit

3.1 Explication du jeu

Foldit est un Serious Game de type Jeu/Recherche scientifique. Le but de ce "jeu" est le pliage de protéine. En effet vous vous retrouvez des structures tridimensionnelles de protéines et vous devez les plier pour les organiser dans l'espace du mieux qu'il est possible. C'est dans un genre de puzzle game. Ces énigmes se résolvent sur internet.

A l'heure d'aujourd'hui il n'existe pas d'algorithme efficace pour résoudre ce genre d'énigmes et ainsi ce Serious Game met à contribution le grand public pour aider la recherche Scientifique à résoudre ces énigmes.

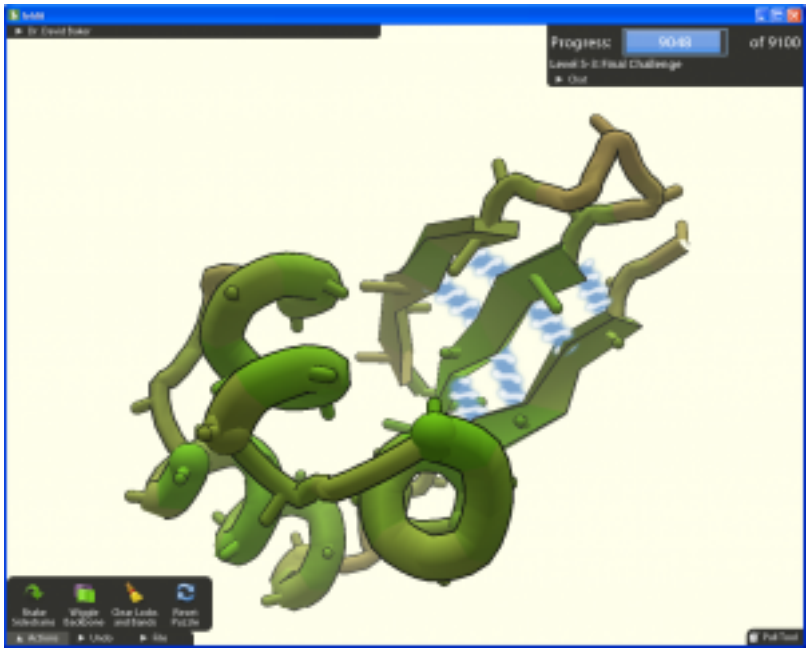


FIGURE 2 – Foldit

3.2 Public

Le publique visé est dans le grand public. Il n'est pas demandé d'avoir de grandes connaissances en Chimie et pour ainsi dire aucune connaissance n'est nécessaire. Il faut juste avoir un peu de logique, et essayer de résoudre ces puzzles.

4 Mechem

FIGURE 3 – Mechem

4.1 Explication du jeu

Ce Serous Game met en jeu des robots. Il faut en effet vaincre les robots ennemis lors d'affrontements multi-joueurs. Pour parvenir à ceci on vous propose de personnaliser vos robots à partir de différents éléments chimiques. Ces éléments

vous apportant des compétences et vous pouvez donc par la suite utiliser ces compétences pour vaincre les robots ennemis.

Le but de ce Serious Game est d'apprendre les rudiments de la chimie. Il veut donc diffuser un message éducatif. Il se joue de plus sur internet exclusivement.

4.2 Public

Le public visé par cette application est principalement les collégiens-lycéens (12-16 ans) voir moins (8-11ans).

5 Projet M2C

FIGURE 4 – ProjetM2C

5.1 Explication du jeu

Ce Serious Game de type jeu d'enquête Point&Click permettra au joueur de découvrir les 10 phases de la conception d'une molécule de chimie en un produit finit. Il mettra donc en avant les métiers de projet de chimie. Ce jeu dispose de très bonnes informations sous formes de fiches (les nanomédicaments en action, la chimie verte, le fard égyptien, les colorants dans les aliments, les pigments du peintre, les parfums, les matériaux textile dans le sport, les pictogrammes, etc...).

Il est donc très intéressant pour quelqu'un qui s'intéresse à la chimie. Cependant quelqu'un ne s'y intéressant pas du tout ne trouverait pas le plaisir d'y jouer. En parlant de jouer, ce jeu possède en plus des mini-jeux pour comprendre certains effets chimiques. Il se joue aussi exclusivement sur internet.

5.2 Public

Le public visé par ce Serious Game est les adolescents entre 12-16 ans.

6 Le retour d'une mystérieuse maladie

6.1 Explication du jeu

Ce Serious Game vous place sur une île mystérieuse ou vous êtes l'un des chercheurs expert en génomique. Cette île appelée Génomia peuplée uniquement de nain est mise en péril par l'apparition d'une mystérieuse maladie mettant en péril la population locale.

On vous propose donc d'enquêter sur cette maladie. Vous devez donc prélever des échantillons de peau pour en extraire l'ADN. Par la suite vous pouvez réaliser des tests en laboratoire à la manière d'un vrai chercheur en génomique afin de trouver un remède en utilisant l'esprit de déduction.

FIGURE 5 – maladie

6.2 Public

Le public visé par ce Serious Game est les Lycéens (entre 12-16 ans) et il ne fonctionne uniquement sur internet.

7 Autres jeux

Il existe d'autres jeux basés sur la thématique de la Chimie, cependant trop peu d'informations sur ceux-ci pour pouvoir en faire une description fiable. Voici ces jeux :

– Elemental

FIGURE 6 – Elemental

– Terraform

FIGURE 7 – Terraform

FIGURE 8 – CoumpoundReaction

- Compound Reaction
- Science en jeu

FIGURE 9 – Science en jeu

- Ludo Quimico

FIGURE 10 – Ludo Quimico

8 Références

– http ://serious.gameclassification.com/