Plateforme de création de "mini" jeux 3D sur le WEB

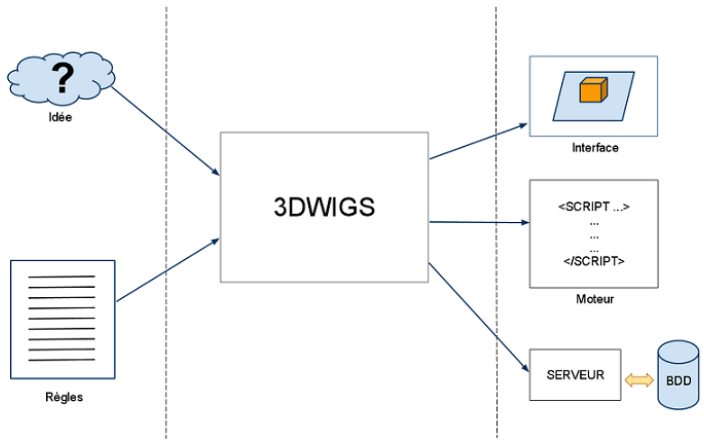
A. Berlon B. Bouzillard W. Chegham T. Clergeau A. Faghihi
M. Guichaoua Q. Israël K. Kien G. Lamine T. Le Corron
B. Le Galludec E. Le Normand A. Lubecki D. Marginier
A. Sanvoisin A. Tolba Mohamed P. Weinzaepfel L. Zadith

5 décembre 2010

1 Présentation du sujet

Ce projet consiste à créer un outil auteur de mini-jeux 3D destinés au Web. En effet, les récentes arrivées de HTML 5 et d'outils 3D comme WebGL permettent désormais l'affichage d'objets 3D dictement intégrés dans les pages internets. Cependant, les contenus 3D dans les pages Web ne sont actuellement que très peu interactifs. Fort du succès des jeux flash, les contenus interactifs 3D comme les mini-jeux ont naturellement leur place sur le Web. Or s'il existe de nombreux outils tel que Google SketchUp pour créer et manipuler les objets 3D en eux-mêmes, il reste tout de même un effort important à faire en ce qui concerne les interactions avec ceux-ci et en particulier la création de mini-jeux en 3D.

La conception d'un outil de création de mini-jeux 3D pour le Web nécessite, d'une part la description des objectifs, des règles, des interactions, du scénario du jeu qui permettra de générer le code du jeu, et d'autre part les élément 3D constitutifs. Il peut également être souhaitable de pouvoir sauvegarder une partie : un jeu ne peut pas forcément se finir en quelques minutes, le fait de pouvoir reprendre une partie commencée un autre jour est alors nécessaire.



Le schéma général du projet est présenté sur la Figure 1. La partie gauche représente un fichier écrit par l'utilisateur contenant les règles du jeu qu'il souhaite créer respectant une certaine syntaxe qu'il faut définir.

Il s'agit de créer un langage, à la fois suffisamment abstrait pour être accessible à n'importe quel utilisateur, mais également suffisamment riche pour pouvoir décrire un maximum de jeux. Le langage est défini par une grammaire. Cette dernière doit permettre de décrire à la fois :

- les objectifs ;
- les règles ;
- le scénario, les niveaux, la logique de score ;
- les interactions avec le(s) joueur(s).

Le challenge est difficile car il existe de très nombreuses catégories de jeux : par exemple, un jeu de gestion n'a, à première vue, aucun point commun avec un jeu de volley ou un jeu de plateforme.

Il serait illusoire de vouloir décrire absolument tous les jeux à l'aide d'une seule et unique grammaire. En effet, les jeux décrits par une grammaire sont forcément restreints.

Toutefois, de nombreuses similarités existent entre plusieurs jeux. Il s'agit donc de les exploiter afin de définir un langage général de description de jeux.

Le fichier définissant le jeu via la grammaire se fera à l'aide d'un éditeur spécial, par exemple créé via eclipse.

La partie centrale du schéma correspond à un compilateur. Il permet de convertir le fichier de description du jeu en un script javascript exécutable dans une page Web et permettant de jouer et interagir avec l'environnement. Le langage javascript a été choisi pour la génération du jeu car il est actuellement le plus utilisé pour les interactions dans les pages Web.

L'affichage 3D seront gérées à l'aide de WebGL. La création et l'édition des objets 3D se fera à l'aide d'outils déjà très complets tel que Google SketchUp. *compléter un peu la partie 3D, dire ce qu'est WebGL tout ca peut être ?*

Dans un premier temps, des exemples classiques de mini-jeux seront présentés et analysés afin de mieux identifier les différences et points communs entre différents types de jeux. Cette analyse mettra en évidence des concepts classiques présents dans plusieurs jeux. Dans une seconde partie, les langages proposés pour la description des jeux seront exposés en mettant en évidence leurs possibilités et leurs limites. Enfin, les concepts récurrents vus dans l'analyse et présents dans les grammaires seront détaillés.

2 Les mini-jeux

Le contenu des mini-jeux est très varié : jeu de rôle, jeu de gestion, jeu de plateforme, etc. L'analyse des différences et des ressemblances entre ceux-ci, afin de définir ensuite une grammaire de description de jeux, est d'autant plus complexe.

Les difficultés d'implémentation des mini-jeux sont elles aussi différentes. 8 exemples concrets de jeux sont développés par la suite. Cela permet d'appuyer l'analyse de ce que contient un mini-jeu.

Ces jeux n'ont pas été choisi par hasard : chacun possède un intérêt dans la description de ses règles et son implémentation.

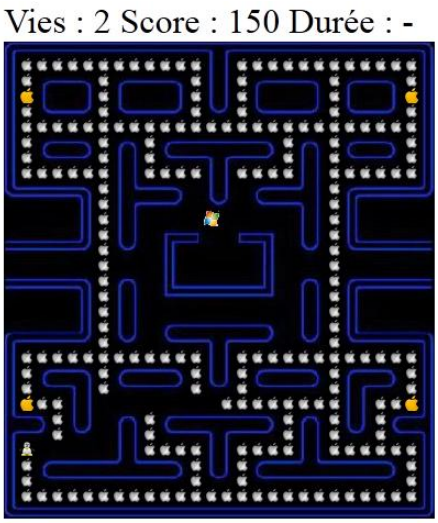
tableau à vérifier

Jeu	Intérêt majeur
Pacman	Système de bonus
1942	Tirs
Volley	Logique de score
Course	Intelligence artificielle
Mario	Niveaux
Game&Watch	Vague d'ennemis
Billard	Collisions
Gestion	Ressources

Tout d'abord, chaque jeu est présenté succintement, illustré par une capture d'écran d'une implémentation effectuée. Dans un second temps, une analyse de leur contenu sera effectuée, appuyée par un tableau comparatif et les diagrammes UML de ces jeux.

2.1 Présentation des mini-jeux

Pacman

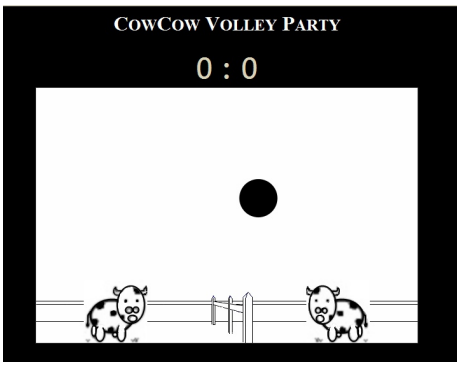


Ce jeu est une adaptation du jeu classique de Pacman. Le joueur contrôle le TUX via les touches du clavier. Son but est de manger toutes les pommes sur le terrain tout en évitant les Microsoft qui essayent de l'attraper. Pour arriver à cet objectif, le joueur dispose de 3 vies. Le TUX peut manger des pommes en or pour pouvoir détruire les Microsoft pendant une courte durée et marquer des points supplémentaires. Concernant l'affichage, le joueur voit le nombre de vies restantes, son score ainsi que le temps durant lequel les Microsoft restent vulnérables. Le jeu est adaptable car il est facile d'initialiser la carte à partir d'images et d'un fichier Json.

1942

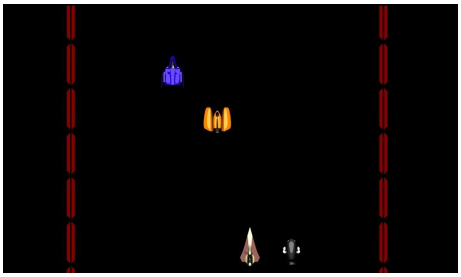
Dans ce shoot them up (jeu d'action où le joueur fait face à une multitude d'ennemis), le joueur contrôle un vaisseau armé de deux canons pour détruire tous les véhicules adverses et gagner ainsi des points. Le joueur possède 3 vies pour faire le maximum de points. Lorsque le joueur perd une vie, il devient invincible durant une courte durée pour reprendre la main. Le vaisseau est contrôlé via le clavier. En ce qui concerne les ennemis, ils suivent des déplacements prédéfinis qui peuvent être paramétrés. Le joueur n'est pas obligé de tuer tous les ennemis mais le but est quand même de faire le plus grand score.

Volley



CowCow volley party est un mini jeu humoristique mettant en scène deux vaches jouant au volley. On peut jouer à ce jeu en mode solo contre une IA ou à deux joueurs. Le but est de remporter 2 sets, sachant que remporter un set revient à marquer *xxx* points. La vache dispose de 3 coups différents : passe courte, passe longue et un smash. Les règles de ce jeu sont les mêmes que celles du volley classique. La vache est contrôlée au clavier aussi bien en mode multijoueur qu'en mode solo.

Course



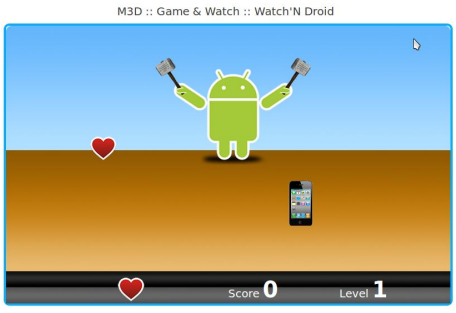
Ce jeu de course futuriste pour le web permet au joueur de se frotter à des intelligences artificielles pour faire le meilleur temps. Ce n'est pas un jeu de course classique : en effet des bonus se trouvent sur le circuit : un turbo, un bonus permettant d'augmenter le temps d'un des autres joueurs, un autre changeant la position d'un adversaire, etc. De plus, il existe différentes zones de circuit ayant des effets divers : inversion des commandes, changement de vitesses, etc. Le joueur peut choisir son véhicule parmi une sélection de 12 vaisseaux différents. Le contrôle se fait au clavier. Ce jeu est adaptable : en effet, il est possible de créer facilement des circuits ainsi que des nouveaux bonus. Il serait aussi possible de mettre en place un système de lecture de fichier pour configurer les paramètres des différentes courses.

Mario

Ce mini-jeu de plateforme reprend le principe des jeux du style de mario bros. Un personnage, contrôlé au clavier et représenté par un rectangle, peut se déplacer sur les côtés ou sauter. Il doit avancer au maximum, sans être touché par des ennemis, dessinés par des triangles, et sans tomber dans les trous du terrain. Le personnage peut faire perdre des points de vie aux ennemis jusqu'à les tuer en leur sautant dessus. S'il les touche sur les côtés, il meurt et la partie est terminée. La caméra avance lorsque le joueur avance suffisamment. Le joueur peut revenir à gauche jusqu'à la limite de l'écran. Le terrain est généré aléatoirement et est infini. Le but du jeu est d'obtenir le plus grand score. Le score diminue avec le temps et augmente lorsqu'un ennemi est tué ou que le personnage avance. Il est facilement possible de changer le type de fonctionnement du jeu pour passer dans un mode où le but est de passer d'un niveau à un autre avec des niveaux prédéfinis au préalable.

Capture à venir

Game & Watch



Watch'N'Droid est un mini-jeu du style Game & Watch. Le joueur doit chasser ses ennemis avant que ces derniers ne l'atteignent. Les ennemis apparaissent un par un en bas de l'écran et montent pour atteindre le joueur à une certaine vitesse. Cette dernière varie en fonction du niveau dans lequel le joueur se trouve. L'utilisateur est armé de deux marteaux pour détruire l'ennemi et donc marquer des points. Lorsque l'ennemi atteint le joueur se dernier perd une vie. Lorsqu'il ne lui reste que deux vies, des vies bonus apparaissent sur l'écran et le joueur peut les ramasser. Lorsque le nombre de vie arrive à zéro, le joueur a la possibilité d'enregistrer son score si ce dernier fait partie des cinq meilleurs scores qui sont actuellement enregistrés.

Billard

Ce mini-jeu de billard est un jeu multijoueurs au tour par tour. Il reprend les règles classiques du billard anglais (chaque joueur doit rentrer toutes les boules d'une couleur puis la noire). D'un point de vue gameplay, le joueur contrôle la queue qui pointe automatiquement vers la boule blanche via sa souris. Le joueur peut ainsi choisir l'angle avec lequel il compte frapper la boule blanche et en fonction du temps pendant lequel le joueur laisse le clic de la souris enfoncé, le tir est plus ou moins fort. Pour l'affichage, chaque joueur voit le nombre de boules rentrés ainsi que sa couleur. Une icône verte apparait au niveau du joueur qui doit jouer. Le cadre bleu en bas de l'écran affiche la personne ayant gagné la partie !

Gestion

Le jeu commissariat est un jeu de gestion du style Farmville. Le but est de maintenir et améliorer le commissariat en fonction des ressources disponibles. Les ressources sont au nombre de quatre : le nombre de policiers, l'argent, l'indice IGPN et l'alcool. Il y a trois actions disponibles pour le joueur. Il peut envoyer des policiers en mission dans un quartier choisi pour ramener de l'argent ainsi qu'un prisonnier. Si un prisonnier est ramené, le joueur a la possibilité de libérer le prisonnier contre de l'argent ou de le tabasser (avec un fort risque de pénalité). Il peut aussi acheter des équipements ainsi que de l'alcool avec son argent. L'alcool est une ressource qui diminue constamment, le joueur doit veiller à toujours en avoir pour ne pas perdre. Il peut aussi perdre avec un indice d'IGPN trop élevé, cet indice monte avec toutes les mauvaises actions (tabasser un prisonnier, etc.).

2.2 Comparaison des mini-jeux

Ces mini-jeux ont été développés et leurs diagrammes UML sont disponibles en annexe. De plus, ils permettent d'appuyer l'analyse de la description d'un mini-jeu. Le tableau suivant récapitule les différents objectifs des jeux et leur style de terrain.

Jeu	Niveaux	Fin de niveau	Victoire	Score	Terrain
Pacman	Oui	survie + tout ramassé	tous niveaux	Oui	Grille
1942	Oui	survie + ligne d'arrivée	tous niveaux	Oui	Progression verticale
Volley	Non	score	score	Oui	Plateau
Course	Oui	survie + ligne d'arrivée	tous niveaux	Oui	Ruban
Mario	Oui	survie + ligne d'arrivée	tous niveaux	Oui	Progression horizontale
Game&Watch	Oui	tout tué	tous niveaux	Oui	Plateau et grille
Billard	Non	toute boule rentrée	score	Oui	Plateau
Gestion	Non			Oui	Grille

Le tableau suivant permet quant à lui de comparer les aspects concernant le contrôle et les personnages.

Jeu	Personnage	Contrôle	Autres actions	PNJ [1]
Pacman	Oui	direct via clavier		Oui
1942	Oui	direct via clavier	Tir	Oui
Volley	Oui	direct via clavier	Tir, Saut	Oui
Course	Oui	direct via clavier	Utilisation bonus	Oui
Mario	Oui	direct via clavier	Saut	Oui
Game&Watch	Oui	direct via clavier	Frappe	Oui
Billard	Oui (si queue=joueur)	direct via clavier	Force	Oui
Gestion	Oui mais inutile	clic, sélection et action	Clic pour actions	Non

Ces tableaux permettent de mieux identifier les différences et points communs entre les différents jeux.

Par exemple, les objectifs au cours d'un niveau (ou pour un jeu sans niveau) se limitent souvent à atteindre une certaine zone appelée ligne d'arrivée, ou éventuellement à remplir des conditions de temps ou de ressources. En effet, à la fois la vie et le score peuvent être vus comme une ressource : il s'agit d'un état, et une condition sur celui-ci est nécessaire à la victoire. En faisant des conjonctions et des disjonctions de ces différentes possiblités, il est possible de définir les conditions de victoire pour tous les jeux (sauf le jeu de gestion).

En revanche, si on regarde comment le monde est construit, il est très différent d'un jeu à l'autre. Pour certains comme Pacman, il est défini par une grille, pour d'autres comme le jeu de course, il est défini via un ruban.

Le tableau permet de distinguer clairement le jeu de gestion des autres jeux. La grammaire proposée par la suite ne prendra pas en compte ce type de jeux. La comparaison se base donc sur les autres jeux, pour définir des concepts qui seront détaillés dans les parties suivantes. Retirer du langage de description les jeux de gestion n'empêchent pas de couvrir un nombre important de jeux. Il serait possible par exemple de définir plusieurs grammaires, chacune couvrant une catégorie de jeux afin de pouvoir créer plusieurs types de jeux. Ainsi une autre grammaire pourrait spécialement être créee pour les jeux de gestion. De la même façon, il est difficile de trouver des points communs entre les 7 mini-jeux restants et un jeu de rôle classique où le joueur doit réaliser plusieurs quêtes.

2.3 Analyse des mini-jeux

On remarque tout d'abord que tous les mini-jeux possèdent une boucle de rafraîchissement. Cette dernière permet à la fois d'effectuer les différentes actions telles que le déplacement des personnages non joueurs et de gérer les différents 'timer' du jeu.

Ensuite, dans les mini-jeux se retrouvent souvent la notion d'un avatar, petit personnage que le joueur peut déplacer. Différentes actions peuvent se greffer à cet avatar selon les jeux, souvent le déplacement, le saut, le tir. Cet avatar a des caractéristiques que l'on retrouve dans plusieurs jeux comme la vie, d'autres plus rares comme une quantité de points de magie. De même, dans de nombreux jeux se trouvent la notion d'ennemis : des personnages non joués, dont le comportement est prédéfini.

De plus, la notion de ressources est très utile. Il s'agit d'une variable qui définit un état par exemple sur les entiers tel que le score. Celle-ci peut être modifiée selon les évènements qu'il se passe dans le jeu.

Enfin, dans tous les mini-jeux, le concept de collision est très utile. Celle-ci cause la plupart du temps de nombreuses conséquences (mort d'ennemis, de joueurs, incapacité de passer un obstacle, etc.)

3 Grammaires

Trouver une grammaire riche, complète et accessible à n'importe quel utilisateur est difficile. C'est pourquoi, en réalité deux grammaires sont présentées.

- La première est dite de 'haut-niveau'. Elle permet de décrire la majorité du contenu du jeu dans un langage simple de compréhension.
- La seconde est dite de 'bas-niveau'. Elle permet de manipuler chaque attribut et est beaucoup plus proche de l'implémentation finale que la première grammaire. C'est par exemple elle qui

permettra de définir le comportement de l'intelligence artificielle, chose très difficile à mettre en oeurvre à haut-niveau.

Le schéma de compilation se complexifie alors : un fichier décrivant le jeu dans la grammaire de haut-niveau est compilé afin de donner un fichier respectant la grammaire de bas-niveau. A ce niveau là, l'utilisateur peut effectuer de nouveaux ajouts ou modifications. Le second compilateur produit alors le script final du jeu en javascript.

3.1 Grammaire haut-niveau

La description du jeu se déroule en plusieurs phases. Elle commence par les entités du jeu : personnage, ennemis, alliés et leurs attributs. Viennent ensuite les définitions d'actions de base et des commandes. Enfin, les règles du jeu sont spécifiées.

3.1.1 Définition des entités du jeu et de leurs attributs

Il s'agit de lister les différents personnages ou objets qui apparaissent dans le jeu. Pour faciliter le travail de l'utilisateur, diverses classes sont déjà créées pour définir les entités. La classe de base est la classe Object avec des attributs de base comme la position, l'orientation, la taille, etc. De celle-ci héritent de nombreuses autres classes comme Character, Vehicle ou Weapon chacun ayant des nouveaux attributs spécifiques. *Inclure le schéma des classes ?*

La déclaration d'un nouveau personnage se fait alors via un identificateur et le mot-clef 'is'. Par exemple `Mario is Character` permet de déclarer Mario comme un personnage de classe Character, et peut donc posséder tous les attributs de cette classe. Ces derniers sont initialisés à des valeurs par défaut. Pour modifier cette initialisation, par exemple pour changer l'attribut `lifeMax` de Mario, les mots clefs 'has' et 'at' sont utilisés : `Mario has lifeMax at 3`. De même, il est nécessaire de pouvoir rajouter de nouveaux attributs à Mario. La syntaxe est la même. Ainsi, `Mario has energie at 8` ajoute un attribut énergie à Mario, initialisé à 8. Les nouveaux attributs doivent obligatoirement être initialisés grâce à 'at'.

De plus, s'il existe divers personnages qui possèdent aussi cette énergie, il est souhaitable de ne pas avoir à répéter cette ligne. Ainsi, de nouveaux types peuvent être crées grâce au mot-clef 'type' : `type Plombier is Character. Plombier has energie at 8. Mario is Plombier. Luigi is Plombier.` *a-t'on l'héritage multiple et si oui, comment ce la se passe en cas d'attributs du même nom ?*

Ensuite, en même temps que la déclaration de la classe d'une nouvelle identité, il est possible de la définir comme jouée par le joueur, ennemi, allié ou neutre. (Si cela n'est pas précisé, la valeur est mise à neutre par défaut. `Mario is Plombier player. Luigi is Plombier ally. type Boss is Character ennemy. Bowser is Boss.` De plus, dans le cas où ce n'est pas un personnage joué, il peut être déclaré comme multiple. Cela signifie qu'il est possible d'en générer plusieurs avec la seule commande 'generate'. Par exemple `Koopa is Character enemy multiple. generate 5 Koopa in zone`. Permet de créer 5 Koopa dans `zone` qui aura dû être définie auparavant. Outre 'in', les mots-clefs 'on' permet de placer des objets juste au-dessus de la zone définie par le mot suivant, et le mot-clef 'at' génère le premier objet à l'endroit indiqué et les autres proches de celui-ci, tout en évitant les collisions. *A confirmer que ce soit ça pour multiple*

Enfin, il est possible de manipuler des listes de ces classes.

```
type humain is Character.
type elfe is Character.
type nain is Character.
type hobbit is Character.
type magicien is Character.
gandalf is magicien ally.
CommunauteDeLAnneau is list of 1 elfe with 1 nain with 2 humain with gandalf with 4 hobbit.}
```

3.1.2 Autres classes prédéfinies

D'autres classes sont prédéfinies : une classe Game concernant l'ensemble général du jeu, une classe Camera contenant les informations sur les caméras : certains comportements classiques sont prédéfinies comme un suivi à la première ou troisième personne, ou une caméra libre.

De plus, 2 clases permettent de gérer respectivement les compteurs et les ressources de type temporels.

3.1.3 Définition de nouvelles actions et assignation des commandes

Il est ensuite possible de définir de nouvelles actions via le mot-clef 'definition' et 'means'. Par exemple, en reprenant l'exemple de Mario : `Mario is Character player. definition suicide means mario dies.`

Les commandes sont ensuite définies après le mot-clef 'command'. Elles sont générées soit par l'appui d'une touche X du clavier ('key X') soit par une action Y sur la souris ('mouse Y'). `command mouse rClick for suicide` permet par exemple de causer l'action nommée suicide lorsque le joueur clique sur le bouton droit de la souris.

De plus, de nombreuses actions sont prédéfinies pour les personnes du jeu comme jump ou move left ou concernant le jeu en général comme pause, save ou gameover. `command mario is key space for jump, key Z for move forward, key Y for move backward, key S for move left, key D for move right.`

Il est possible de désactiver (respectivement d'activer) certaines touches du clavier ou action de la souris au cours du jeu. Pour cela, il faut utiliser le mot-clef 'desactive' (respectivement 'activate'). `desactive key Z` désactive les actions lors de l'appui sur la touche Z. Il est également possible de désactiver toutes les commandes `desactive commands` ou toutes les commandes claviers `desactive key`.

3.1.4 Déclaration des règles du jeu

Les règles du jeu sont définies par le mot-clef 'rule' et 'then'. Par exemple `rule mario dies then gameover.`

Il est également possible de manipuler les attributs des différentes entités. Prenons le cas où un ennemi a été défini `bowser is Character ennemy` et que si mario entre en collision avec lui , alors il perd un point de vie, réduit son score de 100 et provoque un saut. `rule mario touches bowser then sub 1 for life of mario, sub 100 for score of game, mario jump.`

Outre 'sub' d'autres mots clefs existent pour les autres opérations arithmétiques élémentaires, ainsi qu'un 'assign' qui change directement la valeur de l'attribut.

Certains concepts sont déjà définies : ainsi 'touches' est causée par une entrée en collision entre les deux entités, dies signifient l'arrivée de l'attribut life à 0, kills se produit lorsque le premier personnage tue le second. *j'ai un peu de mal à voir tout comment marche les declencheur et les conséquences dans la grammaire, du coup, je veux bien que quelqu'un nous explique tout ca, il doit manquer pas mal de choses là encore.*

3.1.5 Exemple de mini-jeu décrit dans la grammaire 'haut-niveau'

Nous pensions éventuellement à un simple jeu à caméra fixe avec terrain prédéfini ou le personnage doit éviter certains ennemis et prendre toutes les pièces avant d'atteindre un point d'arrivée, en fait genre un pacman sans bonus quoi :p, parce que mario on a le problème des collisions par le haut ou pas, ainsi que la génération du terrain

3.2 Grammaire bas-niveau

La grammaire bas-niveau est beaucoup plus proche de l'implémentation finale que la haut-niveau. Ainsi, toutes les ressources doivent avoir un identifiant unique, les évènements qu'ils proviennent du joueur (clavier, souris) ou de la moficiation de l'état d'un personnage sont gérés par des signaux. La description d'un jeu est constitué de ressources, d'entités, de caméra, d'une boucle de rafraîchissement, d'un gestionnaire d'évènements. Un moteur physique permettant de tester les collisions et contenant également les forces générales du jeu (gravitation, vent, etc.) est également disponible.

3.2.1 Définition des ressources

Une ressource est une variable du jeu qui pourra ensuite être attribuée à une entité. Elle est identifiée par un nom unique.

`marioLife 1` permet de créer une ressource marioLife initialisée à 1.

Il existe un type spécial de ressource : les ressources temporelles. Celles-ci sont mises à jour via la boucle de rafraîchissement générale du jeu. Pour la définir, cette fois 2 valeurs sont données suites à la ressource : le pas du timer, et sont initialisation : `timeStarBonus 10000 0` permet de créer une ressource de nom 'timerStarBonus' (pouvant correspondre à la durée du bonus donné par une étoile) avec un pas de 10 secondes, et initialisée à 0 milisecondes.

Il est également possible de faire des énumérations de ressources. *il faudrait un joli exemple*

3.2.2 Définition des entités et caméras

On considère que le terrain est représenté par exemple par une matrice de points dans l'espace surlaquelle est aplliquée une texture. La matrice pourra être remplacée selon le type de terrain pour un certain jeu. *Ca veut un peut rien dire ça, non ?*

L'ensemble des entités d'un jeu est alors constitué d'un terrain et d'un ensemble d'objet. Chaque objet est identifié par un nom, ainsi que par deux fichiers créés par des outils de dessin d'objets 3D, un fichier .obj contenant l'ensemble des points de l'objet, et un ficier .mat contenant la texture appliquée sur les faces de l'objet. De plus chaque objet contient un ensemble de paramètre comme la vitesse, la position, etc. *vérifier le format de sortie de Sketchup*

Les caméras sont définies de manière similaire aux entités : chaque caméra est définie par un nom, une position et une orientation.

comment on associe des ressources à une entité ?

3.2.3 Boucle de rafraîchissement et gestionnaire d'évènements

Les divers évènements du jeu sont gérés via un système de signaux.

Le jeu dispose d'une boucle de rafraîchissement principale. Celle-ci génère un signal signalUpdate-Counter émettant un signal régulièrement. De même, il permet de vérifier les touches enfoncées et actions de la souris à chaque tic du timer.

Enfin, chaque mise à jour de ressource nommée X émet un signal updateX.

Les actions lors de la réception d'un signal sont gérées par un gestionnaire d'évènements. Ce dernier est défini par un ensemble de signaux et d'instructions. Les instructions sont données comme dans un langage classique. Elles sont composées des mises à jours de variable en utilisant les opérateurs arithmétiques classiques, les constantes, les autres ressources et des nombres aléatoires. Des instructions conditionnelles sont également disponibles, les booléens étant générés via des comparaisons d'expressions.

Les instructions peuvent également reprendre des concepts classiques de mini-jeux : pause, nouvelle partie, fin de partie, sauvegarde de la partie. *insérer un petit exemple*

3.2.4 Exemple de mini-jeu dans la grammaire bas-niveau

to do ...

4 Ontologie

Partie sur la définition des concepts

5 Stratégie de Développement

Partie sur la stratégie de développement à écrire ici