3DWIGS

3D Web Interactive Game Studio

Rapport Post Mortem

Equipe Mini3D : Berlon Antoine, Bouzillard Jérôme, Chegham Wassim,
Clergeau Thomas, Faghihi Afshin, Guichaoua Mathieu, Israël Quentin,
Kien Emeric, Le Corronc Thibault, Le Galludec Benjamin,
Le Normand Erik, Lubecki Aurélien, Marginier David, Sanvoisin Aurélien,
Tolba Mohamed Amine, Weinzaepfel Philippe et Zadith Ludovic

Responsable Projet : Docteur Rémi Cozot

Année de réalisation : Année Universitaire 2010-2011

Ce document a pour objectif de présenter la phase de développement du projet de Master 1 Informatique intitulé « Plateforme de création de mini-jeux 3D sur le WEB ». Ce projet a pour but le développement d'un outil auteur de création de contenus 3D interactifs au sein de pages Web.

Ce rapport fait suite au document « Plateforme de création de mini-jeux 3D sur le WEB – Analyse » réalisé pour l'Unité Enseignement GPL, Gestion de Projet Logiciel, du premier semestre de Master 1. Il a pour objectif d'expliquer les différentes démarches suivies concernant le développement du logiciel et d'analyser l'efficacité des choix qui ont été pris pour la gestion du projet, ceci dans le but d'obtenir le résultat final qui sera présenté durant les soutenances du Lundi 9 Mai et la démonstration du Mardi 10 Mai 2011.

Le rapport est divisé en cinq parties, la première est un rappel de l'analyse du sujet réalisée lors du premier semestre. La seconde partie expose le choix de stratégie de développement finalement suivi par le groupe Mini3D. Dans un troisième temps, sont faites une analyse des difficultés rencontrées lors du second semestre et une analyse de la façon dont le groupe aurait pu les éviter. Puis dans une quatrième partie sont explicitées les réussites lors de la phase de développement. Enfin la partie 5 présente le résultat obtenu à la fin du projet de Master1.

Table des matières

I. RÉSUMÉ DE LA PHASE D'ANALYSE

Suite à la phase de réflexion et d'analyse du 1er semestre, sont ressortis deux langages de description et une stratégie de développement.

Le premier langage est un langage de description dit de « haut niveau ». Il avait été réalisé dans le but d'être le plus simple possible afin qu'il soit accessible à un large publique et non uniquement à des initiés en programmation. Il avait en effet pour objectif de permettre la description d'une majorité de type de jeu avec simplicité.

Le second langage de description, dit de « bas niveau », avait pour vocation d'être plus proche de l'implémentation finale. Celui-ci devait ressembler, après compilation, au code JavaScript interprété par un navigateur Web supportant le HTML5. Ce langage de description s'avérait être d'une assez grande complexité, n'ayant pas pour but d'être utilisé par des utilisateurs non programmeurs mais par des experts en informatique. En effet, les plus avertis pouvaient ainsi avoir une encore plus grande liberté dans la réalisation de leur contenu vidéo ludique.

Le schéma de compilation était donc le suivant: un fichier décrivant le jeu dans le langage de « haut-niveau » devait être compilé afin de donner un fichier respectant le langage de « bas-niveau ». A cette étape, l'utilisateur devait pouvoir effectuer de nouveaux ajouts ou modifications. Le second compilateur devait produire alors le script final du jeu en JavaScript, interprétable par un browser supportant le HMTL5.

En plus de ces deux langages de description de mini-jeux, le groupe Mini3D disposait d'une stratégie de développement précise. Celle-ci prévoyait les différentes tâches que l'ensemble du groupe devait réaliser, en commun ou en sous-groupe, durant le second semestre afin de pouvoir développer le logiciel : baptisé 3DWIGS pour 3D Web Interactive Game Studio.

Ci-dessous, le diagramme de Gantt qui représente la stratégie de développement initialement prévue :

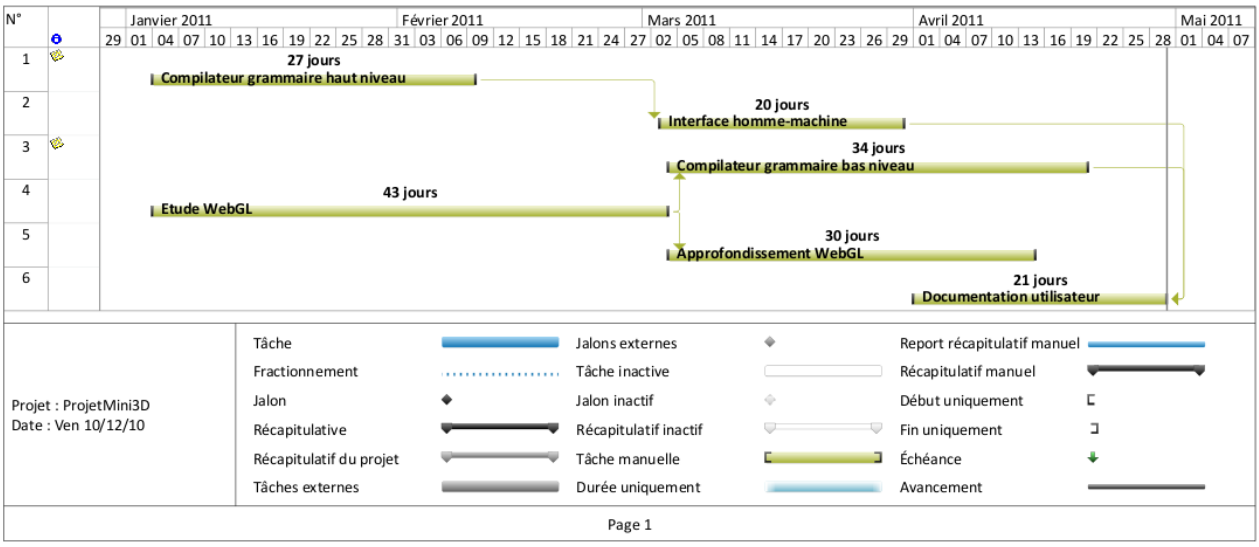


Illustration 1: Diagramme de Gantt du planning prévisionnel pour le 2nd semestre

II. UNE NOUVELLE STRATÉGIE DE DÉVELOPPEMENT

Au début de la phase de développement, une autre stratégie de développement a été suivie, principalement en suivant les conseils du responsable de projet.

Cette nouvelle stratégie, du type « Scrum », a suivi un cycle itératif, en effet les résultats des sous-groupes étaient mis en commun chaque couple de semaines. Elle fut préférée à celle soumise au 1er semestre car elle a permis une meilleure répartition de l'ensemble du groupe et s'est avérée plus pertinente que la stratégie pensée de prime abord qui ne présentait des résultats qu'à la fin du semestre.

Cette nouvelle stratégie de développement a consisté à réaliser un mini-jeu d'une grande simplicité évoluant en complexité au fil des itérations. Ce mini-jeu porte le nom de « jeu du lapin ». Pour la première itération, le « jeu du lapin » devait mettre en scène deux personnages. L'un d'entre eux est une entité jouable, le chasseur, qui dispose de la capacité de tirer avec une arme et de se mouvoir sur un axe de déplacement. La seconde entité est quant à elle non-jouable et a pour représentation un lapin. Les règles et but du jeu sont des plus simples pour correspondre au degré de complexité lié aux premières itérations. En effet pour obtenir la victoire, il suffit au chasseur de toucher le lapin une fois avant que celui-ci n'ait le temps d'effectuer un trajet prédéfini sinon il y a défaite. Le chasseur suit la contrainte suivante, il ne peut lancer qu'un projectile à la fois et doit attendre la disparition de celui-ci (sortie d'écran ou contact avec obstacle) afin de pouvoir tirer à nouveau.

Pour répondre à ce nouveau choix de développement, le groupe Mini3D s'est séparé en plusieurs sous-groupes allant de 2 à 5 personnes avec pour chacun des objectifs précis. Le groupe Mini3D disposant d'un effectif plutôt conséquent, chaque sous-groupe s'est vu attribué un responsable, ceci facilitant le lien entre les sous-groupes et la mise en commun des résultats à chaque itération.

Le groupe de projet s'est séparé de la manière suivante :

- ✔ Groupe Contenu (2 personnes) avec pour mission l'adaptation des mini-jeux 2D développés au premier semestre au langage « haut niveau » et la réalisation des éléments 3D de ces mini-jeux.
- ✔ Groupe IHM (4 personnes) ayant pour objectif la réalisation d'une interface homme machine pour le logiciel 3DWIGS.
- ✔ Groupe Compilation (4 personnes) avec pour but la création des deux compilateurs pour les langages de description réalisés durant le premier semestre.
- ✔ Groupe Moteur 3D (5 personnes) avec pour mission la conception d'un moteur physique 3D offrant un ensemble de fonctions pour la réalisation concrète des mini-jeux.
- ✔ Groupe Documentation (2 personnes) avec comme dessein l'écriture d'une documentation complète au fur et à mesure de l'avancement du projet.

En parallèle, plusieurs personnes se sont occupées de l'intégration des différents modules pour la finalisation du logiciel.

Le diagramme de Gantt qui correspond à ce nouveau choix de développement se trouve en annexe de ce rapport.

III. FAIBLESSES, DIFFICULTÉS, CHOIX PRIS DURANT LE DÉVELOPPEMENT DE 3DWIGS

Le choix de cette nouvelle stratégie de développement possède de nombreux points forts du fait qu'elle soit basée sur un système de cycle itératif. Cependant, la complexité du projet en lui-même a posé au groupe un souci temporel qui fut difficile à surmonter.

1. Le cycle itératif

Dans la partie précédente, nous avons présenté la nouvelle stratégie de développement que le groupe a choisie pour la réalisation du projet. Les itérations successives devaient suivre la conception du mini-jeu évolutif « jeu du lapin » en l'enrichissant de fonctionnalités au fil des itérations.

Cependant, le choix de suivre une telle stratégie imposait au groupe Compilation de réaliser les deux compilateurs en un temps très limité. Ce qui s'est avéré être une tâche très difficile au vu de la complexité de ceux-ci et du temps imparti pour la première itération, i.e. deux semaines.

Certes, les deux analyseurs ne devaient pas forcément être complets à la fin de cette itération mais devaient permettre d'obtenir le « jeu du lapin » dans sa première version (itération 1) sans aucune erreur durant les phases de compilation. De ce fait, cette tâche s'est avérée très longue et le groupe n'a pas pu respecter les délais qu'imposait la nouvelle stratégie de développement, à savoir une itération toute les deux semaines.

Pour résoudre ce problème de temps, il a été décidé de s'affranchir de l'itération 1 de deux semaines en la remplaçant par une itération plus longue d'un mois afin d'instaurer les bases du compilateur pour la suite du cycle de programmation.

Un autre des problèmes survenu lors du développement de cette première itération est encore venu du compilateur. En effet le groupe Compilation s'est aperçu durant la réalisation du compilateur du langage dit de « bas niveau », que celui-ci était très proche du travail réalisé par le groupe Moteur 3D. En effet, le code JavaScript sorti après compilation de ce second langage de description était sensiblement le même que celui offert par le moteur 3D. De ce fait, il fut décidé par l'ensemble du groupe de projet de s'affranchir de ce langage de description.

La suppression de cette partie a eu pour conséquence d'enlever le second niveau de compilation initialement prévu dans le schéma de fonctionnement du logiciel. Pour rappel, le langage de description « bas niveau » avait pour objectif d'offrir une plus grande liberté aux informaticiens expérimentés en leur permettant d'accéder directement à un niveau de programmation plus bas.

En aucun cas, ce programmeur ne perd la liberté que lui conférait le langage dit de « bas niveau » car cet utilisateur peut directement utiliser les fonctionnalités que lui offre le moteur physique de 3DWIGS pour développer le contenu vidéo-ludique qu'il désire.

En plus de la suppression du langage de description « bas niveau », d'autres modifications ont dû être apportées au langage dit de « haut niveau ». En effet, celui-ci s'est avéré être incomplet sur de nombreux points entrainant des délais de réflexion supplémentaires.

En effet, le système d'appartenance à une équipe alliée ou ennemie était trop simpliste et il était impossible de créer des mini-jeux avec plusieurs équipes. Une entité pouvait juste avoir l'appellation 'player', 'ally', 'enemy' ou 'neutral'.

Maintenant, il est possible de créer autant d'équipes et de joueurs que l'on veut. Les joueurs sont, soit définis comme humain, soit définis en tant qu'IA. On peut aussi déclarer des joueurs sans équipe 'Player solo' ou placer des équipes qui seront sous la bannière d'une autre. Ainsi, les joueurs sont bien différenciés et on peut leur assigner des entités à contrôler.

La gestion des appartenances entre objets avait été laissée en suspend. Mais part la suite, les mots-clé 'grasps', 'ingests' et 'expels' ont été ajoutés au langage.

L'IA, elle aussi mise en suspend, reprenait exactement les mêmes règles de grammaire. Les doublons ont été supprimés dans le langage et la partie a été modifiée en une suite de règles du jeu.

Enfin, le contenu des commandes et des règles du jeu ont été placés dans les définitions car ils étaient les mêmes. Cela a simplifié le travail de l'équipe Compilation.

2. La réorganisation des groupes

Suite au retard qui s'est accumulé durant les premières semaines de développement, une réorganisation de l'ensemble du groupe de projet fut décidée afin de combler ces délais. En effet, la partie Compilation n'avait pas été évaluée à sa juste mesure, ce qui a eu pour effet de retarder la phase d'intégration des différentes parties du logiciel. Cette réorganisation s'est passée de la façon suivante, les trois groupes les plus avancés (Contenu, IHM et Moteur3D) ne pouvant plus évoluer sans un compilateur au point se sont joints au développement de ce dernier. Cette réorganisation a rapidement permis de faire évoluer le compilateur du langage de description afin de pouvoir assister à la première intégration complète des modules du logiciel.

3. Comment auraient pu être évités ces problèmes ?

Dans les deux parties précédemment développées, il a été constaté que les différents problèmes étaient survenus autour du cœur du logiciel : le compilateur.

L'ensemble du groupe Mini 3D avait mal évalué le travail à fournir pour chaque partie du logiciel. Effectivement, le travail à produire pour la réalisation du compilateur a été sous-estimé par rapport à celui pour l'apprentissage de la technologie WebGL. En effet l'apprentissage de cette technologie inconnue des étudiants et très peu documentée a été surestimée, c'est pourquoi cette tâche a tout d'abord mobilisé la majorité des membres, laissant légèrement de côté la partie compilation. Il aurait été plus sage de mobiliser davantage de personnes à la réalisation du compilateur, partie essentielle au projet.

IV. LES RÉUSSITES DU PROJET MINI 3D

Dans la partie précédente, nous avons présenté certaines étapes du développement durant lesquelles des erreurs ont été commises par le groupe. L'étude de l'historique de conception du logiciel a montré que les problèmes rencontrés lors de la phase de réalisation de celui-ci provenaient principalement d'une mauvaise analyse des priorités et donc d'une mauvaise affectation des effectifs dans les sous-groupes. Après un examen des événements, le constat fut rapide à établir, le groupe Compilation aurait dû réunir plus de membre dès le début afin d'assurer le bon développement du projet Mini3D. Cependant et heureusement, la phase de développement du projet a aussi connu des réussites sur de nombreux points. En effet, des étapes dans la réalisation du logiciel se sont soldées par de réels succès.

1. Matérialisation d'un concept abstrait avec partage des tâches

En cette fin de semestre, le groupe a conscience que le projet Mini3D n'a pas atteint les objectifs fixés par l'ensemble des personnes qui a participé au développement de ce logiciel (principalement vis à vis du client). Cependant, il est considéré que le travail qui a été effectué durant ce semestre fait parti des réussites du projet. En effet, le groupe n'est parti de rien pour arriver à ce résultat (pas de reprise de code ou de modèle) et a dû découvrir un nouveau domaine auquel la plupart des membres n'avait jamais fait face : le développement de contenu graphique.

Le groupe a su déterminer et cibler les principales exigences que le projet requerrait et réaliser un partage des tâches adapté (sauf pour la partie compilation) afin de répondre à ces exigences. De plus, avoir pu mettre en place les bases du développement de ce logiciel fut une étape très enrichissante pour les membres du projet. Que ce soit dans la prise de décision ou des choix de développement en passant par la création d'une architecture logicielle, le projet a montré les difficultés que cela représentait, et a ainsi apporté une bonne expérience pour de futurs projets à l'ensemble des personnes de Mini3D.

2. Apprentissage de nouvelles technologies et développement en WebGL

Dans le planning prévisionnel, l'apprentissage du WebGL constituait une étape critique pour le bon développement du projet du fait que cette technologie était une inconnue totale pour l'ensemble du groupe. En effet, dans la nouvelle stratégie de développement, deux des sous-groupes créés devaient avoir affaire à cette nouvelle technologie : les groupes IHM et Moteur 3D. De ce fait, durant la répartition des groupes, ces derniers furent avantagés en se voyant attribuer un effectif conséquent afin de prévenir d'éventuelles difficultés ; ceci dans le but d'éviter de se retrouver bloqué sur une nouvelle technologie complexe et très peu documentée. Or ces difficultés ne se sont pas réellement fait sentir, l'apprentissage du WebGL ayant été facilité par l'utilisation d'une librairie assez complète: GLGE. Du fait, le choix d'utiliser cette librairie a permis de gagner beaucoup de temps malgré un manque de documentation détaillée et une fiabilité imparfaite due au fait que WebGL et la librairie GLGE sont toujours en cours de développement. Le but de GLGE est de masquer l'utilisation directe du WebGL au développeur Web, qui peut alors se concentrer sur la création d'un contenu plus riche pour le web.

Malgré l'utilisation de cette librairie, les groupes ont dû se familiariser avec WebGL du fait de l'existence de nombreuses erreurs dans le code de GLGE. En effet il était important de pouvoir différencier la provenance des incohérences rencontrées. Il fallait savoir si celles-ci avaient pour origine le code développé par les équipes du groupe Mini3D ou si elles venaient de l'implémentation de la librairie auquel cas des patchs de corrections furent apportés.

Outre WebGL, le groupe a fait face à d'autres nouvelles technologies telles que HTML5 et des deux API LocalStorage et IndexedDB.

3. Réalisation de contenu 3D

Le groupe Contenu, malgré un effectif minimal a réussi les différentes tâches qu'il avait à accomplir en réalisant l'ensemble des traductions des mini-jeux 2D qui fut réalisé durant la phase d'analyse du 1er semestre. De plus le groupe Contenu a réalisé l'ensemble des modèles 3D des personnages, éléments de décors et cartes (cartes simples sans relief) des ces mini-jeux. Malgré des connaissances assez faibles en création de contenu 3D, des objets plutôt réalistes sont sortis de l'imagination de ce sous-groupe !

V. RÉSULTAT

Maintenant, nous allons détailler l'état d'avancement du projet qui n'est pas encore finalisé. Au vu de l'enchaînement des difficultés, c'est une grande déception pour le groupe de ne pas avoir un programme fonctionnel et répondant aux attentes de chacun des membres de l'équipe. En effet, le logiciel 3DWIGS n'a pas atteint le niveau de développement espéré. Mais avoir un logiciel terminé reste un objectif à atteindre. Néanmoins le fait de penser à l'éventualité d'une reprise du logiciel 3DWIGS permet à l'effectif de continuer le travail afin de voir une version finale et opérationnelle du logiciel répondant à l'ensemble des attentes du responsable de projet, le Dr. Rémi Cozot.

1. Description de l'état de l'interface homme machine de 3DWIGS

L'interface utilisateur développée pour 3DWIGS a été construite dans le but d'offrir un moyen rapide et simple pour réaliser ses mini-jeux. L'usager a la possibilité d'importer ses propres modèles Collada dans l'éditeur, ou d'en sélectionner parmi ceux proposés par 3DWIGS.

Une fois le modèle importé, l'utilisateur pourra éditer en temps réel les propriétés de celui-ci :

- ✔ sa position,
- ✔ son orientation,
- ✔ sa taille.

De plus, l'interface de 3DWIGS dispose d'un éditeur de code offrant ainsi à l'utilisateur la possibilité de rédiger directement la description de ses mini-jeux. Cet éditeur dispose d'une coloration syntaxique du langage 3DWIGS ainsi que d'une fonctionnalité d'auto-complétion pour une partie des règles du langage. L'éditeur est également capable de détecter tout changement dans la zone d'affichage afin de pouvoir effectuer une mise à jour de son contenu.

L'interface possède aussi un système de sauvegarde afin de permettre à l'utilisateur de conserver ses réalisations. Elle a été développée en exploitant deux fonctionnalités offertes par HTML5 qui sont les API LocalStorage et IndexedDB permettant la sauvegarde du contenu de l'usager sur sa machine personnelle.

2. Description de l'état du Moteur3D de 3DWIGS

Le moteur physique de 3DWIGS utilise principalement la librairie GLGE dans sa version 0.8 et permet le chargement et l'animation d'éléments 3D importés par fichier Collada.

En effet, le moteur permet la génération d'un environnement 3D, vierge au départ, qui est appelé scène par la suite, où il est possible de charger un ensemble d'objets 3D afin de pouvoir les faire évoluer dans cet univers virtuel. Les différents éléments constituant la scène répondent à une organisation de graphe de scène qui permet une gestion optimale pour l'application des déplacements, la détection de collisions, l'affectation de nouveaux éléments et d'autres fonctionnalités.

Dans cet environnement 3D, l'utilisateur peut faire évoluer un seul ou un ensemble d'objets via l'utilisation du clavier et suivre les déplacements de celui-ci en lui affectant une caméra ou en déplaçant celle qui a été créée en même temps que la scène.

L'ensemble des objets de la scène peut aussi être soumis à des forces comme la gravité. En effet, le moteur de 3DWIGS peut simuler l'application de forces distinctes sur un objet ou un ensemble d'objets via l'utilisation de fonctions implémentées par l'équipe de gestion du Moteur 3D. Cependant, cette interaction n'est pas la seule proposée par le moteur physique. Celui-ci peut aussi gérer la détection des collisions entre différents éléments de la scène et réaliser des actions en fonction des règles de l'utilisateur. Le système de gestion de collisions du logiciel est basé sur l'utilisation de volume englobant (BoundingBox) afin de rendre les collisions entre les différents objets peu gourmandes en temps de calcul.

3. Description de l'état du Compilateur de 3DWIGS

Durant le dernier mois de développement, le groupe Mini3D a dû changer à nouveau son organisation afin de rattraper le retard dans la conception du compilateur du logiciel. Malheureusement, malgré une avancée rapide, tous les objectifs n'ont pas pu être atteints. A l'heure de la rédaction de ce rapport, la correction et l'apport de nouveau contenu est toujours en cours.

Cependant certaines parties du compilateur ont été réalisées, en effet, la génération des « entités » et des « types » a pu être faite selon la stratégie de développement.

Certaines actions apparaissant dans les « définitions » du langage de description ont pu être gérées par le groupe Compilation.

Concernant les règles du jeu, la gestion de cette partie reste en cours de réalisation mais le travail est encore important. De plus, les commandes au clavier sont gérées, pas encore celles de la souris. Pour l'instant et surtout par manque de temps, l'intelligence artificielle n'a pas pu être correctement implémentée. Elle est actuellement gérée en utilisant un enchaînement de règles afin de simuler un comportement « humain ».

VI. CONCLUSION

L'étude de l'historique de développement du projet Mini3D a mis en évidence l'importance de la stratégie de développement réalisée durant le 1er semestre de Master 1 et les conséquences vis-à-vis de la qualité de celui-ci. En conséquence, l'élaboration trop peu approfondie de cette partie a eu pour répercussion une perte de temps considérable, ce qui a engendré la remise en cause de l'intégralité du projet. Le travail fourni pour la réalisation des correctifs d'erreurs et d'oublis de la phase d'analyse, a également engendré un gaspillage de force du fait qu'une partie du travail qui fut réalisée s'est avérée inutile parce que celui-ci ne correspondait plus aux demandes initiales. A contrario, l'équipe de projet a connu de réels succès tels que l'apprentissage de nouvelles technologies et la satisfaction d'arriver ensemble à un résultat concret à partir d'une idée abstraite. De plus l'ensemble de l'effectif de l'équipe Mini3D a ainsi connu une expérience dans le développement en groupe d'un vaste projet sur une année universitaire complète. Nous pouvons conclure qu'une phase d'analyse pertinente, une bonne stratégie de développement et beaucoup de motivation sont essentielles à la bonne réalisation d'un projet.

VII. ANNEXE

1. Diagramme de Gantt

Liste des tâches

Nom	Date de début	Date de fin	Ressources
UI	12/01/11	29/04/11	Antoine Berlon Thomas Clergeau Chegham wassim David Marginier
UI design Mettre en place la nouvelle IHM	12/01/11	25/02/11	Chegham wassim
v1	12/01/11	11/02/11	
v2	14/02/11	25/02/11	
Ground Ajout d'un sol dans le canvas	28/02/11	04/03/11	
Code Editor Module	07/03/11	18/03/11	
design	07/03/11	10/03/11	
Sync	10/03/11	18/03/11	
DB Module	03/02/11	09/04/11	
Local Storage Mettre à jour de la position/rotation/scale des objets en Local Storage	17/02/11	09/04/11	Afshin Faghihi Antoine Berlon Chegham wassim
IndexedDB Mettre à jour de la position/rotation/scale des objets en IndexedDB	03/02/11	09/04/11	David Marginier

Camera Module	07/03/11	05/04/11	
Camera FPS	07/03/11	26/03/11	Thomas Clergeau
Gestion de la caméra (à la FPS)			Chegham wassim
Camera Sketchup	28/03/11	05/04/11	Thomas Clergeau
Gestion de la caméra (à la Sketchup)			Chegham wassim
XML Parser Module	11/04/11	15/04/11	Chegham wassim
Gestion du parseur			
Upload Module	15/04/11	29/04/11	
Ajax Upload	15/04/11	21/04/11	
HTML5 Upload	21/04/11	29/04/11	
Documentation	17/01/11	09/04/11	
Test	24/01/11	09/04/11	
Compiler	12/01/11	17/03/11	Benjamin Le Galludec
			Mathieu Guichaoua
			Tolba Mohamed
			Amine
			Quentin Israel
Entities	04/02/11	10/02/11	
Definitions	04/02/11	17/02/11	
Commands	10/02/11	17/02/11	
Rules	10/02/11	24/02/11	
IA	09/03/11	17/03/11	
Media	09/03/11	17/03/11	
Tests	12/01/11	16/03/11	
Type	04/02/11	05/02/11	
3D Content	12/01/11	10/03/11	
Contenu pur	12/01/11	10/03/11	Aurélien Lubecki
			Erik Le Normand
jeu du lapin	12/01/11	10/02/11	
watch'n droid	12/01/11	10/02/11	
mario (plateforme générées aléatoirement)	12/01/11	10/02/11	
mario (niveau prédéfini)	12/01/11	10/02/11	
pacman	10/02/11	17/02/11	
billard	10/02/11	17/02/11	
1942	10/02/11	17/02/11	
fps deathmatch sans équipes	10/02/11	15/02/11	
fps deathmatch avec équipe (jusqu'à 3)	15/02/11	17/02/11	
fps capture du drapeau	17/02/11	19/02/11	
volley	17/02/11	24/02/11	
jeu de course	17/02/11	24/02/11	
mini rpg avec phases de jeu	24/02/11	10/03/11	
création modèles 3D	24/02/11	10/03/11	
création de tous les modèles de base pour l'utilisateur			
musiques ?	24/02/11	10/03/11	
(pas indispensable)			
Contenu autre	03/02/11	03/03/11	Aurélien Lubecki
concepts jeux	03/02/11	10/02/11	
généraliser les jeux pour créer des bases spécifiques lors de la création de jeux pour l'utilisateur			

formaliser ces jeux et voir avec le groupe compilation pour les ajouts à la grammaire (classe Game, nouveaux attributs 'world' et 'type')			
réflexion IA	03/02/11	10/02/11	
chercher des idées pour l'IA (regrouper les idées générales entre les types d'entités, les types de jeux)			
code IA	10/02/11	03/03/11	
coder des IA toutes faites en haut niveau ou javascript (voir avec Wassim), rajouter les mots-clés manquants dans la grammaire au besoin			
prévoir la possibilité de paramétrer les IA et d'en créer de nouvelles simplement pour l'utilisateur			
3D Engine	12/01/11	16/04/11	
Ajout des objets	12/01/11	26/01/11	Le Corronc Thibault Ludovic Zadith
Collision v1	12/01/11	26/01/11	Jérôme Bouzillard
Gestion clavier	12/01/11	26/01/11	Ludovic Zadith
Suppression d'objet	26/01/11	09/02/11	Le Corronc Thibault Ludovic Zadith
Déplacement	12/01/11	09/02/11	Philippe Weinzaepfel
Gestion des forces	23/02/11	09/03/11	Philippe Weinzaepfel
Collision v2	09/02/11	12/03/11	Le Corronc Thibault Ludovic Zadith Jérôme Bouzillard
Gestion caméras	24/01/11	24/02/11	Emeric Kien
Test et documentations code	21/03/11	16/04/11	Le Corronc Thibault Emeric Kien Ludovic Zadith Philippe Weinzaepfel Jérôme Bouzillard
Documentations	02/03/11	23/04/11	Afshin Faghihi
Intégration	14/03/11	16/04/11	Aurélien Sanvoisin

Liste des Ressources

Nom	Rôle par défaut
Afshin Faghihi	Editeur de doc
Tolba Mohamed Amine	Développeur
Antoine Berlon	Développeur
Aurélien Lubecki	Graphiste
Aurélien Sanvoisin	Chef de projet
Benjamin Le Galludec	Développeur
David Marginier	Développeur
Emeric Kien	Développeur
Erik Le Normand	Graphiste
Jérôme Bouzillard	Développeur
Ludovic Zadith	Développeur
Chegham wassim	Développeur
Mathieu Guichaoua	Testeur
Philippe Weinzaepfel	Développeur
Thomas Clergeau	Développeur
Le Corronc Thibault	Développeur
Quentin Israel	Développeur

Diagramme de Gantt

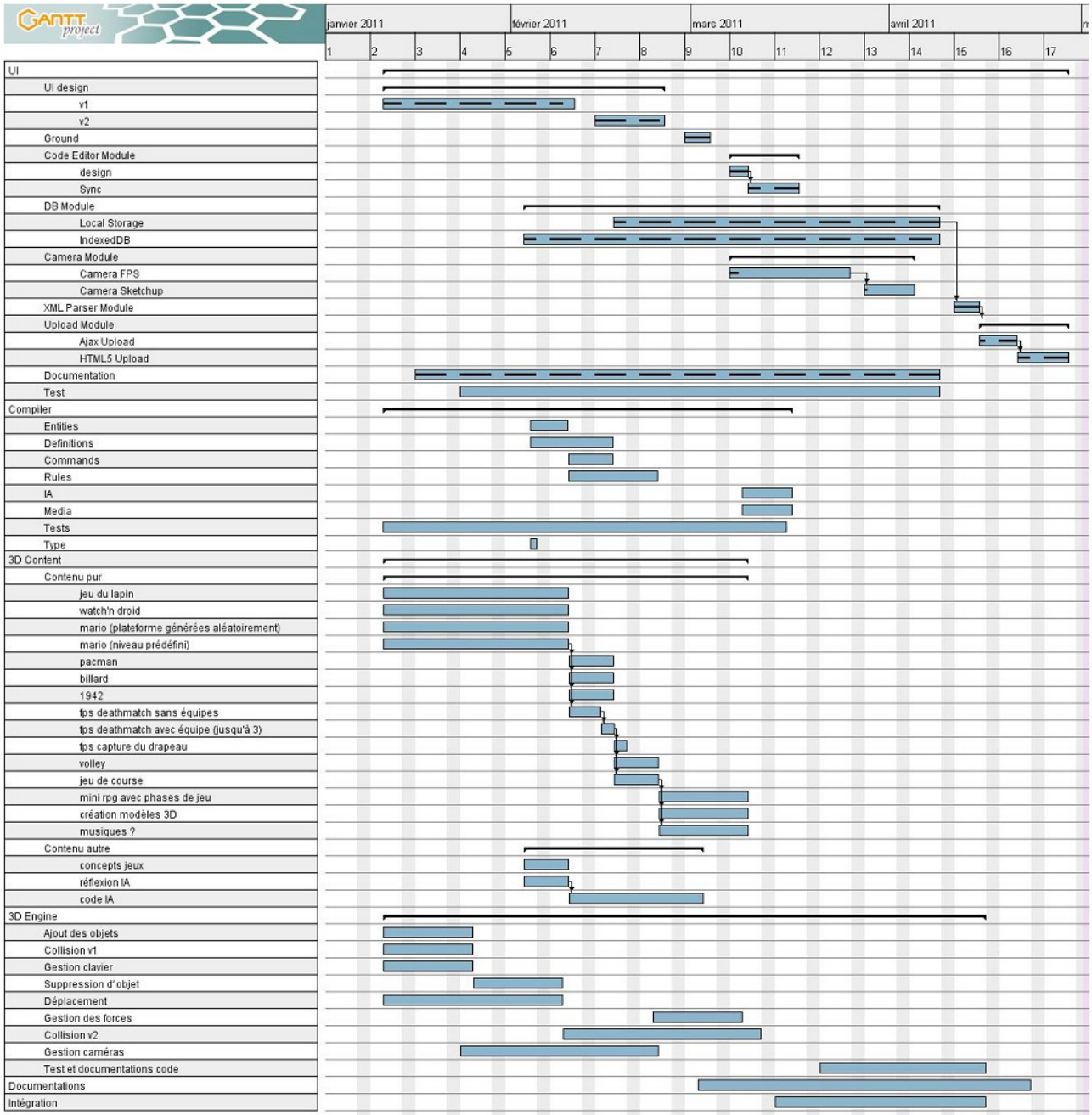
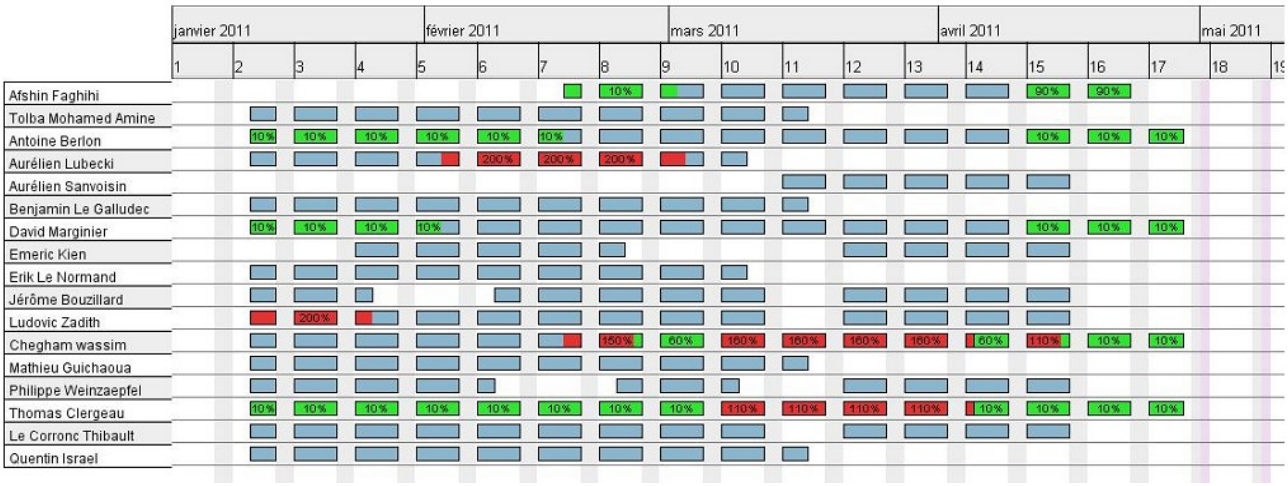


Diagramme des Ressources



2. Rapport de test

Nous avons effectué des tests JUnit sur la partie compilateur du code, car nous avons jugé qu'elle était la plus importante et donc à vérifier en priorité.

Les tests ont été effectués de manière "classique" : pour chaque fichier, un document de test a été écrit et nous avons rédigé des fonctions testant les différents paramètres (en entrée et en sortie) de chaque fonction d'origine. Dans le cas où une fonction ou des variables étaient privées, elles ont été passées en "public" le temps du test, puis la fonction test a été commentée, et les éléments concernés repassés en "private".

L'intégralité du compilateur a été testé de cette manière. Au final, les tests se sont bien passés, sans problèmes majeurs. Le cas échéant, ces problèmes ont donné lieu à des corrections sur le code d'origine.

Ci-dessous la visualisation Eclipse des tests JUnit du compilateur.