Présentation du projet pour les SAE S2.01, S2.02, S2.05 et S2.06

1 Le sujet : un Zelda-Like

Cette année, vous allez devoir inventer et programmer un Zelda à l'ancienne : en 2D avec vue du dessus donc plutôt à l'image des jeux sur Game Boy comme par exemple *A Link to the Past.* Vous avez le droit de vous inspirer largement des Zelda existants mais devez inventer votre propre univers.

L'originalité de votre univers sera un plus.

2 Les fonctionnalités prioritaires

Voici les fonctionnalités qui nous intéressent du point de vue pédagogique. Il faut donc que vous vous y intéressiez en priorité. Si vous ne parvenez pas à coder toutes ces fonctionnalités, cela ne signifie pas que vous n'aurez pas la moyenne.

- Le terrain doit être une tileMap engendrée à partir d'un fichier ou d'un tableau d'entiers. Le terrain n'est pas généré aléatoirement. Il n'est pas nécessaire (et surtout pas au début) d'avoir une scrolling map (une map qui se déplace en même temps que Link). On peut commencer avec une map statique pas plus grande que l'écran sur laquelle tout se déroule. De même, il est conseillé de commencer avec une map faite de 2 ou 3 tuiles. La construction d'une jolie map en utilisant un outil comme Tiled viendra plus tard.
  Quelques références pour comprendre et utiliser les tilemaps :
  - Tiles and tilemaps overview - Game development | MDN : tout petit article synthetisant assez bien l'utilité des tiles.
  - series de videos sur le développement de jeu vidéo : épisodes 20 à 23 sur les tile Map avec Tiled : jeu developpe 20 Les tuiles #1
  - tiled : http://www.demonixis.net/creer-une-map-avec-tiled/

- Link sait pousser, lancer des objets, se battre se déplacer, parler. Il est commandé au moyen du clavier (pas à la souris).
- Link démarre avec une seule arme, il en découvre d'autres (au moins 1) qui changent radicalement son comportement (pas seulement un peu plus de points d'attaque, par exemple le boomerang ou l'arc ou toute autre arme qui lance des projectiles).
- Link peut découvrir au moins 1 accessoire qui augmente ses possibilités (pas des armes). Par exemple quelque chose qui permet de voler ou d'aller sur l'eau ou de casser des éléments du décor ...
- Gestion de la vie : les coeurs.
- Il y a au moins 2 types d'ennemis avec des comportements (déplacement, attaque …) très différents. L'utilisation de l'algorithme BFS ou Dijkstra pour au moins un type d'ennemis serait appréciée.
- Un donjon avec un boss qui possède un schéma d'attaque élaboré.

3 Les contraintes techniques

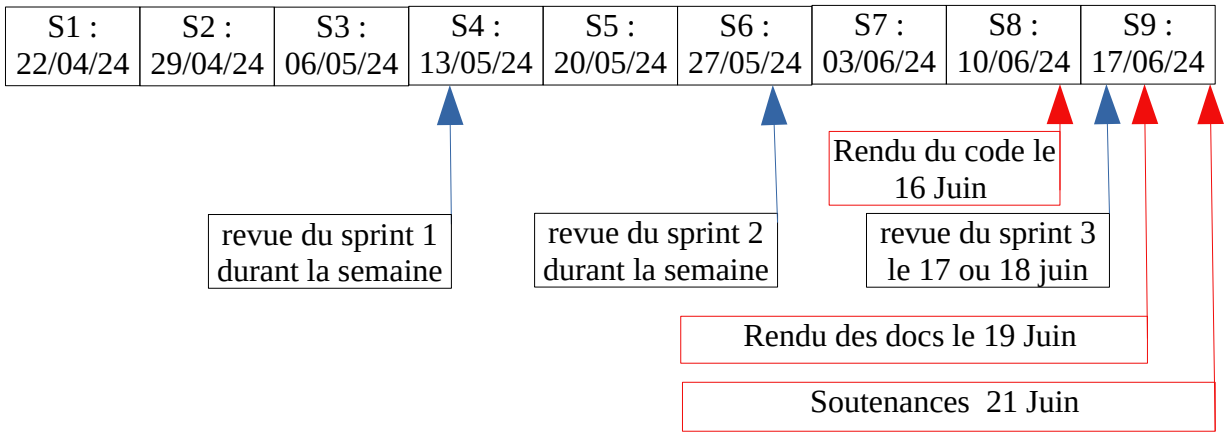
- Votre programme doit respecter une stricte séparation entre le modèle et la vue (cf loups moutons). Ceci n'est pas une architecture couramment utilisée dans les jeux vidéos donc attention à la copie de code trouvé sur internet : il y a fort peu de chances qu'il trouve grâce à nos yeux…
- La conception Objet de votre programme doit être très soignée. Vous verrez plus bas au paragraphe évaluation que le barème donne une place importante à la qualité du code.
- Votre programme doit pouvoir fonctionner avec des maps différentes. Les personnages doivent être capable de trouver leur chemin ou de se déplacer sans se cogner dans les obstacles, quelque soit la map.
- La partie statique de votre vue doit être spécifiée en utilisant FXML et le Scene Builder. Bien sûr, cela ne concerne pas l'affichage de la map qui devra être construite dynamiquement à partir d'un fichier et des images pour les tuiles.
- Il est interdit d'utiliser des API orientée Jeu ou des API pour la recherche de chemin dans la map.
- Vous devez utiliser Git/github : Votre enseignant doit être membre du projet. Vous devez avoir au moins 2 branches :
  - la branche master qui doit être propre, c'est à dire contenir du code mis en production.
  - Une branche dev qui est celle sur laquelle vous travaillez.
- Vous devez utiliser Trello, en respectant le modèle qui vous a été donné.

4 Organisation et évaluations

Ce projet regroupe les SAEs S2.01, S2.02, S2.05 et S2.06. Le projet donne donc lieu à une note dans chacune de ces SAE selon des critères d'évaluation différents qui seront détaillés dans la suite du document. Chaque note tient compte du code produit et d'un certain nombre de documents à fournir.

De plus, des heures provenant des ressources R2.01, R2.02, R2.03 et R2.10 sont allouées au projet afin de nous permettre de vous encadrer et de vous assister. Il pourra y avoir un contrôle individuel dans ces matières pendant la période de projet.

Le projet dure 9 semaines, et sera découpé en 3 **sprints** . Chaque sprint donne lieu à une évaluation. A l'issue du projet, il y a 3 rendus : le **rendu du code**, le r**endu des documents**, et la **soutenance**.



4.1 Le premier sprint

Il est fixé par les enseignants :

- Dans un document appelé cahier d'initialisation que vous ne devez plus changer par la suite et que vous envoyez à vos enseignants : description de votre jeu
    - L'univers du jeu, la quête principale (son but).
    - Mécanique des actions de Link (quelle touche pour pousser, tirer etc.) et des actions utilisateurs.
    - Les armes et les objets possédés par Link ou qu'il peut obtenir et comment ils influent sur ses capacités.
    - les éléments de la map et leur caractéristiques (obstacle indestructible ou descrutible, ressources ...)
    - Quelques ennemis avec leur mode de déplacement et d'attaque.
- initialisation de Trello et rédaction des cartes pour le sprint1 (enseignants référents du groupe sont ajoutés).
- codage : création d'une map basique + link qui se déplace. Coté modèle : un tableau d'entier ou de char décrivant les éléments du terrain (éventuellement engendré à partir d'un fichier). Coté vue création d'une TileMap pour afficher le terrain.
- Déplacement basique d'un personnage sur la map sans gestion de collision et sans BFS coté modèle et vue.
- Mise en place de la gameLoop.
- rédaction des cartes pour le deuxième sprint et répartition.

4.2 Les autres sprints :

- Au départ de chaque Sprint, vous décidez de ce que vous allez faire et répartissez les tâches (Trello).
- A l'issue de chaque sprint, la dernière version de votre projet doit être déposée sur la branche master de Git et une réunion se tient avec les enseignants : on fait le point sur le degré de réalisation du sprint, sur la qualité de la réalisation, sur l'investissement de chacun et la cohésion du groupe.
    - Évaluation "noire" : : pour SAE S2.06 (cf critères d'évaluation)
    - Evaluation "blanche" : pas une note définitive mais une indication du niveau de votre travail vous permettant de vous améliorer d'ici la fin du projet. A vous de réussir à faire évoluer votre projet.
    - Evaluation de l'apport de chacun dans le sprint servant à différencier éventuellement la note des membres du groupe.

5 Critères d'évaluation par matière

Les barèmes pour chaque SAE intervenant dans le projet sont présentés dans le tableau ci-dessous, puis explicités plus en détail dans les paragraphes qui suivent.

S2.01		**S2.02**		**S2.06**		**S2.05**	
POO	**25**	**Structure de données**	**30**	**soutenance**	**30**	**document utilisateur**	**40**
diagrammes de classes	**10**	**algorithmes**	**70**			**Trello**	**30**
tests	**15**			**notes d'itérations**	**50**	**Git**	**30**
programmation évènementielle et JavaFX	**30**			**expérience utilisateur**	**20**		
ampleur et qualité du code	**20**						
	100		**100**		**100**		**100**

5.1 SAE S2.01

Diagrammes de classe (10 points)
1 : Vous donnerez une vision d'ensemble de la partie modèle de votre programme à l'aide d'un ou de plusieurs diagrammes de classe commentés.
2 ; Vous choisirez des parties du modèle que vous considérez particulièrement intéressantes du point de vue de la programmation à objets (héritage, composition, polymorphisme…).
Vous expliquerez vos choix et les illustrerez par des diagrammes de classe.

Chaque diagramme utilisé doit être focalisé sur un objectif de communication. Il ne doit par exemple pas forcément montrer toutes les méthodes et dépendances, mais juste ce qui est nécessaire pour montrer ce que le diagramme veut montrer. Chaque diagramme doit être commenté.

POO (25 points)
Compréhension et mise en œuvre judicieuse de la programmation objets et de la programmation Java en particulier : classes, instances, composition héritage , encapsulation, principe de responsabilité unique, principe ouvert-fermé.

Tests Junit : (15 points)
Une ou deux classes bien choisies devront être accompagnées de tests Junit cohérents et raisonnablement complets.

Ampleur et qualité du code (20 points) : nombre de fonctionnalités présentes dans votre programme et qualité de leur programmation (lisibilité, respect des normes de programmation etc.).

Programmation événementielle et programmation javaFX (30 points)
Compréhension et mise en œuvre judicieuse de la programmation événementielle, de la programmation JavaFX en particulier, et de l'architecture MVC : utilisation de properties, de bindings, listeners, gestion des évènements, collection observable, definition fxml de la partie statique de la vue. Degré de sophistication technique de l'IHM (plusieurs images pour les sprites, map scrolling, drag and drop …).

5.2 SAE S2.02

Structures de données (30 points)
Bonne utilisation de tableaux, collections simples (ArrayList), collections avancées (map, set…)

algorithmique (70 points)
Qualité et difficulté des algorithmes présents dans votre programme.

5.3 SAE S2.06

Soutenance (30 points)
C'est la prestation de communication qui est notée. 25 mn d'exposé + 10 minutes de questions.

Contenu de l’exposé :

1. (5 mn) : cette partie doit s’adresser aux utilisateurs de votre jeu. Vous y décrirez votre jeu et ferez une courte démonstration commentée (cela peut se faire en direct ou dans une vidéo). C’est un travail de communication de l’ensemble du groupe et la parole doit être équitablement répartie.
2. (15 mn soit 5mn par personne) : cette partie s’adresse à vos enseignants. C’est un travail de communication individuel. Chaque étudiant doit faire un bilan personnel répondant au minimum aux points suivants :
   - ce que j’ai fait dans le projet ;
   - mon avis sur le fonctionnement du groupe et sur ma place dans le groupe ;
   - ce que j’ai appris / mes difficultés et mes points forts.
3. De plus chaque groupe doit fournir la répartition en pourcentage de l’apport de chacun des membres du groupe dans le projet, sous la forme d’un camembert

Note d’itérations (50 points)
l’ensemble des revues de sprints donne lieu à une note prenant en compte le travail réalisé, l’investissement du groupe et l’investissement des membres du groupe dans le projet et le bon fonctionnement de l’équipe.

Expérience utilisateur (20 points)
ergonomie, fluidité, raffinements graphiques ou sonores de l’IHM.

.

5.4 SAE S2.05

Document utilisateur (40 points) :

- Objectif et univers du jeu.
- Présentation du jeu avec captures d’écran.
- Mécanique des actions utilisateurs.
- Caractéristiques des armes et accessoires de Link, des types d’éléments de la map, des ennemis (tableau de relation ...).
- Toutes fonctionnalité définie et utilisable doit être documentée. A l’inverse il est hors de question (et donc pénalisant) de documenter une fonctionnalité non encore utilisable.

Utilisation de Trello et git (60 points)
Bonne utilisation des outils Trello et Git.