Compte rendu de réunion du Sprint 3

Backlog

```
(Clément) - Refactor des liens entre les Character et MapBuilder
(Clément) - signaler erreur de map si nombre de passages != 0 ou !=2
- attaques
   (Clément) → orienter l'attaque
(Adèle) → attaque corps à corps perso
   --|> verif : être sur une case adjacente
   --|> dégât sur le monstre
   (Adham) -- | > délai sur attaque : ajouter reload sur interface et dans le modèle
(Adèle, Clément) → attaque distance perso
   --|> attaque sur 3 cases (à travers les murs)
(Raphaël) → dégât causé (par le monstre ou par le perso)
   --|> animation clignote
   --|> vie baisse
   --|> si vie perso = 0 : game over
   --|> si vie monstre = 0 : monstre disparait
(Adham) → niveaux multiples
 → 2 types de niveaux
   --|> intermédiaire : trésor = niveau suivant
   --|> final : trésor = victoire
- Revoir tests
    (Raphaël) → Animation, PacmanGame, EffectMagic
    (Adèle) → Trap, PacamanCharacter
```

Sprint review sprint 3

```
(Clément) - Refactor des liens entre les Character et MapBuilder FAIT
(Clément) - signaler erreur de map si nombre de passages != 0 ou !=2 FAIT
- attaques
 (Clément)--> orienter l'attaque FAIT
(Adèle)--> attaque corps à corps perso FAIT
   --|> verif : être sur une case adjacente
   --|> dégât sur le monstre
   (Adham) -- |> délai sur attaque : ajouter reload sur interface et dans le modèle FAIT
 (Adèle, Clément)--> attaque distance perso FAIT
   --|> attaque sur 3 cases (à travers les murs)
 (Raphaël) --> dégât causé (par le monstre ou par le perso) FAIT
   --|> animation clignote
   --|> vie baisse
   --|> si vie perso = 0 : game over
   --|> si vie monstre = 0 : monstre disparait
(Adham) - niveaux multiples FAIT
 --> 2 types de niveaux
   --|> intermédiaire : trésor = niveau suivant
   --|> final : trésor = victoire
- Revoir tests NON FAIT
 (Raphaël) --> Animation, PacmanGame, EffectMagic
 (Adèle) --> Trap, PacmanCharacter
```

Sprint rétrospective

→ Problèmes rencontrés :

- Récupération de fichiers non testés qui ont dû être corrigés après l'heure prévue de mise en commun ;
- Classe de tests qui n'ont pas été mise à jour.

→ Points positifs :

- Les fonctionnalités à implanter étaient plus simples à implanter donc elles ont été plus rapides à développer ;
- Le projet était stable à la sortie du sprint 2, ce qui nous a permis d'avoir une base stable pour implanter les fonctionnalités de ce sprint et être plus efficaces ;
- La réunion de fin de sprint plus tôt qu'à l'accoutumé ce qui nous permis de régler les conflits de mise en commun, ou erreur de développement plus calmement que pendant les autres sprints.

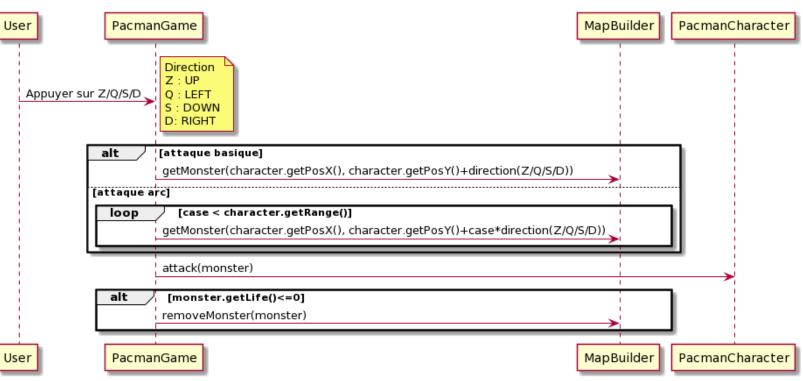
→ Comment résoudre les problèmes que nous avons rencontrés :

- Détailler très précisément comment faire certaines fonctionnalités pour être sûre que la personne qui la développe ne rende le projet non évolutif, ou incompatible avec certaines parties du projet ;
- Lancer systématiquement les tests <u>avant</u> de push pour s'assurer que le projet est stable et que les tests sont à jour.

Note: la revue des tests n'a pas été faite sur ce sprint, nous avons préféré garder cette tâche pour la stabilisation du projet. Les tests présents sont fonctionnels mais ne sont pas en adéquation avec ce que nous avons pu apprendre en cours. Il paraissait préférable de tous les corriger sur un même sprint que de les modifier de manière éparse.

Conception

GESTION DES ATTAQUES (PACMAN)



<<Java Class>> G Character model posX: double posY: double speed: double ♦ life: int range: int movingStrategy: MovingStrategy animation: Animation Character(double, double) <<Java Class>> setMovingStrategy(MovingStrategy):void MonsterCharacter getMovingStrategy():MovingStrategy model getAnimation():Animation strategies: List<MovingStrategy> mooveRight():void MonsterCharacter(double, double) mooveLeft():void setPosX(double):void mooveUp():void setSpeed(double):void setPosY(double):void getSpeed():double getStrength():int getLife():int isMonster():boolean mooveDown():void toString():String setPosX(double):void setPosY(double):void getPosX():double getPosY():double canMoove(double,double,MapBuilder):boolean setDamage(int):void agetStrength():int isMonster():boolean attack(Character):void getMovingStrategyType():String <<Java Class>> resetLife():void PacmanCharacter model usitedCoordinates: List<int[]> -pacmanCharacter setRange(int):void 0..1 getRange():int setPosX(double):void setPosY(double):void getStrength():int setCoordinates(double, double):void setMovingStrategy(MovingStrategy):void getMovingStrategy():MovingStrategy getVisitedCoordinates():Iterator<int[]> toString():String

GESTION DES PERSONNAGES ET DES ATTAQUES

<<Java Class>> PacmanGame model Fscale: int mapBuilder: MapBuilder p isFinished: boolean executedEffect: Ground monsterMooveCounter: int a canHit: boolean monsterCanHit: boolean evolve(Cmd):void attackMonster(int,int):void attackPlayer():void mooveMonster():void consumeGroundEffect(int,int):void anMoove(Character, double, double):boolean printGame(Cmd):void isFinished():boolean getNbMonsters():int getWidth():int getHeight():int getCharacterPosX():double getCharacterPosY():double getMonsterPosX(int):double getMonsterPosY(int):double getMapBuilder():MapBuilder ■ resetPosition():void getNearestEffectiveGround(double,double):Ground getCollidingGrounds(double,double,MapBuilder):Iterator<Ground> SisBlocked(double, double, MapBuilder): boolean ScalculateDistance(double, double, double, double):double doEffect(boolean, Cmd):void o getScale():int oetCharacter():PacmanCharacter delayPacmanAttack(int):void delayMonsterAttack(int):void

canHit():boolean