**« Backlog » de sprint #001**

Produit : Hitit

Conçu par : La Classe Manteau

## Nom des membres :

Vincent Boulet (Leader), Frédérick Bonnelly (Assurance-qualité), Tanahel Huot-Roberge (Tests), Tommy Montreuil (Secrétaire).

## Éch**é**ancier (période):

Du 9 février au 23 février.

## Légende :

* Vert, indique que ces items sont réalisés.
* Jaune, indique que ces items sont en cours de réalisation.
* Rouge, problème ou questionnement important qui demande une rencontre d’équipe.
* Aucune couleur, indique que ces items ne sont pas encore faits ou commencés, **on peut toujours les enrichir mais il faut le consentement de toute l’équipe**.

## « Backlog » de sprintw

|  |  |
| --- | --- |
| **2** | |
| Acteur ou rôle : | Professeur |
| Scénario ou story : | En tant que professeur, je veux pouvoir choisir parmi plusieurs projectiles différents. |
| Détail et description des **items** à faire : | 1. FXML : Mettre en place le choix des projectiles à l’aide de « radio-boutons ». (Boutons Sphère, Cube, Flèche). Les radios boutons seront positionnés dans un titledpane. 2. Programmer une méthode qui lie le projectile choisie au lanceur. 3. Programmer les méthodes suivantes (Voir vpp).      * Ajouter les méthodes de validation des attributs pour chacun des projectiles. * Méthodes : validerMasse() : entre.1 kg et 1000kg . ValiderTaille() dans la classe interface : entre 0 et 1000 pour le rayon et 0 et 500 m pour les côtés du cube. * FXML : Mettre en place le titledpane des paramètres donc à l’aide de labels et text fields. (Paramètres : Masse, Rayon). |
| Tests d’acceptation : | JUnits : tester le fonctionnement de chaque méthode programmé ci-haut. S’assurer que l’on peut bien entrer des données dans l’interface et que les sections se grisent quand on sélectionne un type de projectile. |
| Complexité : | Simple |
| Effort : | Modéré |
| Commentaires : | N/A |

|  |  |
| --- | --- |
| **2** | |
| Acteur ou rôle : | Professeur |
| Scénario ou story : | En tant que professeur, je veux pouvoir choisir parmi différents lanceurs. |
| Détail et description des **items** à faire : | 1. FXML : Mettre en place le choix des lanceurs à l’aide de « radio-boutons ». (Canons, ressort). Les radios boutons seront positionnés dans un titledpane. 2. Programmer les méthodes suivantes (Voir vpp). 4. Ajouter les méthodes de validation :   ValiderPositionsIni(pos x(min 0 et max 300) , pos y(0 à 300)), validerAngle( entre 0 et 90 exclus), validerVitesse( 0 à 1000 m/s) validerCRappelRessort(0 de a 1000 N­­/m). ValiderEtirement(0 à 5m).validerEnergie 0 a 50 Kj).   1. FXML : s’assurer que lorsque l’on choisit un type de lanceur, le bon type de projectile disponible devient accessible au choix. |
| Tests d’acceptation : | JUnits : tester le fonctionnement de chaque méthode programmé ci-haut. S’assurer que l’on peut bien entrer des données dans l’interface et que les types de projectiles de grisent lorsque l’on choisit un lanceur précis. |
| Complexité : | Modéré |
| Effort : | Modéré |
| Commentaires : | N/A |

|  |  |
| --- | --- |
| **3** | |
| Acteur ou rôle : | Professeur |
| Scénario ou story : | En tant que professeur, je veux pouvoir choisir parmi différentes cibles. |
| Détail et description des **items** à faire : | 1. FXML : Mettre en place le choix des cibles à l’aide de « radio-boutons ». (Cercle concentrique, cible mouvante, dans les airs ou au sol). Les radios boutons seront positionnés dans un titledpane. Lier la vue aux paramètres de la cible. 2. Programmer les méthodes suivantes (Voir vpp).      1. Ajouter les méthodes de validation : validerRayon(de 0 a 200m). validerOrientation( de 0 à 90 degrés), validerVitesses( de 0 a 2m/s). 2. FXML : Mettre en place le titledpane des paramètres donc à l’aide de labels et text fields. (Paramètres  Orientation, Rayon, Vitesse x et vitesse y). |
| Tests d’acceptation : | JUnits : tester le fonctionnement de chaque méthode programmé ci-haut. S’assurer que l’on peut bien entrer des données dans l’interface. |
| Complexité : | Modéré |
| Effort : | Modéré |
| Commentaires : | N/A |

|  |  |
| --- | --- |
| **4** | |
| Acteur ou rôle : | Professeur |
| Scénario ou story : | En tant que Professeur je veux pouvoir ouvrir l’application. |
| Détail et description des **items** à faire : | 1. Faire le FXML du menu principal. 3. Ajouter le titlePaned pour les forces (Vent, Gravité, Densité de l’air) 4. Faire le FXML de la page d’accueil. 6. Mettre en place la Contrôleur. (Setter les attributs FXML : positionCibleX, positionCibleY, inclinaisonCible, vitesseCible, positionLanceurX, positionLanceurY, vitesseLanceur, angleLanceur, rayonProjectile, masseProjectile, largeurProjectile(cube), radioLanceurs, radioProjectiles, radioCibles, radioObstacles, radioForces. 7. Chaque radio button obstacle ouvre un pane avec les propriétés correspondantes. Les propriétés seront pour la position en x et la hauteur. Le pont laisse 2 possibilités de tir, au-dessus et en dessous. La fenêtre oblige l’utilisateur à faire passer le projectile dans un espace limité. Le toit offre une hauteur limite. 8. Les radios buttons sont gérés par groupe (type) donc les lanceurs ensemble, les projectiles ensemble, les cibles ensemble et les obstacles en semble. Donc il y a un écouteur par groupe de radio-button. 9. Faire une méthode qui écoute les boutons mode libre et mode exercice et qui mène l’utilisateur au bon menu. Pour l’instant les menus libre et exercice seront identiques. 10. Faire les méthodes d’écoute pour les boutons dans le menu. Structure de méthode : Écouter « nom du paramètre » pour les textfield et Écouter  « nom du radio button » pour les radios button. 11. Mettre en place la classe Main. 12. Faire la méthode initialize : Elle set les paramètres par défaut, donc cible et lanceur à une position y qui vaut 0 à partir du bas et à une position x par défaut. Pour le lanceur, il est positionné à 10 px de la bordure gauche et la cible à 10px. Le lanceur de base sera le canon. Il n’y a pas d’obstacles lors de l’ouverture. Aucune force non plus. 13. Faire apparaitre le canon et la cible lors de l’ouverture de l’application. 14. Ajouter un bouton qui lance la simulation dans le bas à droite, mais sans listener. |
|  | S’assurer que l’application ouvre et qu’il y a une page d’accueil qui mène aux modes ainsi que les boutons de la fenêtre de simulation fonctionne et que l’on peut entrer les paramètres dans les text fields. S’assurer que le lanceur et la cible apparaissent en ouvrant l’app. S’assurer que tous les paramètres nécessaires sont affichés à l’écran. |
| Complexité : | Modéré ++ |
| Effort : | Modéré + |
| Commentaires : | Cette tâche est plus complexe que les 3 autres, ainsi les 3 autres coéquipiers pourront aider à compléter cette tâche s’ils ont terminé la leur. |