**« Backlog » de sprint #000**

Produit : Astéria

Conçu par : EquBolduc

## Équipe :

|  |  |  |
| --- | --- | --- |
| **Nom** | **Initiales** | **Responsabilité** |
| Émile Grégoire | É. G. | Chef d’équipe |
| Jonathan Samson | J. S. | Responsable des livrables |
| Simon-Pierre Deschênes | S.-P. D. | Responsable de la qualité |
| Jérémie Bolduc | J. B. | Responsable des réunions |

## Éch**é**ancier:

Du 11 février au 25 février 2015.

## 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 sprint

|  |  |  |  |
| --- | --- | --- | --- |
| **Scénario #2** | | | |
| Acteur | | | Développeurs |
| Scénario | | | En tant que développeur, je veux pouvoir changer de contrôleur et de vue facilement afin de simplifier les tests sur le modèle physique. |
| Description | | | Créer un contrôleur principal qui contiendra les vues et les contrôleurs.   * 1. Qui et temps      1. É.G.      2. 30 min.   2. Préconditions   3. Règles d’affaires      1. Créer un contrôleur principal qui contient les vues et les contrôleurs.      2. Rendre ce contrôleur accessible depuis partout (Singleton).      3. Créer les interfaces contrôleurs et vues.   4. Règles d’affaires alternatives      1. S’assurer que les contrôleurs et les vues ne sont pas null.   5. Tests d’acceptation      1. Créer des tests unitaires avec des classes contrôleurs et vues bidons.   6. Post-conditions      1. Le contrôleur principal est accessible partout.   Implémenter les changements entre les contrôleurs.   * 1. Qui et temps      1. É.G.      2. 30 min.   2. Préconditions      1. Le contrôleur principal est accessible partout.   3. Règles d’affaires      1. Charger le contrôleur.      2. Ajouter les écouteurs nécessaires.   4. Règles d’affaires alternatives      1. S’assurer que le contrôleur n’est pas null.   5. Tests d’acceptation      1. Créer des tests unitaires avec une classe contrôleur bidon.   6. Post-conditions   Implémenter le chargement des vues.   * 1. Qui et temps      1. É.G.      2. 30 min.   2. Préconditions      1. Le contrôleur principal est accessible partout.   3. Règles d’affaires      1. Charger le fichier FXML.      2. Charger le fichier CSS.   4. Règles d’affaires alternatives      1. S’assurer que la vue n’est pas null.   5. Tests d’acceptation      1. Créer des tests unitaires avec une vue bidon.   6. Post-conditions |
| Tests d’acceptation | | | Créer deux contrôleurs de test et de vue pour s’assurer que les changements s’opèrent bien. |
| Complexité | | | 3 |
| Effort | | | 2 |
| Commentaires | | |  |
| **Scénario #3** | | | |
| Acteurs | | | Développeurs |
| Scénario | | | En tant que développeur, je veux pouvoir me servir d’une horloge interne qui me retournera des variations de temps utiles dans les formules de physique. |
| Description | | | 1. Écrire un thread qui calcule la différence de temps entre un temps x et y. 2. Dans le contrôleur principal, démarrer le thread. 3. Accéder aux valeurs de variations de temps. |
| Tests d’acceptation | | | Demander d’afficher le nombre de secondes ou minutes qui se sont écoulées entre deux appels de la méthode. |
| Complexité | | | 2 |
| Effort | | | 1 |
| Commentaires | | |  |
| **Scénario #4** | | | |
| Acteurs | | | Développeurs |
| Scénario | | | En tant que développeur, je veux utiliser une classe Vecteur afin d’illustrer les mouvements entre les différents corps. |
| Description | | | 1. Implémenter l’addition de vecteurs. 2. Implémenter la soustraction de vecteurs. 3. Implémenter la multiplication de vecteurs par des scalaires. 4. Implémenter le produit scalaire de vecteurs. 5. Implémenter le produit vectoriel. 6. Implémenter la normalisation de vecteurs. 7. Implémenter le calcul des modules de vecteurs. |
| Tests d’acceptation | | | Effectuer différentes opérations vectorielles et vérifier la cohérence des résultats via des tests unitaires. |
| Complexité | | | 1 |
| Effort | | | 1 |
| Commentaires | | |  |
| **Scénario #5** | | | |
| Acteurs | | | Développeurs |
| Scénario | | | En tant que développeur, je veux pouvoir travailler avec des corps physiques qui seront soit statiques, soit en mouvement. |
| Description | | | 1. Créer l’interface Corps qui sera utilisée par les vaisseaux et les planètes.    1. Qui et temps       1. J.B.       2. 1 h.    2. Préconditions       1. La classe Vecteur est opérationnelle.    3. Règles d’affaires       1. Créé les méthodes relative à la masse.       2. Créé les méthodes relative à la vitesse.       3. Créé les méthodes relative à la position.       4. Créé les méthodes relative aux forces.    4. Règles d’affaires alternatives       1. S’assurer que la vue n’est pas null.    5. Tests d’acceptation       1. Créer des tests unitaires avec une vue bidon.    6. Post-conditions   Créer la classe Planète qui implémente l’interface Corps.   * 1. Qui et temps      1. J.B.      2. 1 h 30 min.   2. Préconditions      1. L’interface corps est créée.   3. Règles d’affaires      1. Créé les attributs de la classe (property).      2. Compléter les méthodes de l’interface Corps.   4. Règles d’affaires alternatives      1. S’assurer que les attributs ne sont pas null   5. Tests d’acceptation      1. Créer des tests unitaires avec des données bidon.   6. Post-conditions   Créer la classe Vaisseau qui implémente l’interface Corps.   * 1. Qui et temps      1. J.B.      2. 1 h 30 min.   2. Préconditions      1. L’interface corps est créée.   3. Règles d’affaires      1. Créé les attributs de la classe (property).      2. Compléter les méthodes de l’interface Corps.      3. Créé les méthodes pour tourner.   4. Règles d’affaires alternatives      1. S’assurer que les attributs ne sont pas null   5. Tests d’acceptation      1. Créer des tests unitaires avec des données bidon.   6. Post-conditions |
| Tests d’acceptation | | | Vérifier que les valeurs retournées ont du sens. |
| Complexité | | | 1 |
| Effort | | | 2 |
| Commentaires | | |  |
| **Scénario #6** | | | |
| Acteurs | | | Développeurs |
| Scénario | | | En tant que développeur, je veux que les différents corps puissent s’attirer entre eux afin que le jeu soit réaliste physiquement. |
| Description | | | 1. Récupérer tous les corps. 2. Analyser les propriétés des différents corps. 3. Pour chaque corps, calculer la force résultante exercée par les autres corps. |
| Tests d’acceptation | | | Mettre plusieurs corps à proximité et vérifier que leur comportement est réaliste grâce à des calculs fait à la main. |
| Complexité | | | 3 |
| Effort | | | 3 |
| Commentaires | | |  |
| **Scénario #7** | | | |
| Acteurs | | | Utilisateur |
| Scénario | | | En tant qu’utilisateur, je veux pouvoir voir les différents corps afin de pouvoir voir ce qui se passe. |
| Description | | | 1. Charger la vue. 2. Récupérer les différents nœuds à afficher. 3. Afficher les nœuds. |
| Tests d’acceptation | | | Démarrer le programme et vérifier que la fenêtre s’affiche. |
| Complexité | | | 2 |
| Effort | | | 1 |
| Commentaires | | |  |
| **Scénario #8** | | | |
| Acteurs | | | Utilisateur |
| Scénario | | | En tant qu’utilisateur, je veux pouvoir mettre le jeu en pause afin de pouvoir réfléchir. |
| Description | | | 1. Mettre l’horloge interne sur pause. 2. Créer la classe ContrôleurMenu. 3. Créer la classe VueMenu. 4. Charger le contrôleur. 5. Charger la vue. |
| Tests d’acceptation | | | Démarrer le programme, appuyer sur pause et s’assurer que le menu pause fonctionne. |
| Complexité | | | 2 |
| Effort | | | 1 |
| Commentaires | | |  |
| **1** | | | |
| Acteur ou rôle : |  | | |
| Scénario ou story : |  | | |
| Détail et description des **items** à faire : |  | | |
| Tests d’acceptation : |  | | |
| Complexité : |  | | |
| Effort : |  | | |
| Commentaires : |  | | |