**« Backlog » de sprint #003**

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 |

## Échéancier:

Du 11 mars au 1er avril 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 #13** | |
| Acteur | Utilisateur |
| Scénario | En tant qu’utilisateur, je veux pouvoir recommencer le jeu à n’importe quel moment. |
| Description | 1. Écouter la touche R.   * 1. Qui et temps      1. J.B.      2. 10 min.   2. Préconditions   3. Règles d’affaires      1. Mettre un écouteur sur la touche R      2. Enlever l’écouteur lorsque le menu pause est activé   4. Règles d’affaires alternatives   5. Tests d’acceptation      1. La physique s’arrête dans le jeu   6. Post-conditions   2. Créer un bouton dans le menu pause et l’écouter.   * 1. Qui et temps      1. J.B.      2. 30 min.   2. Préconditions   3. Règles d’affaires      1. Créer un bouton dans le menu pause      2. ‘Mettre un écouteur sur le bouton   4. Règles d’affaires alternatives   5. Tests d’acceptation   6. Post-conditions   3. Replacer les corps à leur place de départ.   * 1. Qui et temps      1. J.B.      2. 1 h   2. Préconditions   3. Règles d’affaires      1. Obtenir la liste des corps à leur position de départ      2. Remplacer les corps   4. Règles d’affaires alternatives   5. Tests d’acceptation   Post-conditions |
| Tests d’acceptation | Appuyer sur R et vérifier que les corps reviennent à la position de départ, essayer aussi avec le bouton du menu |
| Complexité | 2 |
| Effort | 2 |

|  |  |
| --- | --- |
| **Scénario #14** | |
| Acteur | Utilisateur |
| Scénario | En tant qu’utilisateur, je veux pouvoir sauvegarder mon progrès au jeu. |
| Description | 1. Sauvegarde du niveau   * 1. Qui et temps      1. S.-P.D.      2. 25 minutes   2. Préconditions      1. Les objets Niveau sont terminés.   3. Règles d’affaires      1. Créer un flux d’écriture avec un dossier de destination des niveaux sauvegardés.      2. Écrire un objet Niveau dans le dossier de destination.   4. Règles d’affaires alternatives   5. Tests d’acceptation      1. Sauvegarder un niveau et aller vérifier que le fichier de sauvegarde est bien présent dans le dossier de destination.   6. Post-conditions   2. Chargement du niveau   * 1. Qui et temps      1. S.-P.D.      2. 25 minutes   2. Préconditions      1. La sauvegarde est terminée.   3. Règles d’affaires      1. Créer un flux de lecture avec un dossier source.      2. Lire un objet Niveau dans le dossier source.   4. Règles d’affaires alternatives   5. Tests d’acceptation      1. Charger un niveau et vérifier que les éléments sont aux bons endroits dans le niveau.   6. Post-conditions |
| Tests d’acceptation | Sauvegarder un niveau et le charger par la suite. Si le niveau est identique au début et à la fin, la sauvegarde et le chargement fonctionnent. |
| Complexité | 3 |
| Effort | 3 |
| Commentaires |  |
| **Scénario #15** | |
| Acteur | Utilisateur |
| Scénario | En tant qu’utilisateur, je veux que le tableau se finisse lorsque je réussis un objectif. |
| Description | 1. Créer la classe Objectif   * 1. Qui et temps      1. É. G.      2. 45 min.   2. Préconditions   3. Règles d’affaires      1. Écrire et documenter une interface permettant de vérifier si l’objectif est atteint.      2. Écrire une classe ObjectifRayon qui vérifie si un vaisseau se retrouve dans un certains rayon.   4. Règles d’affaires alternatives   5. Tests d’acceptation      1. Vérifier par des tests unitaires que ObjectifRayon répond aux critères.   6. Post-conditions      1. Un objectif de base existe.   2. Implémenter la vérification des objectifs.   * 1. Qui et temps      1. É. G.      2. 45 min.   2. Préconditions      1. L’interface Objectif et une classe ObjectRayon existe.   3. Règles d’affaires      1. Ajouter un objectif dans la classe Niveau.      2. Dans ContrôleurJeu, vérifier dans la méthode update() si l’objectif est atteint.      3. Afficher à l’écran la réussite du tableau.   4. Règles d’affaires alternatives      1. Si aucun objectif n’est défini, ne pas vérifier la réussite du tableau.   5. Tests d’acceptation      1. Vérifier par un tableau de démo que l’objectif est atteint.   6. Post-conditions      1. Les objectifs sont vérifiés par le jeu. |
| Tests d’acceptation | Créer un niveau de test et vérifier si l’atteinte de l’objectif provoque la réussite de l’objectif. |
| Complexité | 3 |
| Effort | 3 |
| Commentaires |  |
| **Scénario #16** | |
| Acteur | Utilisateur |
| Scénario | En tant que qu’utilisateur, je veux pouvoir choisir mon niveau dans une interface différente afin de voir combien de niveau le jeu comporte. |
| Description | 1. Créer un nouveau contrôleur et une nouvelle vue qui gèreront l’interface graphique de sélection de niveau préconçus.  Qui et temps   * + 1. Jonathan Samson     2. 2 heures   Préconditions  1.2.1 Aucune précondition, c’est un contrôleur indépendant des autres.  Règles d’affaires  1.3.1 On doit pouvoir choisir son niveau parmis plus de 5 niveaux disponibles.  1.3.2 Il doit y avoir une fenêtre qui affiche les caractéristiques du niveau.  1.3.3 Il doit y avoir un bouton jouer pour commencer le niveau sélectionné.  1.3.4 Il doit y avoir un bouton pour le mode campagne.  Règles d’affaires alternatives  1.4.1 On doit être capable de revenir au menu principal facilement selon un bouton retour.  Tests d’acceptation  1.5.1 Avoir les boutons fonctionnels suivants : Campagne, #x, Jouer et Retour.  1.5.2 Avoir la petite fenêtre qui affiche les caractéristiques du niveau.  Post-conditions  1.6.1 Nous avons une interface joueur pour la sélection du niveau. |
| Tests d’acceptation | Avoir les boutons fonctionnels suivants : Campagne, #x, Jouer et Retour.  Avoir la petite fenêtre qui affiche les caractéristiques du niveau. |
| Complexité | 3 |
| Effort | 2 |
| Commentaires |  |
| **Scénario #17** | |
| Acteur | Utilisateur |
| Scénario | En tant que qu’utilisateur, je veux customiser les objets spatiaux dans l’éditeur de niveau. |
| Description | 1. Sélectionner les objets spatiaux et les modifier.  Qui et temps   * + 1. JS ET JB     2. 3 heures   Préconditions  1.2.1 Avoir un éditeur de niveau qui fonctionne.  Règles d’affaires  1.3.1 Avoir accès à différents paramètres d’objets spatiaux comme la masse, la texture.  1.3.2 Il doit y avoir une toolbar pour contrôler ces paramètres.  Règles d’affaires alternatives  Tests d’acceptation  1.5.1 Les objets modifiés sont jouables dans les niveaux.  1.11 Post-conditions  1.6.1 Nous avons accès à des objets modifiés par le joueur. |
| Tests d’acceptation | Les objets modifiés fonctionnent dans le jeu. Ils sont customisables. |
| Complexité | 3 |
| Effort | 2 |
| Commentaires |  |
| **Scénario #18** | |
| Acteur | Développeur |
| Scénario | En tant que développeur, je veux que des ensembles de corps et d’objectifs d’un niveau soient stockés dans une classe Niveau afin de pouvoir stocker différents niveaux. |
| Description | 1. Création de l’attribut point de départ   * 1. Qui et temps      1. S.-P.D.      2. 5 minutes   2. Préconditions   3. Règles d’affaires      1. Créer l’attribut point de départ du type Vecteur.   4. Règles d’affaires alternatives   5. Tests d’acceptation      1. Décider d’un point de départ et vérifier qu’il est respecté.   6. Post-conditions   2. Création de l’attribut titre du niveau   * 1. Qui et temps      1. S.-P.D.      2. 5 minutes   2. Préconditions   3. Règles d’affaires      1. Créer l’attribut titre du niveau.   4. Règles d’affaires alternatives   5. Tests d’acceptation      1. Décider d’un titre et vérifier qu’il est respecté.   6. Post-conditions   3. Création des méthodes d’accès et d’altération   * 1. Qui et temps      1. S.-P.D.      2. 15 minutes   2. Préconditions      1. Les différents corps doivent être terminés.   3. Règles d’affaires      1. Créer les méthodes d’accès et d’altération de l’attribut point de départ.      2. Créer les méthodes d’accès et d’altération de l’attribut titre du niveau.      3. Créer les méthodes d’accès et d’altération de la liste de corps.      4. Créer les méthodes d’accès et d’altération de la liste d’objectifs.   4. Règles d’affaires alternatives   5. Tests d’acceptation      1. Mettre un niveau dans un objet Niveau et reconstruire le même niveau par la suite seulement grâce à l’objet Niveau.   6. Post-conditions   4. Création du constructeur   * 1. Qui et temps      1. S.-P.D.      2. 15 minutes   2. Préconditions      1. Les différents attributs doivent être terminés.   3. Règles d’affaires      1. Créer le constructeur avec comme paramètres : un vecteur point de départ, un une liste de corps et une liste d’objectifs.   4. Règles d’affaires alternatives   5. Tests d’acceptation      1. Créer un objet niveau en lui passant différentes valeurs en paramètre et vérifier que les valeurs sont respectées.   6. Post-conditions |
| Tests d’acceptation | Créer un objet niveau en lui passant différentes valeurs en paramètre et vérifier que les valeurs sont respectées. Par la suite, modifier les valeurs et vérifier que les changements sont bien effectués. |
| Complexité | 1 |
| Effort | 1 |
| Commentaires |  |