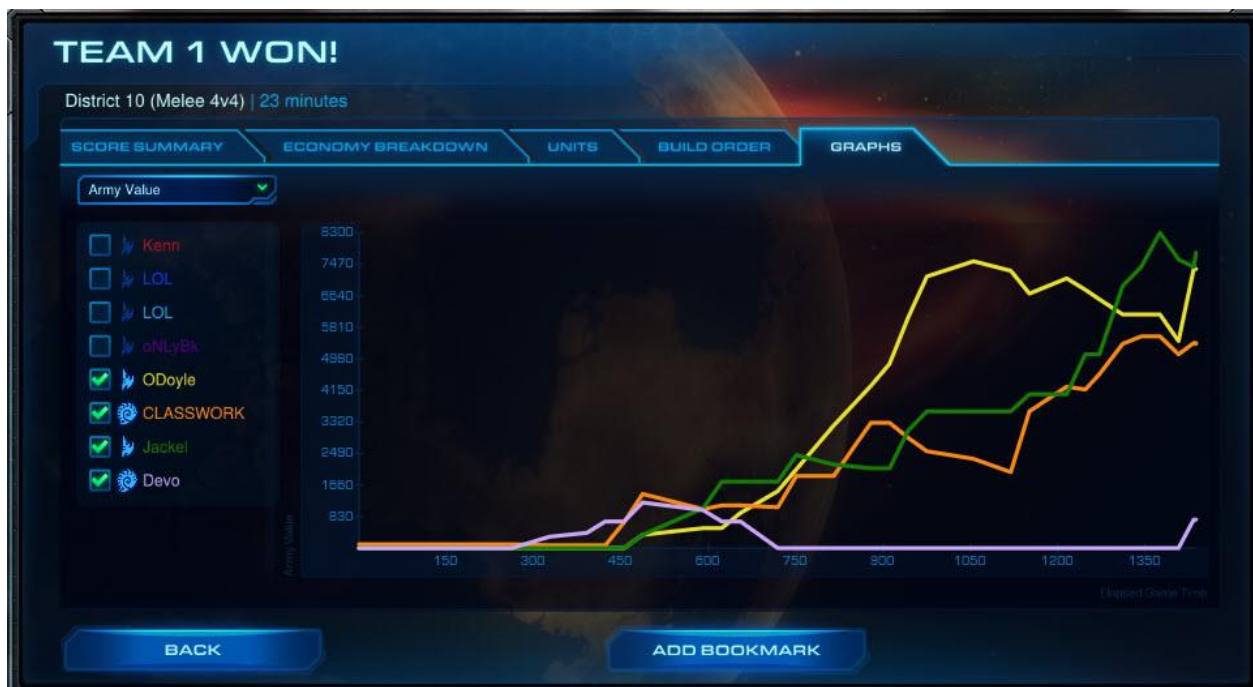


Projet Oracle – Phase 3

Gestion d'utilisateurs et de statistiques



Objectifs

- Ajouter au projet Tankem les concepts de joueurs, d'expériences et de statistiques.
- Modifier Tankem pour supporter ces nouvelles fonctionnalités.
- Concevoir et développer une application permettant de faire la gestion de ces nouveaux éléments :
 - Gestion d'un joueur.
 - Gestion de l'expérience d'un joueur.
 - Consultation des statistiques.
- Concevoir et développer une base de données et ses requêtes effectuant la gestion de ces éléments.

Sommaire

À cette étape du projet, la gestion d'utilisateurs ainsi que la collecte de données sera centrale. Au démarrage de Tankem une présentation personnalisée des deux protagonistes est effectuée suite à leur connexion.

Modalité

Ce travail vaut 40% de la session et doit être fait en équipe de **4 à 5** personnes. Vous pouvez évidemment changer les équipes.

Ateliers

Voici les thématiques des ateliers:

- Atelier 1 : Les propriétés
- Atelier 2 : Design pattern Singleton
- Atelier 3 : Connection à la BD : paresseux versus pressé

On s'attend à ce que les équipes mettent en application ce qui est enseigné à tous les ateliers de la session.

Livrables

Vous devrez remettre le dossier original du jeu mais il contiendra de manière structurée :

- Script SQL nommé `creationTableUtilisateur.sql` qui contiendra les commandes de création des tables des utilisateurs.
- Script SQL nommé `creationTableStatistique.sql` qui contiendra les commandes de création des tables pour les statistiques.
- Utilitaire nommé `populerStatistiques.abc` permettant de créer le script d'insertion des données tests pour l'analyse du système.
- Tankem modifié.
- Vos deux applications de gestion des joueurs.
- Une capture d'écran de l'historique de votre projet GIT à la racine du projet.
- Fichier `RapportTravail.txt` présentant :
 - Le(s) responsable(s) pour chacune des cinq tâches demandées.
 - La proportion relative du travail réalisé de chaque étudiant pour chacune des tâches.

- Le nombre de minutes d'absence non justifiés en classe pour chaque étudiant.
- Commentaires si nécessaire.
- Fichier `LisezMoiPhase3.txt` présentant succinctement chacun des livrables.

Division des tâches

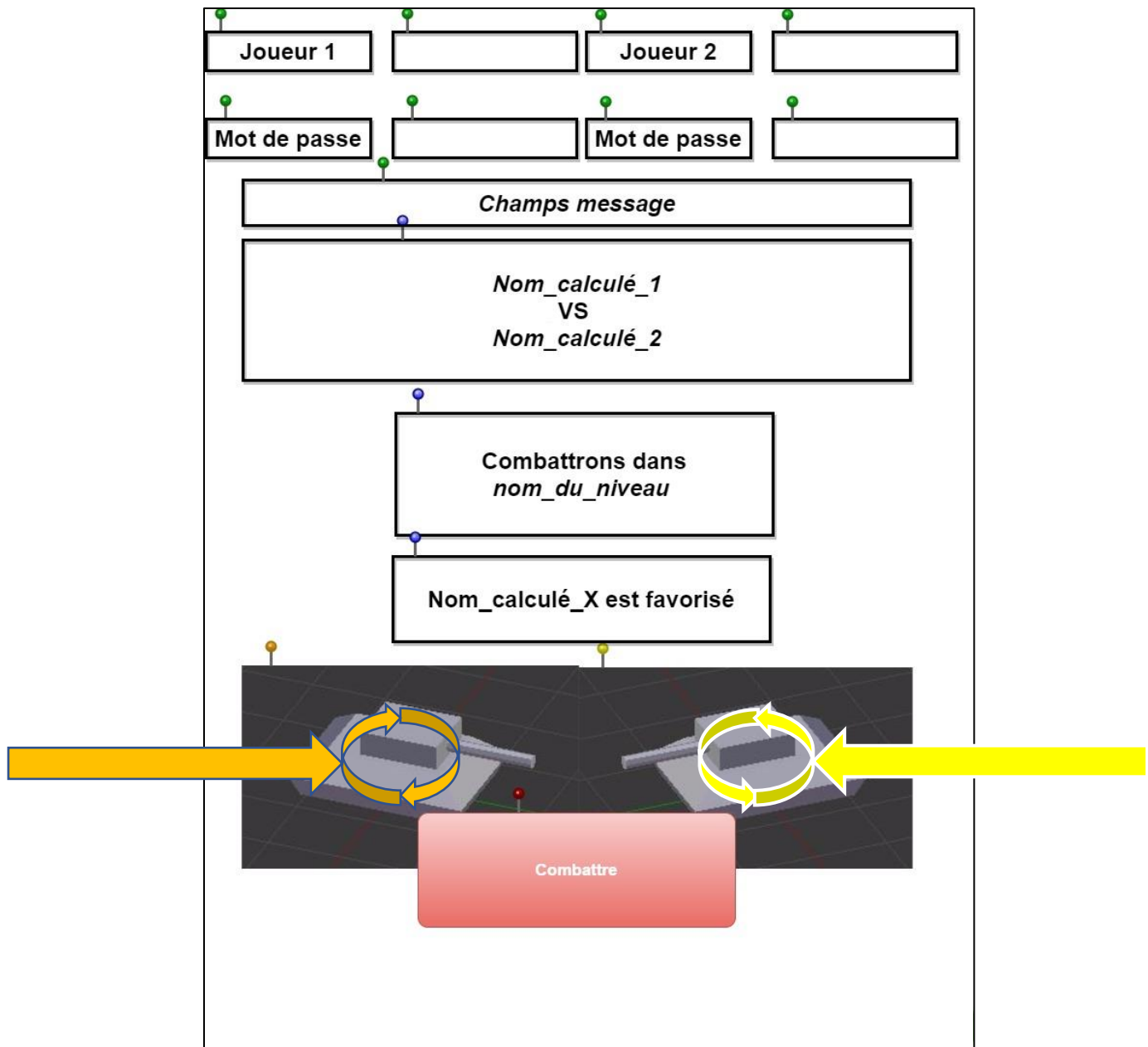
- Base de données (1)
 - Création des tables et autres objets dans la base de données.
 - Création des requêtes préparées, procédures, trigger, ...
 - Utilitaire d'insertion de données fictives.
- Modifications dans Tankem (2)
 - Interface usager
 - Singleton de connexion
 - DAO + DTO
 - Collecte des données
 - Outils de calcul : nom calculé, expérience – niveau – *CoolPoint*, ...
- Logiciels de gestion des joueurs (2)
 - Logiciel 1 : insertion + suppression
 - Interface simple
 - DAO + DTO
 - Logiciel 2 : modification + gestion *CoolPoint* + statistiques :
 - DAO + DTO
 - Interface de connexion du joueur.
 - Interface de consultation et modification des paramètres du joueur.
 - Gestion des *CoolPoints*.
 - Consultation des statistiques de partie.

Modification Tankem – Spécifications techniques

Démarrage






Après le menu de sélection de niveau, vous devrez intégrer une nouvelle fenêtre qui permettra aux joueurs de s'authentifier. Le menu devra permettre, sur la même page, aux deux joueurs de se connecter sur leurs comptes respectifs. On créera ensuite une animation pour présenter les deux joueurs.

Voici une maquette de ce qu'on s'attend.



- **Champs d'authentification (Nom et mot de passe)** : Les joueurs vont entrer leurs informations d'authentification tour à tour. Selon le succès ou l'échec de l'opération, un message sera affiché dans le champ de message. On exige que les joueurs soient enregistrés pour jouer.
- **Champ de message** : ce champ (qui est au centre) va afficher les informations pertinentes selon le succès/l'échec d'une authentification ou l'inaccessibilité à la BD. Le texte est laissé à votre discrétion mais il doit être pertinent.
- **Champs VS** : affiche le nom calculé du joueur 1 vs le nom de joueur calculé du joueur 2 (voir plus loin pour le nom calculé).

- **Champs du favori** : Le joueur ayant le plus haut niveau doit être favorisé et on doit l'indiquer ici. Si les joueurs ont le même niveau, on indique : Aucun favoris.
- **Présentation des chars** : lorsqu'un usager s'authentifiera correctement, on voudra introduire le tank en 3D du combattant avec une animation que vous devez coder. Vous ajusterez la couleur des tanks afin qu'elles correspondent à celles déterminées par les joueurs.
 - Pour le joueur 1, le tank doit effectuer une translation de l'extérieur gauche vers sa position finale. Il doit tourner dans le sens horaire pendant la sélection des caractéristiques.
 - Pour le joueur 2, le tank doit effectuer une translation de l'extérieur droite vers sa position finale et doit tourner dans le sens inverse (i.e. anti-horaire).
 - Voir la maquette et les flèches colorées.
- **Bouton combattre** : Ce bouton va apparaître avec une animation lorsque les deux joueurs seront authentifiés.
 - Il devra avoir une animation de grossissement/rapetissement continu pour être visible et pour monter la tension. Quoique non essentielle, une animation suivant un rythme de battement cardiaque est souhaitée.
 - Lorsqu'on appuiera sur le bouton, la séquence suivante devra s'effectuer : les deux chars qui tournent devront tourner jusqu'à ce qu'il se mette face à face et un texte Nom_Joueur_1 Vs Nom_Joueur_2 devra apparaître en gros et être animé en continu (à votre choix). Ensuite on fera une transition vers le jeu.

En résumé : les items avec un  doivent être présent dès l'ouverture de la fenêtre. Lorsque le joueur 1 est correctement authentifié, les éléments  apparaissent. Lorsque le joueur 2 est correctement authentifié, les éléments  apparaissent. Les  et  apparaissent quand les deux joueurs sont authentifiés.

Modification du déroulement d'une partie

La couleur des chars doit être conforme à ce qui est déterminé pour les joueurs.

Les caractéristiques liées à la balance de chaque joueur doivent être ajustées pour les quatre paramètres indiqués : vie, force, agilité et dextérité. Voir la section décrivant la gestion de l'expérience pour plus de détail.

Fin de la partie

À la fin d'une partie, on récompense les deux joueurs avec de l'expérience.

Considérant :

e , l'expérience gagnée

f , si le joueur gagnant est non favori $f = 1$, sinon $f = 0$

V_g , le nombre de vies restantes au joueur gagnant

V_p , le nombre de vies perdues au joueur gagnant

- On félicite le gagnant en lui donnant plus d'expérience selon le calcul suivant :

$$e = 100 + 100f + 2V_g$$

- Le joueur perdant gagne aussi de l'expérience selon le calcul suivant :

$$e = 2V_p$$

À la fin de la partie, on ajoute aux joueurs leurs nouvelles expériences et on affiche la phrase du joueur gagnant, un sommaire du calcul de l'expérience ainsi que le total, le tout avec une animation au choix. Vous avez la liberté de la présentation mais ce doit être clair et visuellement intéressant.

Si l'un ou l'autre des joueurs augmente de niveau, on doit le signaler avec une animation au choix. On vous recommande de vous créer une classe (`AnalyseFinPartie` par exemple) dans laquelle vous allez isoler le calcul de l'expérience, la détermination si le niveau est augmenté, etc. Pour le calcul du changement de niveau, consulter la rubrique un peu plus loin.

Collecte de données

Chaque partie devra avoir un résumé dans la base de données. Entre autres, on devra enregistrer :

- Date et heure du début et de la fin de partie.
- Les joueurs participants.
- Le niveau joué.
- Le joueur gagnant.
- Pour chaque joueur :
 - Pour chaque arme :
 - Combien de fois on a tiré au total
 - Le nombre de coups tirés qui ont atteint leur cible.
 - Le nombre de coups reçus.
 - La distance parcourue.

Expérience et niveau d'un joueur

Un concept de niveau est ajouté afin de permettre au joueur de modifier les caractéristiques de son tank.

Pour chaque niveau atteint, 5 *CoolPoint* sont attribués et le joueur peut les distribuer à chacun des quatre promotions disponibles. Chacun de ces paramètres modifie un aspect de la balance :

- Vie : 5% plus de vie par *CoolPoint*
- Force : 5% de plus de dégât par *CoolPoint*
- Agilité : 2.5% sur la vitesse du tank par *CoolPoint*
- Dextérité : gain de 2.5% progressif sur le rechargement des armes par *CoolPoint*

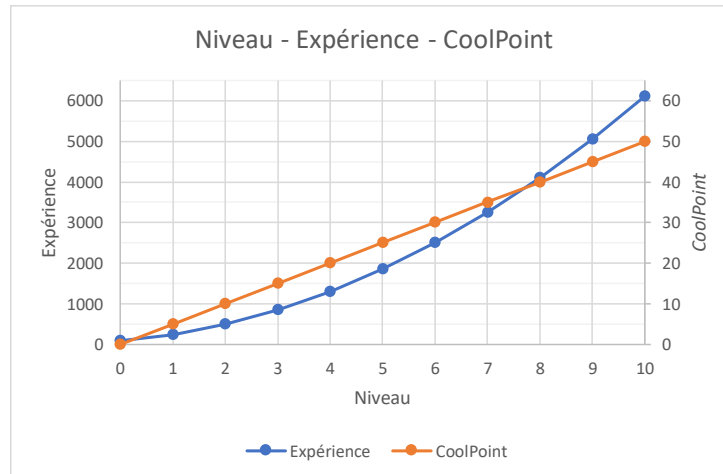
Voici la formule qui permet de déterminer si un joueur change de niveau et la relation avec les *CoolPoint*.

$$s = 100(n + 1) + 50n^2$$

$$\text{CoolPoint} = 5n$$

où s , est le seuil du niveau d'expérience requis pour passer au niveau suivant

n , est le niveau courant du joueur



L'attribution des *CoolPoint* doit se faire par le logiciel de gestion.

Nom de joueur calculé

Selon la manière dont un joueur distribue ses points, on va lui attribuer un nom de joueur qui doit être calculé. Le nom de joueur calculé (ou nom calculé) est son nom auquel on ajoute des qualificatifs selon ses caractéristiques.

Voici la structure d'un nom calculé : « Nom_Joueur qualificatifA qualificatifB ». Les règles pour choisir les qualificatifs sont :

1. Choisir le qualificatif A selon le nombre de points de l'attribut ayant le plus de points.
2. Choisir le qualificatif B selon le nombre de points de l'attribut ayant le deuxième plus grand nombre de points.
3. Si dans 1) ou 2), deux attributs ont le même de points, on en choisit un au hasard comme qualificatif A et l'autre comme qualificatif B.
4. Si un joueur a des points dans un seul attribut, il a seulement le qualificatif A.
5. Exception : un joueur avec le maximum de points dans tous les attributs aura le qualificatif : *Dominateur*.

Attributs	Critère	Qualificatif A	Qualificatif B
Vie	>= 1	le fougoux	fougoux
	>= 5	le pétulant	pétulant
	>= 10	l'immortel	immortel
Force	>= 1	le crossfiter	qui fait du crossfit
	>= 5	le hulk	brutal
	>= 10	le tout puissant	Marc-André
Agilité	>= 1	le prompt	prompt
	>= 5	le lynx	lynx
	>= 10	le foudroyant	foudroyant
Dextérité	>= 1	le précis	précis
	>= 5	l'habile	habile
	>= 10	le chirurgien	chirurgien

On vous recommande de créer une classe (`AnalyseDTOJoueur` par exemple) dans laquelle vous allez créer une fonction qui prend en paramètre votre `DTOJoueur` et qui retournera le nom calculé.

Singleton de connexion

Vous êtes responsable de fournir une classe de singleton afin que tous les DAO de Oracle puissent se connecter avec une connexion unique. Les explications nécessaires seront fournies durant l'atelier sur le sujet.

Logiciels de gestion du joueur – Spécifications techniques

Vous devez concevoir et réaliser deux logiciels qui permettront la gestion d'un joueur.

Paramètres pour chaque joueur

Le concept de joueur prend une place importante pour la troisième phase. Les informations à gérer sont:

- le nom du joueur (un alias unique)
- un mot de passe (la sécurisation n'est pas nécessaire pour ce prototype)
- la couleur de son tank
- une phrase « punch » qui sera affichée lorsque le joueur aura gagnée
- toutes les données nécessaires à l'attribution de ses CoolPoints.

Gestion explicite des joueurs

On vous demande de faire un logiciel simple de gestion explicite des joueurs. C'est-à-dire, que les tâches d'ajout et de suppression sont à réaliser par ce petit utilitaire.

Vous devez créer un logiciel Python en mode console et qui interagit simplement par des questions au clavier. Les deux fonctions suivantes sont à réaliser :

- Ajouter joueur : (...)
- Supprimer joueur : (...)

Gestion des paramètres du joueur, de ses CoolPoint et de l'analyse de parties

Votre logiciel doit offrir les interfaces usager permettant de couvrir ces tâches :

- La connexion d'un joueur.
- L'édition d'un joueur :
 - Consultation et modification des caractéristiques du joueur :
 - La couleur de son tank (modifiable)
 - Phrase personnelle (modifiable)
 - L'expérience du joueur (non modifiable)
 - Le niveau du joueur (non modifiable)
 - La date et l'heure de la dernière partie jouée (non modifiable)
- L'attribution des CoolPoints :
 - La capacité, pour le joueur sélectionné, d'attribuer ses *CoolPoint* aux différentes classes de promotion (vie, force, agilité et dextérité).
 - Un joueur peut en tout temps reprendre ses *CoolPoint* pour les attribuer autrement.

- L'analyse de parties :
 - La capacité de choisir dans la liste des parties jouées l'une d'entre elle et de consulter toutes les informations de cette dernière.
 - La capacité de voir un bilan de toutes les parties jouées en incluant, entre autres, les statistiques suivantes :
 - Quelle est l'arme préférée du joueur + sous forme de graphique, la répartition de l'usage de chacune des armes.
 - Quelle est l'arme la plus efficace du joueur.
 - Le nombre total de parties jouées, gagnées, perdues + sous forme de graphique, la répartition de cette information.
 - Sous forme d'un graphique de type « ligne du temps », la densité de partie jouées dans le temps.
 - Une autre statistique de votre choix.

La disposition de l'interface usager est laissé à votre discrétion. Néanmoins, sachez que des points liés à l'ergonomie sont considérés (très peu pour l'esthétisme).

Vous devez créer et ajouter des données fictives dans la base de données pour effectuer des tests pertinents. Sont attendu au moins trois joueurs fictifs qui ont au moins fait 250 parties les uns contre les autres. Assurez-vous que ces données génèrent des statistiques soient pertinentes à l'analyse. Attention, pour cette partie, vous devrez créer un utilitaire générant un script d'insertion des données aléatoires mais cohérentes. Cet utilitaire doit être paramétrables et « intelligent ».

Vous pouvez utiliser n'importe quel outil de développement pour réaliser ce logiciel (en autant qu'il vous ait été enseigné).

Politique d'évaluation

Elles sont analogues à la phase 1 et 2:

- N'oubliez pas que les absences vont compter si vous en abuser.
- Si un membre sous-performe/surperforme de manière importante (environ 20% de différence par rapport aux autres) par rapport à son groupe, un ajustement individuel sera considéré à la discrétion de l'enseignant.