420-440-SF
PROGRAMMATION D'APPLICATIONS
TRAVAIL PRATIQUE #2
« MANIPULATION DE POLYGONES »

- Compte pour 15% de la session.
- Travail à réaliser en équipe de 3.
- Une remise (voir calendrier).

Objectifs pédagogiques

Ce deuxième travail vise à développer les compétences suivantes :

016S	Exploiter un langage de programmation structurée
0170	Exploiter des données sur fichiers
0171	Corriger des programmes
0177	Assurer la qualité d'une application

Contexte

En partant du code de l'application **PolyWinSDL** (laboratoire 5), vous devez créer une application qui exécute une série de manipulations (translation et rotation) sur des dessins (bonhomme, auto, maison, etc.). Un dessin peut être composé de toutes les formes qui héritent de la classe ***Shape*.**

Spécifications et contraintes de réalisation

L'application

- La cadence est de 30 images par seconde.
- Le remplissage des formes n'est pas à faire.
- Les manipulations possibles sont la translation et la rotation (voir **Annexe 1**).
- Les dessins sont stockés dans un fichier XML nommé **shapes.XML** (voir fichier exemple shapes.xml).
- Les manipulations sont stockées dans un fichier XML nommé **manipulations.XML** (voir fichier exemple manipulations.xml).
- Voir **Annexe 2** pour un exemple de *main.*

Le diagramme de classes

- Le diagramme de classe se trouve dans le fichier **diagrammesUML.simp** (s'ouvre avec https://www.softwareideas.net/)
- Les classes du diagramme doivent se retrouver dans le code.
- Il est possible de ne pas suivre à la lettre le diagramme de classe. Toute modification aux diagrammes doit être acceptée par votre professeur.
- **Les classes en jaune** indiquent **un ajout** par rapport au laboratoire 5.
- **Les classes en bleu** indiquent **des modifications** à faire par rapport au laboratoire 5. Toutes modifications au code impliquent aussi des modifications des tests unitaires.

La gestion des erreurs

- La gestion des erreurs doit se faire **avec les exceptions**.

Les libraires externes

- La lecture des fichiers XML peut se faire avec PugiXML ou tout autre libraire (voir exemple).
- SDL est la librairie graphique à utiliser pour ce travail.
- Favoriser BOOST pour tous les traitements de chaines de caractères.

Les tests unitaires et tests d'intégration

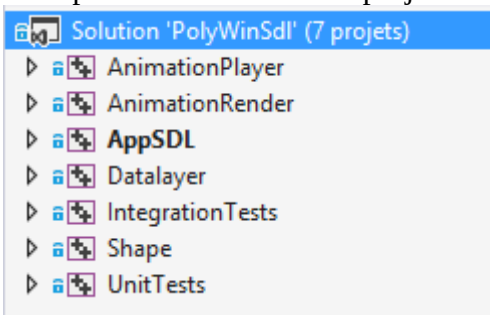
- La majorité du code doit être couvert par les tests.
- Le code de l'application sur GitHub doit toujours être couvert par des tests.

GitHub

- Utiliser GitHub de façon régulière.
- Les commentaires sur GitHub doivent être pertinents.
- La solution doit toujours être compilable.

Structure de la solution

- La solution VS2013 doit contenir les 7 projets suivants (voir document **creationProjets.docx** pour exemples de la création des projets **IntegartionTests** et **DataLayer**).



- Les références à chacun des projets sont les suivantes (voir document **creationProjet.docx**):

Projet	**Références**
AnimationPlayer	AnimationRender
AnimationRender	Shape
AppSDL	AnimationPlayer, AnimationRender, DataLayer, Shape
DataLayer	AnimationRender
IntegrationTests	DataLayer, Shape
Shape	Aucune
UnitTests	AnimationRender, Shape

Calendrier des cours avant la remise

Sem	**Groupe lundi – mercredi**
8	24 mars : temps TP
	27 mars : temps TP
9	31 mars : temps TP
	3 avril : congé de Pâques
10	7 avril : temps TP
	10 avril : temps TP
11	*14 avril : Encadrement et activités périscolaires*
	17 avril : **remise TP2 fin de journée**

Fonctionnalités et réalisations additionnelles - Bonus

Toutes fonctionnalités ou réalisations additionnelles à l'application sont susceptibles d'obtenir des points en bonus, pour un maximum de 10 points (selon la difficulté) et de 100 points de la note totale.

Toutefois, ces fonctionnalités ne seront considérées que si les spécifications de l'application ont été respectées et réalisées selon les critères de qualité exigés : l'évaluation du travail devra au préalable obtenir une cote supérieure ou égale à B.

Quelques suggestions (non exhaustives) :

- Ajouter une manipulation (par exemple le redimensionnement).
- Ajouter des événements sonores à l'animation.
- Gérer le remplissage des polygones.

Évaluation

Tests unitaires et tests d'intégration Qualité des tests Couverture de tests Choix et réalisation des jeux d'essais pour les tests d'intégration	40%
Qualité du code Noms des méthodes et attributs significatifs Clarté et simplicité Respect des standards Gestion des erreurs Voir document « **liste de vérifications.pdf** »	30%
Fonctionnalités Le programme est entièrement fonctionnel Respect des contraintes Il n'y a aucune erreur à l'exécution. Il n'y a aucun bogue.	20%
Professionnalisme Avancement régulier du projet sur GitHub Qualité des commentaires sur GitHub Le projet est toujours compilable sur GitHub Le code de l'application sur GitHub doit toujours être couvert par des tests	10%

Le travail doit être reparti également entre chacun des coéquipiers. Un coéquipier qui participe moins pourra se faire retrancher des points. La participation sera suivie en classe et par les remises sur GitHub.

À noter : Si le professeur le juge à propos, toute étudiante ou tout étudiant pourra être convoqué à une rencontre d'évaluation pour vérifier son degré d'acquisition des connaissances et d'appréhension de la solution proposée.

Documents à remettre

Remise du travail sur GitHub.

C’est la dernière version déposée sur GitHub (avant la date et l’heure de remise) qui sera corrigée.

Si nécessaire, indiquez dans un **fichier texte nommé « README.txt»**:

a. Les éléments non fonctionnels du projet
b. Les réalisations additionnelles (bonus)

Annexe 1

Manipulations mathématiques sur un point

La translation
Pour effectuer une translation, vous n'avez qu'à additionner les valeurs relatives pour obtenir la nouvelle coordonnée.

Par exemple, si la coordonnée initiale est (3, 5) et que vous effectuez une translation de (-4, 7), la nouvelle coordonnée sera (-1, 12)

La Rotation (à partir de l'origine)
Pour effectuer une rotation, vous devez effectuer une multiplication par la matrice de rotation suivante :

Cos A	-Sin A
Sin A	Cos A

Où A est l'angle désiré.

Par exemple, supposons que vous vouliez effectuer une rotation de 90 degrés, la matrice devient :

0	-1
1	0

Pour effectuer une multiplication de coordonnées (X, Y) par une matrice de rotation, vous devez effectuer le calcul suivant :

(X * Cos A + Y * Sin A, X * -Sin A + Y * Cos A)

Par exemple, si le point initial est (2, 3) et que vous désirez effectuer une rotation de 90 degrés, le nouveau point sera:

(2 * 0 + 3 * 1, 2 * -1 + 3 * 0) = (3, -2)

Pour faire une rotation à partir d'un centre de rotation autre que l'origine :

1) Faire une translation en soustrayant le centre de rotation
2) faire la rotation à partir de l'origine
3) Faire une translation en ajoutant le centre de rotation

Références : http://www.games-creators.org/wiki/Introduction_aux_matrices

Annexe 2

Exemple de code qui devrait se retrouver dans le main

```
IWindowAPI * windowAPI = new SDLWindowAPI(640, 480, "Afficher des formes");

IWindowRender * windowRender = new WindowsRender(*windowAPI);

IAnimatedShapeRepository * animatedShapeRepository
   = new AnimatedShapeRepository("../TestFiles/shape/composites.xml", *windowAPI);

IManipulationRepository * manipulationRepository
   = new ManipulationRepository("../TestFiles/manipulation/manipulations.xml");

AnimatedShapePlayer * animation
   = new AnimatedShapePlayer(*windowRender, * animatedShapeRepository , *manipulationRepository);

animation->load();
animation->run();


delete shapeRepository;
delete manipulationRepository;
delete animation;
delete windowAPI;
delete windowRender;
```