

Scale for project Nibbler

Git repository

Introduction

Nous vous demandons, pour le bon déroulement de cette notation :

- De rester courtois, poli, respectueux, constructif lors de cet échange. Le lien de confiance entre la communauté 42 et vous en dépend.
- De bien mettre en évidence auprès de la personne notée (ou du groupe) les dysfonctionnements éventuels.
- D'accepter qu'il puisse y avoir parfois des différences d'interprétation sur les demandes du sujet ou l'étendue des fonctionnalités. Restez ouvert d'esprit face à la vision de l'autre (a-t-il raison ou tort ?), et notez le plus honnêtement possible.

Bonne soutenance à tous !

Guidelines

RAPPELEZ VOUS QUE VOUS NE DEVEZ CORRIGER QUE CE QUI SE TROUVE SUR LE DÉPÔT DE RENDU DE L'ÉTUDIANT.

Il s'agit de faire un "git clone" du dépôt de rendu, et de corriger ce qui s'y trouve.

Si le correcteur n'a pas encore fait ce projet, il est obligatoire de lire le sujet en entier avant de commencer cette soutenance.

Ratings

Define the type of error (if there is an error), which ended the correction.

Ok	Empty work	Incomplete work	Wrong author file	Invalid compilation	None	Cheat	Crash
----	------------	-----------------	-------------------	---------------------	------	-------	-------

Attachments

Subject Intra - Elearning - Subnotions

Sections

Préliminaires

Consignes préliminaires

Vérifiez d'abord les éléments suivants :

- Il y a bien un rendu (dans le dépôt git)
- Pas de triche, les deux étudiants doivent pouvoir expliquer leur code.
- Vous ne devez pas trouver sur le depot de (sources de) bibliotheques que le groupe n'a pas ecrites lui meme. Cela signifie que n'importe quelle bibliotheque exterieure utilisee dans le projet ne doit pas etre presente sur le depot. Cela inclu bien entendu les bibliotheques graphiques, mais aussi des bibliotheques utilitaires comme Boost, etc. Voir le sujet.
- Le groupe corrigé doit fournir un script ou un ensemble de scripts permettant une installation simple et efficace des dependances conformément au sujet. Le groupe corrigé est donc seul responsable de la mise en place de son environnement d'evaluation sur le systeme utilise pour la correction quelque soit l'etat de celui-ci.
- Le correcteur est seul juge pour determiner si cette phase d'installation est trop longue ou inadaptée et peut decider unilateralement de mettre un terme a la soutenance prematurement en flagant le rendu en "incomplet work". Un grand pouvoir implique de grandes responsabilites. Don't be a dick, mais soyez pas naifs non plus.
- Dans l'exécutable principal, le groupe corrigé **NE DOIT PAS**

avoir fait la MOINDRE référence à une bibliothèque d'affichage précise (comme OpenGL, SDL, etc). Cela serait un hors sujet pur et simple. Cela inclut non seulement les primitives d'affichage, mais aussi les primitives de gestion d'évènements. Par exemple la présence de types Qt ou SDL, ou carrément des appels à des fonctions de ces bibliothèques.

- La boucle de jeu doit IMPERATIVEMENT se trouver dans le programme principal et surtout pas dans les bibliothèques. Toutefois, certaines bibliothèques peuvent imposer leur propre boucle, dans ce cas, cette boucle DOIT être manipulée pour être synchronisée avec la boucle du corps et en dépendre entièrement. Le contraire est hors sujet, flagguer "incomplet work". Voir le sujet et plus bas dans ce barème en cas de doute.

- Un makefile comportant les règles habituelles doit permettre de compiler l'exécutable principal ainsi que les 3 bibliothèques dynamiques.

- Si le rendu comporte moins de 3 bibliothèques dynamiques utilisées à l'exécution, flagguer "incomplet work".

Si un élément de cette liste n'est pas respecté, la notation s'arrête là. Utilisez le flag approprié. Vous êtes encouragés à continuer de débattre du projet, mais le barème n'est pas appliqué.

Yes

No

Tests de fonctionnalité

Lancement

L'exécutable principal permet au moins de choisir la taille de la surface de jeu, et choisir une bibliothèque dynamique pour afficher le jeu. Le jeu s'exécute alors normalement.

Yes

No

Erreurs de lancement

Les erreurs au lancement doivent être gérées. Le minimum à gérer est :

- Nombre d'arguments insuffisant.
- Mauvaise largeur / hauteur (nombres négatifs, trop petits, trop grands, pas des nombres ...)

- ...

Il n'est pas acceptable de se contenter de quitter bêtement le programme sans donner de feedback.

Yes

No

Déplacement

Le serpent avance tout seul, et on peut le faire changer de direction à l'aide du clavier. Ce comportement est uniforme dans les 3 bibliothèques rendues.

Yes

No

Nourriture

Le serpent peut manger un "fruit", et grandit ce faisant. Ce comportement est uniforme dans les 3 bibliothèques rendues puisqu'elles sont indépendantes de la logique du jeu.

Yes

No

Défaite

La partie est terminée si on touche un mur ou le corps du serpent. Ce comportement est uniforme dans les 3 bibliothèques rendues puisqu'elle sont indépendantes de la logique du jeu.

Yes

No

Fin volontaire de partie

On peut quitter volontairement la partie en appuyant sur Echap ou tout autre moyen conventionnel. Ce comportement est uniforme dans les 3 bibliothèques rendues.

Yes

No

Changement de GUI

On peut changer volontairement de GUI à l'aide des touches f1, f2 et f3 conformément au sujet. Cette fonctionnalité étant

centrale dans le projet, assurez-vous bien qu'elle soit stable. Verifiez a differents moments de l'execution, dans un ordre quelconque, en switchant rapidement de l'une a l'autre, etc...

Yes

No

Implementation

On rappelle que la boucle de jeu et donc la logique du jeu doit être située dans l'exécutable principal tel que demande dans le sujet. AUCUNE exception n'est acceptable. Si une bibliothèque impose sa propre boucle, alors cette boucle doit être synchronisée avec la boucle de l'exécutable principal. Oui, c'est délicat et oui cette partie du code doit être réfléchie. Mais cela implique surtout que cette bibliothèque graphique n'était probablement pas adaptée à ce projet. La liberté impose de faire des choix...

Boucle de jeu cohérente

La boucle de jeu devrait ressembler à ceci :

```
while (...)  
{  
    traitementEvenements();  
    miseAJourDonneesDeJeu();  
    miseAJourAffichage();  
}
```

C'est le cas ?

Yes

No

Gestion d'événements

La gestion des événements clavier doit être fonctionnelle et propre. Une forêt d'ifs n'est PAS acceptable ici.

Yes

No

Interface core/bibliothèques

Le but principal du projet est double. D'une part, il s'agit de désolidariser les aspects graphiques/inputs du programme en les relocalisant dans une bibliothèque dynamique. Le rendu doit proposer l'utilisation de 3 bibliothèques graphiques différentes. D'autre part, il s'agit d'interagir de manière uniforme avec l'une ou l'autre des bibliothèques dynamiques créées par le groupe.

Les étudiants doivent avoir prévu une interface intelligemment conçue pour interagir avec leurs modules d'affichage. Un minimum serait une interface ressemblant grossièrement à la suivante :

```
class INibblerDisplay
{
public:
virtual void init(...) = 0;
virtual void getEvents(...) = 0;
virtual void updateGameData(...) = 0;
virtual void refreshDisplay(...) = 0;
virtual void stop(...) = 0;
};
```

On devrait y trouver de quoi initialiser la bibliothèque graphique contenue dans la bibliothèque dynamique, de quoi notifier la bibliothèque de l'état du jeu pour affichage, de quoi récupérer les inputs utilisateur, etc. Conformément aux consignes de la partie préliminaire, je vous rappelle qu'aucune référence d'aucune sorte à l'une ou l'autre des bibliothèques graphiques ne doit exister dans le programme principal, et par conséquence, dans cette interface.

Est-ce qu'on a bien une interface adaptée dans le programme principal qui permet de communiquer uniformément avec au moins les trois bibliothèques dynamiques encapsulant les trois bibliothèques graphiques choisies ?

Yes

No

Plug-ins d'affichage objets / Design coherent

Les bibliothèques dynamiques de Nibbler peuvent donc être vues comme des plug-ins implementant une interface particulière (c'est à dire un "contrat" exposant leurs fonctionnalités disponibles pour le programme utilisateur). Cela sous-entend par définition que le code de l'exécutable principal et des bibliothèques dynamiques soit en C++ conformément au sujet. Toutefois, la communication entre un programme et des bibliothèques dynamiques est en C. Cela implique donc d'utiliser une construction syntaxique spéciale "extern C" afin de renseigner le compilateur C++ qu'une petite partie du code est en C et non en C++. Ces petites fonctions C sont appelées "points d'entrée" d'une bibliothèque.

Les points d'entrée peuvent renvoyer des infos génériques sur la

bibliothèque afin d'assurer sa compatibilité avec le programme principal, mais il doit OBLIGATOIREMENT exister un point d'entrée permettant de récupérer une instance de la classe d'affichage contenue dans la bibliothèque dynamique sous forme d'un pointeur sur l'interface dont nous avons parlé dans la section précédente. Dans le cas contraire, le rendu est à côté de la plaque.

Exemple naïf mais parfaitement recevable:

```
extern "C"  
{  
INibblerDisplay* getDisplayModule() { return new NibblerDisplayOpenGL(); }  
}
```

Les points d'entrée doivent être des fonctions très courtes se contentant généralement de renvoyer une donnée, telle qu'une nouvelle instance. Si vous constatez qu'un traitement plus long est fait dans un point d'entrée, cette question n'est pas validée.

Vérifier que les trois bibliothèques fonctionnent de la manière décrite ci-dessus.

Yes

No

Gestion d'erreur des bibliothèques dynamiques

Vérifier que le groupe a prévu une gestion des erreurs possibles au chargement, à l'utilisation et au déchargement des bibliothèques dynamiques à base d'une exception. Vous devez refuser les gestions d'erreurs qui ne sont pas à base d'exceptions, ainsi que les exceptions scalaires.

Yes

No

Le groupe

Approche du projet par le groupe

Nibbler est un projet à réaliser en binômes. Sa complexité relative en fait un candidat idéal pour entamer une approche réfléchie de la répartition du travail.

- 5 étoiles -> Les deux étudiants ont participé de manière active au développement et à la soutenance. Tous deux sont incollables aussi bien sur le fonctionnement général du projet

que sur les detail d'implementation de la partie de leur binome (par exemple concernant les bibliotheques graphiques).

- 4 etoiles -> Les deux etudiants ont participe de maniere active au developpement et connaissent bien leurs parties respectives, mais ne connaissent pas la partie de leur binome en details.

- 3 etoiles -> Pas utilise sur ce bareme.

- 2 etoiles -> Un etudiant est tres en retrait par rapport a l'autre, aussi bien que sur le travail realise que sur la soutenance, mais il semble visiblement avoir participe a hauteur de ses moyens.

- 1 etoile -> Pas utilise sur ce bareme.

- 0 etoiles -> Vous pensez qu'un seul etudiant sur les deux a effectue tout le travail ou presque (quelqu'en soit la raison), quelque soit la qualite du dit travail.

Rate it from 0 (failed) through 5 (excellent)



Bonus

Sons

Il y a une (ou plusieurs) bibliothèque de sons, et elle est gérée de la même manière que les bibliothèques graphiques. c'est-à-dire via une bibliothèque dynamique que l'étudiant a créée, qui est loadée et utilisée au runtime.

Yes

No

Multiplayer

On a deux ou plus snakes qui se battent pour la nourriture ?!
(un mode single player doit rester possible)

Yes

No

Jeu en réseau

L'étudiant a fait un jeu bad ass - multiplayer et en réseau (le bonus precedent est valide aussi)?

Yes

No

D autres bonus

Les étudiants ont ajouté à leur projet d'autres fonctionnalités améliorant l'expérience de jeu ? Comptez-les ici, vous pouvez en noter jusqu'à 5 distinctes.

Exemple de fonctionnalité bonus valide :

- Un système de score
- Des bonus / malus
- Une tête se démarquant graphiquement du reste du corps
- Une vitesse de déplacement de plus en plus rapide
- Des obstacles sur le plateau de jeu
- etc ...

Rate it from 0 (failed) through 5 (excellent)



Conclusion

Leave a comment on this correction

* Comment

....