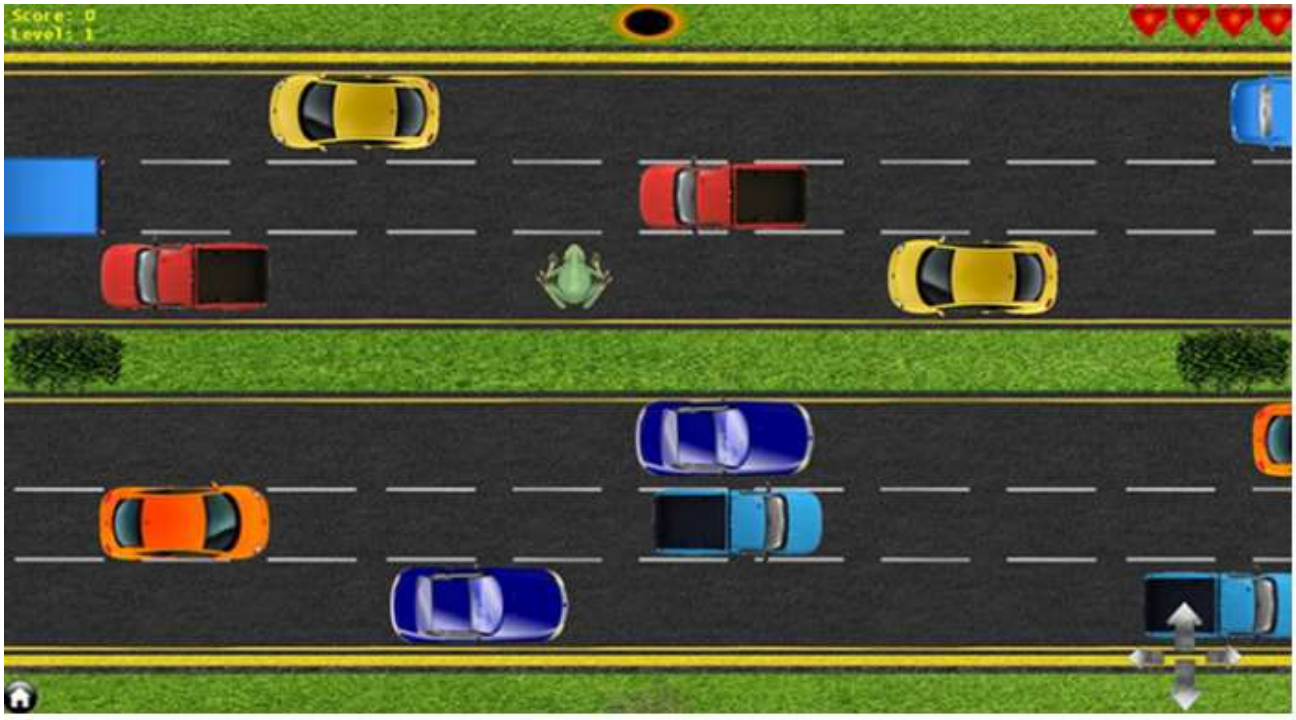
Projet Java III - 2021-2022

Frogger

Introduction :

Frogger est un jeu vidéo d'arcade développé par Konami et sorti en 1981. Il est généralement considéré comme un classique de ce média. Le but du jeu est de diriger une grenouille jusqu'à sa maison. Pour cela, le joueur doit d'abord traverser une route en évitant des voitures qui roulent à différentes vitesses puis une rivière aux courants changeants en passant d'objets en objets. La grenouille meurt si elle touche une voiture ou si elle tombe dans la rivière. Elle peut attraper des mouches bonus au passage.



Le projet de cette année consiste en le développement de votre propre version du jeu Frogger. Dans ce jeu, une grenouille doit traverser en toute sécurité une autoroute sur laquelle roulent des voitures. L'autoroute se compose d'un certain nombre de voies, d'une berne centrale et de deux voies supplémentaires en haut et en bas où la grenouille est en sécurité.

Sur la berne centrale il y a de l’herbe où la grenouille peut s'arrêter, mais sur l'autoroute tout contact avec une voiture qui passe est fatal. Sur la berne centrale il y a également des arbustes qui bloquent le passage de la grenouille. Au début du jeu, la grenouille obtient un certain nombre de vies. Le jeu se termine lorsque toutes les vies de la grenouille ont été épuisées. Dispersés à travers les voies, il y a des pièces de monnaie (ou coins) et des insectes qui peuvent être ramassés et rapporter des points au joueur. Il peut aussi y avoir une pilule qui rend Frogger temporairement invincible.

Le jeu se compose de plusieurs niveaux. Lorsque la grenouille atteint en toute sécurité l'autre côté d'un niveau et a collecté toutes les pièces du niveau, le niveau suivant est chargé. Après chaque niveau, la difficulté augmente : par exemple parce qu'il y a plus de voitures, les voitures se déplacent plus vite, il y a plus de voies, ou parce qu'il y a plus de buissons.

Description des éléments du jeu :

Le terrain de jeu :

Le terrain de jeu se compose d'une autoroute elle-même composée de voies et d'une réserve centrale, toutes les voies sont affichées horizontalement. Dans le terrain de jeu, la grenouille peut se déplacer et interagir avec les autres éléments détaillés dans cette section.

La grenouille :

Le joueur contrôle une grenouille nommée Frogger. Au début de chaque niveau, Frogger commence tout en bas de l’écran, sur une bande où aucune voiture ne peut aller, et où Frogger est en sécurité.
Le but du jeu est d'amener Frogger en haut de l'écran. Dès que Frogger est en haut de l'écran, le niveau est terminé et le niveau suivant est chargé.
Frogger peut se déplacer dans deux directions : verticale (haut-bas) et horizontale (gauche-droite). Dans la direction verticale, il se déplace de bande en bande : l'appui sur une touche de direction le fait passer d'une voie inférieure à une voie supérieure (ou sur la berne centrale).
Dans le sens horizontal, Frogger peut se déplacer librement sur toute la bande : quand une touche de direction horizontale est appuyée Frogger se déplace de quelques pixels vers la gauche ou de quelques pixels vers la droite.

Les voies de circulation :

Dans le jeu, il y a des voies de circulation (ou bande de circulation) où les voitures roulent. Frogger doit utiliser les voies pour se rendre de l'autre côté de l’écran de jeu. Chaque voie a une direction dans laquelle les voitures se déplacent. En plus des voitures, des pièces de monnaie et des insectes peuvent également être trouvés sur les voies.

La berne centrale :

La berne centrale (ou réserve centrale) est une bande sur laquelle aucune voiture ne circule. Elle peut toutefois contenir des arbustes qui sont des obstacles pour Frogger.

Les pièces :

Des pièces sont dispersées dans le terrain de jeu. Lorsque Frogger ramasse une pièce, le joueur gagne des points. Avant que le niveau puisse être terminé avec succès, toutes les pièces doivent être collectées.

Les insectes :

En plus des pièces de monnaie, il y a aussi des insectes sur l'autoroute. Il existe 3 types d'insectes différents. Chaque type d'insecte a une valeur différente qui donne des points supplémentaires pour Frogger. Contrairement aux pièces de monnaie, tous les insectes ne doivent pas être collectés. Ils servent de bonus au joueur pour gagner des points supplémentaires.

La pillule :

Dans chaque niveau on peut trouver une pillule. Lorsque Frogger touche une pilule, il devient temporairement invincible et ne peut donc pas être tué par les voitures qui passent pendant un certain laps de temps. Tant que l'effet de la pilule est activé, Frogger a une couleur différente. Pour garder le jeu équilibré, la vitesse de Frogger est plus lente quand il est invincible.

Les voitures :

Le jeu comprend **4 types de voitures différents qui doivent être présents** dans chaque niveau. Chaque type de voiture a sa propre couleur et son propre comportement. Il existe un type **rouge** qui changera de bande pour poursuivre Frogger, tandis qu'un autre type (**bleu**) ralentira lorsque Frogger est dans la même voie. Il existe également une voiture hasardeuse (**orange**) qui accélère et ralentit au hasard. Enfin il existe une voiture **violette** dont la vitesse dépend du nombre de pièces à ramasser dans le niveau (très lent quand il y a beaucoup de pièces et plus rapide au fur à mesure que ce nombre diminue). Pour simplifier vous pouvez considérer que les voitures peuvent se chevaucher. Attention toutefois, tous les comportements décrits doivent être implémentés.

La rivière :

En plus des voies et de la réserve centrale, certains niveaux peuvent également contenir une rivière. Frogger ne peut pas se tenir debout sur l'eau pour traverser la rivière. Il doit utiliser des troncs d'arbres flottant sur la rivière pour la traverser.

Les troncs d’arbre :

Les troncs (ou bûches) peuvent être utilisées pour traverser une rivière. Contrairement au jeu original, les troncs ne bougent pas et restent au même endroit. Lors de la traversée d'une rivière, Frogger doit se tenir sur une bûche. Si Frogger est sur la rivière mais pas sur un tronc d'arbre, il perd une vie et le jeu est reset de la même manière que s'il avait été heurté par une voiture.

Scores et vies :

En haut de l'écran, le joueur peut lire son score actuel ainsi que le score le plus élevé jamais atteint (meilleur score). De plus, le joueur peut lire en bas de l'écran combien de vie il reste à Frogger. Lorsqu'une voiture parvient à frapper Frogger, Frogger perd une vie et le niveau est redémarré avec une vie en moins. Le jeu se termine lorsque Frogger n'a plus de vies. Lorsque le jeu est terminé, le jeu est redémarré et le score remis à 0. Le score le plus élevé obtenu est conservé.

Les niveaux :

Le jeu se compose de plusieurs niveaux qui augmentent en difficulté. Lorsqu'un niveau est terminé, Frogger passe au niveau suivant. Le jeu devra être composé d’au minimum trois niveaux dont au moins un contient une rivière.

Fonctionnalité supplémentaire :

Additionnement, il vous est demandé d’implémenter au minimum une fonctionnalité supplémentaire originale – par exemple :

- Power up pour marcher sur les buissons
- Item qui modifie le comportement des voitures
- Item qui modifie le comportement de Frogger
- Etc…

Les contraintes et fonctionnalités du projet sont susceptibles d’évoluer au cours du temps. Pensez donc à adapter une stratégie de développement adéquate.

Note : certains points de la description ne sont pas précisés ou sont laissés volontairement vagues. Il revient à vous de faire certains choix d’interprétations. Veillez toutefois à ce que votre approche soit logique et justifiée.

Contrainte de développement :

- Votre programme devra être compilable et exécutable dans un environnement Linux Ubuntu tel qu'une des machines virtuelles utilisées en cours et disponible sur le SharePoint d'eCampus. Un script bash nommé « run.sh » devra permettre la compilation et l'exécution de l'application en utilisant seulement la commande suivante «bash run.sh » en terminale. Si le fichier n'est pas fourni, ou si la compilation et l'exécution ne fonctionnent pas dans l'environnement spécifié, le projet ne sera pas corrigé et sanctionné d'un zéro. **Testez donc avant que tout fonctionne !**

- Votre code devra respecter les principes de designs orientés objet comme vu au cours. Pensez donc à faire des choix logiques de design de classes afin de produire un code propre et maintenable.

- Votre code devra présenter une structure correcte et maintenable. Notamment :
    - Evitez la duplication de code.
    - Evitez les constantes magiques.
    - Evitez le code mort.
    - Commentez intelligemment et suffisamment votre code

  Un code dont la qualité sera jugée insuffisante sera sanctionné d'un zéro.
- Votre jeu devra être développé en utilisant **exclusivement la librairie graphique Java Swing** comme vu en cours avec le jeu du Snake.

Rapport :

Il vous est demandé de rédiger un rapport décrivant votre projet. Votre rapport devra contenir au minimum les sections suivantes :

- Introduction
- Fonctionnalités implémentées
- Analyse
- Conclusion

Introduction : Cette section devra introduire votre projet. Décrire ce qui a été réalisé et présenter brièvement la structure de votre rapport.
Fonctionnalités : Cette section devra expliquer les fonctionnalités offertes par votre application d'un point de vue à la fois fonctionnel et technique.
Analyse : Cette section devra expliquer la structure de votre implémentation en utilisant les outils d'analyses déjà vus durant votre parcours. Si aucun outil n'a été vu pour l'instant, considérez qu'il vous est demander d'élaborer des schémas explicatifs sur la structure de votre code. Attention : Tous les diagrammes doivent être commentés !
Conclusion : Votre conclusion sur le projet. Ce que vous avez réussi à faire ou non durant le projet et les apprentissages que vous en tirez.

Deadline et remise :

La date limite pour la remise du projet est le **Lundi 10 Janvier à 23h59**. Le projet devra être déposé sur eCampus à l'intérieur d'un fichier .zip contenant toutes les sources de votre projet ainsi que le rapport.

Développement et Triche :

- Tout acte de triche sera sanctionné par **un zéro pour l'entièreté du projet**. Des parties de code réutilisés d'un projet existant (d'un autre étudiant ou disponible sur le net) sans références dans votre rapport et sans mention de l'utilité du code utilisé est considéré comme une fraude.

- Pour ce projet, vous ne pouvez pas reprendre des parties du code d'un autre étudiant.

- Pour ce projet vous ne pouvez pas vous inspirer/servir d'un jeu/code disponible sur internet. Vous pouvez toutefois réutiliser les ressources vues en cours tel que le jeu du Snake (http://zetcode.com/javagames/snake/)

- Si vous avez un doute, contactez l'enseignant le plus tôt possible afin d'éviter du refactoring inutile, ou pire, **un zéro**.

Conseil pratique :

Voici quelques conseils qui j'espère pourront vous aider.

- Veillez à ce que votre code ne contienne pas de constantes magique et/ou de duplication qui serait facilement évitable avec l'utilisation de méthodes.
- Veillez à effectivement implémenter les différents comportements des voitures.
- Réfléchissez en terme d'« attribution de responsabilités » en accord avec les principes vu au cours (segmentation logique en classes).
- Ne négligez pas la théorie du cours (vous serez interrogés dessus).
- Ne négligez pas votre rapport. Tachez d'y expliquer/justifier explicitement vos choix d'implémentation. (Exemple : pourquoi un héritage ici ? pourquoi l'implémentation des niveaux comme ceci ? quel avantage en termes de structure ? …)
- Prenez le temps de bien comprendre tout l'énoncé avant de vous lancer (et lisez la FAQ).

FAQ :

- **Puis je ajouter d'autres sections ou sous-sections dans le rapport ?**

Oui. La partie rapport de ce document donne seulement la structure minimum.

- **Puis je coder ou rendre mon rapport en anglais ?**

Oui. Pour ce qui est du code vous devez toutefois respecter les usages corrects des conventions de nommage. Codez en anglais ou en français, pas les deux. L'anglais étant recommandé.

- **Puis je programmer sur Windows avec Eclipse ?**

Oui vous pouvez programmer comme vous le désirez mais vous devez respecter les contraintes de ce document, notamment : votre code doit être exécutable sur un environnement linux Ubuntu via un script run.sh que vous devez fournir en même temps que vos sources (voir les contraintes de développement). Une machine préconfigurée sera disponible sur le SharePoint.

- **Le rapport est-il important ?**

Oui. Le rapport est une **pièce centrale de votre projet** et c'est le premier outil de communication qui me servira à juger de la bonne réalisation du projet, pas seulement du point de vue du code mais également de la méthodologie utilisée.

- **Quel est le niveau de complexité que vous attendez pour les fonctionnalités supplémentaires ?**

La complexité de ce qui été développé sera prise en compte dans l'évaluation. Toutefois, entamer le dev. d'une fonctionnalité trop ambitieux risque de vous

pénaliser si cela implique que vous deviez bâcler votre rapport pour compenser ou omettre des fonctionnalités demandées.

- **Que voulez-vous dire par « tous les diagrammes doivent être commentés ».**

Les diagrammes doivent servir à illustrer et appuyer vos explications sur la structure de votre implémentation. Ils ne remplacent aucunement un texte explicatif revenant sur les points d'attention.

- **Je n'ai pas réussi à tout réaliser. Est-ce que ça vaut la peine de vous rendre le projet ?**

Oui veillez toutefois à être claire sur les parties non implémentées. Il est très déconseiller de dissimuler ou d'« oublier » de mentionner qu'une partie n'a pas été réalisée. Veillez toutefois à bien respecter les consignes. Par exemple, votre code doit pouvoir compiler avec le script bash demandé, la qualité du code doit être suffisante, etc…

- **Puis je réaliser le projet en groupe ?**

Non le projet doit être réalisé individuellement.

- **Que voulez-vous dire par « Votre code devra respecter les principes de designs orientés objet comme vu au cours. »**

L'orienté objet fait intervenir certains principes comme l'héritage ou les méthodes statiques. Il revient à vous de décider quand les mettre en œuvre ou non. Votre approche devra toutefois être logique et justifiée. Cela implique notamment, de faire apparaitre de l'héritage quand cela a du sens, de rendre une classe abstraite quand cela a du sens, d'utiliser intelligemment l'encapsulation etc…

- **Dois-je vraiment faire de l'orienté objet ? Mon programme peut fonctionner sans.**

Vous devez absolument mettre en œuvre l'orienté objet pour ce projet. Cela fait partie des contraintes du projet. Un non-respect de ces contraintes sera pénalisé. L'utilisation de l'orienté objet n'est pas une contrainte faible. Veillez donc à la respecter.

- **Je ne peux vraiment pas utiliser de code venant d'internet ?**

Non appart pour ce qui peut être considéré comme des briques fonctionnelles. (Par exemple le code permettant de lire/écrire un fichier, le code relatif à l'utilisation des listes ou autres structures de données). Dans tous les cas ne prenez aucun risque et contactez l'enseignant le plus tôt possible si vous avez un doute !

- **L’aspect graphique du jeu et la jouabilité sont-ils des critères importants ?**

Ces critères seront pris en compte dans l’évaluation mais sont nettement moins importants que l’implémentation des fonctionnalités et le respect des contraintes. Dis grossièrement : Mieux vaut un jeu moche mais avec toutes les fonctionnalités implémentées qu’un jeu magnifique mais avec des fonctionnalités manquantes.

- **Quels sprites puis je utiliser pour le projet ? Dois-je vraiment trouver une image de grenouille ?**

Vous êtes libre d’utiliser toutes les sprites/images que vous désirez. Pensez toutefois à spécifier leur source dans le rapport. Vous pouvez également utiliser des images abstraites. Vous êtes vraiment libre en ce qui concerne l’aspect purement graphique.

- **Les niveaux peuvent-ils être hardcodés ?**

Oui mais en garantissant une structure de code maintenable.

- **Dois je utiliser une base de données ?**

Non, un simple fichier texte (flatfile) est suffisant.