

Urgent!!! Recherche Bons Lauréats

Cahier des Charges



Sommaire

Sommaire	2
Présentation des acteurs	3
Description du produit	4
But du jeu	4
Conditions	4
Démarrage du jeu	4
La partie	4
Description des cases	5
Détails de l'interface et des niveaux du jeu	6
Contraintes	7
Prévision du déroulement du développement	8
Tache 0 : rédaction du cahier des charges	8
Tache 1 : définir le diagramme des modules.....	8
Tache 2 : développement d'un premier prototype en mode texte.....	8
Tache 3 : développement d'un 2e prototype en mode graphique (SDL).....	9
Diagramme de Gantt	10
Diagramme des modules	11

Présentation des acteurs

La société Union des Cerveaux Bourrés de Logique est une société anonyme au capital de $(1+\sqrt{5})/2$ millions d'euros. Son activité principale vise à l'enrichissement de l'intellect des étudiants de façon ludique.

La société BéCaJu est une multinationale de renommée internationale, issue de l'université de Lyon 1. Elle est composée de trois développeurs à temps partiel. Son activité principale est de répondre aux besoins de l'UCBL. Son expertise est essentiellement centrée sur le C/C++, l'interface graphique, les bases de données élémentaires...

Dans ce contexte, l'UCBL commande le produit « Urgent!!! Cherche Bons Lauréats » à la société BéCaJu. Le présent cahier des charges est composé :

1. d'une description détaillée du produit
2. d'une prévision détaillée du déroulement du développement
3. d'un diagramme de Gantt
4. d'un diagramme des modules

L'ensemble de ces documents sera maintenu pendant toute la durée de développement du produit.

Description du produit

Principe et règles du jeu

‘L’ ‘Urgent !!!! Cherche Bons Lauréats’ est un jeu où l’on doit répondre à des khôlles afin de gagner des points. Le joueur atteignant le nombre de points fixés est le vainqueur.

En partant de la case départ, le premier joueur déplace son pion sur le plateau du jeu suivant le résultat de son lancer de dé. Quand il tombe sur une ‘CaseQuestion’, il doit répondre à la question afin de gagner les points. Le joueur doit se plier aux instructions données par les ‘CarteCheats’. Le joueur pourra aussi être envoyé dans le bureau de Mr. Guillet où il devra alors subir certaines contraintes...

But du jeu

Avant de commencer la partie, les joueurs doivent choisir le nombre de points à atteindre pour terminer le jeu (10, 20, 30, 40). Le gagnant est donc celui qui atteint le premier le nombre de points fixés.

Conditions

2 à 4 joueurs
1 dé

Démarrage du jeu

Pour que le jeu démarre, il faut que les joueurs choisissent le nombre de joueurs puis le nombre de points à atteindre. Ensuite, ils doivent rentrer leur surnom pour pouvoir se repérer sur le plateau. Les joueurs font un premier lancer de dé pour désigner celui qui doit commencer.

La partie

Quand c’est au tour d’un joueur, ce dernier lance le dé et avance du nombre de cases indiqué sur le dé. La case où le joueur est arrêté détermine ce qu’il doit faire. Deux pions ou plus peuvent être arrêtés sur la même case en même temps. Suivant la case où le joueur se trouve, il doit :

- ❖ Répondre a une question tirée aléatoirement
- ❖ Gagner des points
- ❖ Perdre des points
- ❖ Payer des taxes

- ❖ Tirer une carte 'Cheat'
- ❖ Aller chez Mr. Guillet
- ❖ Récupérer des points grâce à la case 'Points Gratuits'

Description des cases

- ❖ **Case Khôlles** : le joueur tire une 'CarteQuestion' correspondant à la matière du professeur avec lequel il passe (Mathématiques, Informatique, Biologie ou Culture Générale). Pour pouvoir gagner des points, il doit répondre correctement à la question posée par la carte (QCM avec 4 possibilités). Une bonne réponse donne un point dans la première moitié du plateau, deux dans la deuxième moitié. Une erreur entraîne la perte du nombre de points correspondant.
- ❖ **Case 'Cheats'** : le joueur pioche une 'CarteCheat' et suit les instructions de la carte. Il existe quatre types d'instructions :
 - Répondre à deux questions dans une matière (la carte prenant effet à la question suivante)
 - Passer un tour (la carte prenant effet au coup suivant)
 - Possibilité de donner à un autre joueur la question piochée (la carte prenant effet quand le joueur le souhaite, il la garde donc dans son jeu jusqu'à l'utiliser)
 - Don d'un des points à n'importe quel joueur qu'il choisit en dehors de lui-même (effet immédiat)
- ❖ **Case 'Taxe'** : le joueur doit donner un de ses points à chacun des joueurs présents sur le plateau
- ❖ **Case 'Impôt'** : la partie entière de 10% des points du joueur va dans la 'CaisseAPoint'
- ❖ **Case RU** : le joueur perd 2 points (il perd du temps à manger alors qu'il devrait travailler)
- ❖ **Case BU** : le joueur gagne 2 points (il travaille pour pouvoir répondre aux prochaines questions)
- ❖ **Case 'Points Gratuits'** : le joueur récupère les points consignés dans la 'CaisseAPoint'
- ❖ **Case '7h45'** : le joueur démarre sa journée. S'il s'arrête sur cette case, il commence à 7h45, il a beaucoup de courage, il gagne donc 2 points.
- ❖ **Case 'Allez voir Mr. Guillet'** : le joueur va directement sur la 'CaseGuillet' sans passer par la case départ
- ❖ **Case 'Vous êtes chez Mr. Guillet'** : le joueur atterrit sur cette case en étant passé par la 'CaseGoGuillet'. Le joueur a alors un problème avec ses transversales : tant que son problème n'est pas résolu, il est coincé sur cette case. Pour pouvoir reprendre la partie, le joueur doit faire 6 ou attendre 3 tours. Mais en s'arrêtant seulement sur la case, c'était comme s'il était spectateur, il passe dans le couloir mais ne subit pas les désagréments de cette case.

Détails de l'interface et des niveaux du jeu

Voici le plateau de base sur lequel se déroule le jeu :



Il est composé de 32 cases :

- 16 cases 'CaseQuestion', chacune étant marquée d'une couleur : celle-ci indique la matière se rapportant à la question qui y sera posée
- 4 cases 'CaseCheat'
- 2 'CaseTaxe'
- 2 'CaseImpot'
- 2 'CaseBU'
- 2 'CaseRU'
- 1 'Case7h45'
- 1 'CaseGoGuillet'
- 1 'CaseGuillet'

- 1 case 'CaisseAPoint'

Les pions ne peuvent se déplacer que sur les cases, sans n'en sauter aucune. Quand un pion s'arrête sur une case, l'action associée à cette case s'effectue :

- Lorsque c'est une 'CaseQuestion', un 'agrandissement' de la case comprenant un nom de professeur, une matière, et un niveau de difficulté s'affiche et une fenêtre s'ouvre dans laquelle se trouvent une question et les quatre réponses possibles
- Lorsqu'il s'agit d'une 'CaseCheat', la carte tirée s'affiche
- Lorsqu'il s'agit d'une 'CaseTaxe', 'CaseImpot', 'CaseBU', 'CaseRU', 'Case7h45', l'affichage n'est modifié que pour le score : le joueur perd ou gagne un certain nombre de points
- Lorsqu'il s'agit de la 'CaseGoGuillet', le pion se déplace jusque dans la 'CaseGuillet'

Contraintes

1. Le jeu sera développé en C/C++
2. La librairie utilisée sera SDL pour la version beta en mode graphique
3. Le code respectera le code ligne standard suivant : code indenté, variables ayant un sens...
4. Le code sera géré et archivé par SVN (grâce à Google Code)
5. La documentation du code sera produite par Doxygen
6. Un diagramme des modules permettra d'avoir une vision de haut niveau de l'implémentation
7. Le code sera fourni par la société UCBL

Prévision du déroulement du développement

Tache 0 : rédaction du cahier des charges

Membres impliqués : tous

Durée : 2 semaines

Tache 1 : définir le diagramme des modules

Membres impliqués : tous

Durée : 2 semaines

Tache 2 : développement d'un premier prototype en mode texte

Durée : 4 semaines

Tache 2.1 : écriture et test du module 'Partie.h'

Membres impliqués : CC

Permet de gérer l'ensemble d'une partie. Ainsi, grâce à ce module, on pourra définir au début de la partie le nombre de joueur, ainsi que les prénoms de chacun. Les compteurs seront initialisés, les jetons seront positionnés sur le plateau et la partie pourra commencer. Ce module gèrera aussi le déroulement de la partie, en vérifiant régulièrement les scores de chaque joueur, et en leur permettant de lancer le dé et se déplacer chacun leur tour tant que le score final n'a pas été atteint... Dans ce dernier cas, la partie s'achève.

Tache 2.2 : écriture et test du module 'Joueur.h/.cpp'

Membres impliqués : BB

Permet de gérer les joueurs, de les faire se déplacer suivant le sens du jeu sur le plateau. Le joueur est caractérisé par un nom qui lui est propre. Le joueur décidera de ce dernier au lancement du jeu.

La position sur le plateau sera modifiée via la valeur du dé, suivant un seul sens. Le score du joueur pourra varier dans les deux sens : aussi bien de façon croissante que décroissante. Le score quant à lui sera toujours supérieur ou égal à zéro.

Le joueur pourra aussi tout au long de la partie conserver des 'CarteCheat', elles lui permettront de passer son tour s'il souhaite ne pas répondre à la question, ou bien l'obligeront à répondre à deux questions pour valider une khôlle... Ces cartes seront accompagnées d'un compteur pour pouvoir permettre au module de tester, à chaque arrivée du joueur sur une case, s'il a une contrainte à respecter, ou s'il possède le droit de passer son tour. Le compteur régressant d'un point à chaque utilisation d'une carte.

Tache 2.3 : écriture et test du module 'DesJoueurs.h/.cpp'

Membres impliqués : BB

Ce module permet de gérer les joueurs simultanément. Le nombre total de joueurs est défini au lancement du jeu, tout comme le nombre de points à atteindre et le nom de chaque joueur. La comparaison des scores des joueurs permet de déterminer quel est le gagnant de la partie lorsque le nombre de points à atteindre est obtenu (ou dépassé) par l'un des joueurs. Le gagnant étant le joueur qui a obtenu ce score.

Tache 2.4 : écriture et test du module 'De.h/.cpp'

Membres impliqués : JD

Ce module permet de faire évoluer le jeu. Un nombre sera tiré au hasard entre 1 et 6 compris et il permettra aux différents joueurs d'avancer de cases en cases tout au long du jeu. Le résultat obtenu est modélisé par un dé.

Tache 2.5 : écriture et test du module 'CarteQuestion.h/.cpp'

Membres impliqués : JD

Ce module permettra de créer des cartes qui permettront à chaque joueur de cumuler des points en répondant de façon correcte aux questions qu'elles contiennent. Chaque carte possède un numéro qui lui permettra d'être répertoriée de façon unique. Chaque carte correspond, en fonction de ce même numéro, à une matière et un niveau de difficulté. De plus, grâce à l'attribut 'nbDeFoisCartePosee', lorsque le joueur répondra de façon correcte à une question, cette dernière ne sera pas reposée dans la partie. A une carte correspond une unique question étant elle-même affiliée à une seule réponse correcte, parmi quatre propositions. Si le joueur répond de façon correcte à la question, il marque le nombre de points indiqué par le niveau de la carte. Si le joueur ne répond pas correctement à la question, il perd le nombre de points correspondant ; ses points étant cumulés sur son score.

Tache 2.6 : écriture et test du module 'Case.h/.cpp'

Membres impliqués : JD

Permet de gérer l'ensemble des cases : en effet, chacune des cases a une action bien précise, qui sera exécutée par ce module. Ainsi, chaque case est initialisée à une position unique, ce qui permettra par la suite en fonction de la position du joueur, de déterminer sur quel type de case il se trouve, et ainsi de définir l'action à effectuer.

Tache 2.7 : écriture et test du module 'CaisseAPoint.h/.cpp'

Membres impliqués : CC

Permet de gérer la 'CaisseAPoint' : le joueur qui se trouve sur cette case voit son score augmenter du nombre de points contenus dans la 'CaisseAPoint'.

Tache 2.8 : écriture et test du module 'Plateau.h/.cpp'

Membres impliqués : BB

Permet d'initialiser le plateau, en modélisant l'ensemble de ses cases.

Tache 2.9 : mise en commun des modules et tests

Membres impliqués : tous

Tache 3 : développement d'un 2e prototype en mode graphique (SDL)

Tache 3.1 : exploration et compréhension de SDL

Membre impliqué : CC

Tache 3.2 : Réalisation du module d'affichage en SDL (gestion des événements de la souris, saisie au clavier, lecture sur fichiers et affichage de texte, gestion de cadres dans une fenêtre)

Membres impliqués : CC, BB, JD

Diagramme de Gantt

Taches	Semaines :									
	1	2	3	4	5	6	7	8	9	10
0	X	X								
1		X	X							
2.1		X	X	X	X					
2.2		X	X	X	X					
2.3		X	X	X	X					
2.4		X	X	X	X					
2.5		X	X	X	X					
2.6		X	X	X	X					
2.7		X	X	X	X					
2.8		X	X	X	X					
2.9				X	X	X	X			
3.1						X	X			
3.2							X	X	X	X

Diagramme des modules

