Projet LO21 : Tower Defense

Les insectes envahissent le jardin de la maison des étudiants !

Objectifs

- Construction d'interface graphique avec Qt ;
- Utilisation de QGraphicsScene et QGraphicsView ;
- Utilisation des notions de l'héritage, du polymorphisme, des méthodes virtuelles (pures), des classes abstraites, etc. ;
- Utilisation des classes Templates (QList, QVector, etc.).
- Utilisation des design patterns de construction d'objet (Builder)

Introduction

Comme chaque année, le premier barbecue du jardin de la Maison des Etudiants se prépare. Ciel bleu, merguez et chipolatas, tout va pour le mieux... Mais des intrus ont décidé de s'inviter pour gâcher la fête : fourmis, moustiques, cafards et guêpes entendent bien prendre possession des lieux et s'accaparer les victuailles !

Les étudiants se mobilisent et chacun y va de sa solution, plus ou moins efficace, pour protéger la MDE de l'invasion : du lance pierres bricolé, au paint-ball, en passant par les boules de pétanque.

Face à l'ampleur de l'invasion, l'administration des études accorde des crédits au BDE pour chaque destruction d'insecte... de quoi recruter de nouveaux exterminateurs parmi les étudiants et empêcher les insectes de gâcher la fête.

Description

Le but est de réaliser un jeu de type *Tower Defense* en C++, en utilisant au mieux la notion d'objet et les fonctionnalités fournies par Qt (programmation événementielle par signaux et slots, objets graphiques, conteneurs…).

Pour des exemples de ce type de jeu, voir le site *http://www.towerdefence.net*

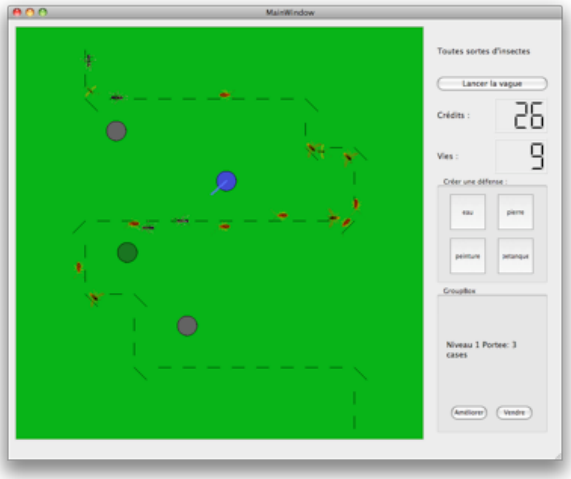
Dans ce jeu les ennemis arrivent par vagues successives et tentent d'arriver au but en suivant un chemin prédéfini. Le but du jeu est de placer stratégiquement des défenses sur la carte afin de détruire tous les ennemis avant qu'ils ne puissent atteindre le but. Chaque défense ayant un coût, il est nécessaire de faire preuve de stratégie.

Quand un ennemi atteint le but, une vie est perdue. On commence avec 10 vies. La partie se termine quand toutes les vies sont perdues.

L'interface du jeu se décompose en deux zones distinctes :

- la carte, sur laquelle les ennemis progressent et où l'on peut positionner des défenses
- le panneau de jeu, qui permet de choisir le type de défenseur à créer ou d'agir sur le défenseur sélectionné. Il fournit également des informations sur le déroulement du jeu.

La figure suivante présente une disposition type de l'interface de jeu :



Des éléments graphiques de base (herbe, but, ennemis, défenseurs) vous sont fournis.

La carte

Le jardin est composé de 16x16 dalles carrées contenant soit un chemin, soit de l'herbe, soit de la boue ou enfin le but.

Les insectes apparaissent sur la case identifiée comme départ, et suivent le chemin jusqu'au but. Les étudiants défenseurs sont courageux, mais pas téméraires, ainsi, ils ne se placent qu'en dehors du chemin des insectes, et évitent à tout prix les zones boueuses.

La carte est fournie sous la forme d'un fichier texte (map.txt) comprenant 16 lignes de 16 entiers, avec le codage suivant :

- 0 = libre
- 1 = chemin vers le Nord
- 2 = chemin vers le Sud
- 4 = chemin vers l'Est
- 8 = chemin vers l'Ouest
- 5 = chemin vers le NE
- 6 = chemin vers le SE
- 9 = chemin vers le NO
- 10 = chemin vers le SO
- Une des valeurs précédentes + 16 = case de départ
- 32 = arrivée des ennemis
- 64 = boue

Les types d'ennemis

Les ennemis (insectes) sont caractérisés par leur taille, leur mode de déplacement (rampant ou volant), leur vitesse de déplacement (en cases par seconde), leur vitalité (hp) et leur résistance aux attaques (resist).

Lorsqu'un insecte est touché, sa vitalité est modifiée comme suit :

Quand la vitalité d'un insecte tombe à 0, il meurt. Chaque ennemi tué rapporte 1 crédit.

En plus des caractéristiques de base, chaque type d'insecte possède des caractéristiques spéciales.

Les fourmis

Caractéristique	Valeur
Déplacement	rampant
Vitalité	5 x taille²
Résistance	1 x taille²
Vitesse	2 + taille/2

Les fourmis accélèrent quand elles sont touchées. Ainsi, leur vitesse de base est multipliée par 1.5 pendant les 5 secondes suivant le dernier coup reçu.

Les cafards

Caractéristique	Valeur
Déplacement	rampant
Vitalité	10 x taille²
Résistance	5 x taille²
Vitesse	2

Les cafards de taille supérieure ou égale à 2 libèrent deux jeunes cafards de taille quand ils meurent.

Les guêpes

Caractéristique	Valeur
Déplacement	volant
Vitalité	7 x taille²
Résistance	4 x taille²
Vitesse	3

Lorsqu'elles meurent, les guêpes s'écrasent au sol et causent un dommage de 5*taille² aux insectes rampants situés à moins d'une case et demi de distance.

Les moustiques

Caractéristique	Valeur
Déplacement	Rampant et volant
Vitalité	6 x taille²
Résistance	1 x taille² en vol 15 x taille au sol
Vitesse	2 + taille/2 en vol 1 + taille/2 au sol

Les moustiques alternent 7 secondes de vol avec 3 secondes de marche au sol. Vulnérables dans les airs, ils savent se cacher au sol, ce qui les rend très résistants. Quand ils sont touchés en vol, les moustiques se posent (pendant 3 secondes).

Les vagues d'ennemis

Les ennemis arrivent à l'entrée du chemin par vagues successives.

Une vague est caractérisée par sa description et le type et le nombre d'ennemis la composant. Un fichier d'exemple de description des vagues (waves.txt) vous est fourni. Il se compose d'une ligne par vague :

Quinze petites fourmis ; fourmi:1:15:25
Les fourmis et une grande guêpe ; fourmi:1:15:25 ; guepe:1.7:1:25
Sept cafards, trois moustiques et dix fourmis géantes ; cafard:1:7 :25; moustique:1.1:3 :30 ; fourmi:3:10:40
[..]

Les informations sont séparées par des points virgules, on trouve d'abord la description de la vague, suivie de la liste des ennemis composant la vague, sous la forme de triplets T:S:N:I où

- T est le type d'ennemi (fourmi, cafard, guepe, moustique)
- S est la taille de l'ennemi (un nombre flottant de 1 à 3)
- N est le nombre d'ennemis
- I est le nombre de pas de temps séparant l'apparition de deux ennemis (en 50^{e} de seconde si le timer principal tourne à 50Hz)

Les défenses

Les défenseurs sont caractérisés par la portée de leurs armes, la cadence de tir, et la force de frappe, et leur niveau. Ils ne peuvent pas forcément toucher toutes les sortes de cibles.

Les caractéristiques de base d'un défenseur peuvent être améliorées en achetant une amélioration : le défenseur peut ainsi passer du niveau 1 aux niveaux supérieurs (jusque 3).

On peut revendre un défenseur. On récupère alors la moitié du coût total de cette défense (construction et amélioration).

Le pistolet à eau

Accessoire quasi indispensable en ces temps de canicule, le pistolet à eau trouve également son utilité dans la guerre contre les insectes. Assez peu efficace, c'est néanmoins une arme rapide.

Caractéristique	Valeur
Cible	Volant et rampant
Portée	2 + niveau/2
Cadence	(4 - niveau/2) / seconde
Frappe	5 * niveau^1.5
Coût de base	8 Crédits
Coût amélioration 1→2	20 Crédits
Coût amélioration 2→3	45 Crédits

Projectile	Eau
Vitesse du projectile	40 cases/seconde

Le lance pierres

Arme efficace contre les cibles volantes quand elle est bien maniée, les ennemis ne résisteront pas longtemps. Son principal inconvénient est le temps nécessaire pour trouver un gros caillou à lancer entre deux tirs.

Caractéristique	Valeur
Cible	Volant
Portée	3 + niveau/2
Cadence	1 / seconde
Frappe	10 * niveau^1.5
Coût de base	12 Crédits
Coût amélioration 1→2	25 Crédits
Coût amélioration 2→3	60 Crédits
Projectile	Pierre
Vitesse du projectile	25 cases/seconde

Le paint-ball

Joueur acharné de paint-ball, son lanceur semi-automatique de billes de peinture verte biodégradable va faire un carnage en engluant toutes sortes d'insectes.

Caractéristique	Valeur
Cible	Volant+Rampant
Portée	4 + niveau/2
Cadence	2 / seconde
Frappe	4 * niveau^1.5
Coût de base	12 Crédits
Coût amélioration 1→2	25 Crédits
Coût amélioration 2→3	60 Crédits
Projectile	Bille de peinture
Vitesse du projectile	30 cases/seconde

Lorsqu'ils sont touchés par les billes de peinture, les insectes sont ralentis. À chaque bille de paint-ball reçue, la vitesse de l'ennemi est réduite à la moitié de sa valeur standard pendant 4 secondes.

Le joueur de pétanque

Partisan du sport tranquille, le joueur de pétanque va pour la bonne cause réussir le carreau et réduire les envahisseurs en purée. N'attendez pas de lui une cadence de tir soutenue, mais sa précision est redoutable, et l'onde de choc produite affecte également les insectes rampants proches de l'impact.

Caractéristique	Valeur
Cible	Rampant
Portée	3 + niveau
Cadence	0.5 / seconde
Frappe	15 * niveau^1.5

Coût de base	15 Crédits
Coût amélioration 1→2	40 Crédits
Coût amélioration 2→3	80 Crédits
Projectile	Boule de pétanque
Vitesse du projectile	20 cases/seconde

Les ennemis situés à au plus 1 case de la cible sont également touchés par le choc, avec une frappe de 10*niveau^1.5

Le musicien (bonus)

Non violent et craintif des insectes, le musicien préfère rester hors de la bataille, mais encourage ses amis en poussant la chansonnette.

Caractéristique	Valeur
Cible	Aucun
Portée	1.5 case
Cadence	NA
Frappe	0
Coût de base	15 Crédits
Coût amélioration 1→2	40 Crédits
Coût amélioration 2→3	80 Crédits

Suivant le niveau du musicien, les défenseurs à sa portée bénéficient d'un bonus d'attaque de 20% (niveau 1), 40% (niveau 2) ou 60% (niveau 3).

Fonctionnalités principales

L'application devra réaliser les fonctions de base suivantes :

- Chargement de la carte ;
- Chargement des éléments graphiques ;
- Lecture du fichier de vagues d'ennemis ;
- Nouvelle partie, Pause ;
- Achat et placement des défenses sur la carte ;
- Animation des ennemis ;
- Amélioration, revente de la défense sélectionnée ;
- Passage à la prochaine vague d'ennemis ;

Par ailleurs, toute amélioration sera la bienvenue (ex : Sauvegarde de la partie courante dans un fichier, sons, types supplémentaires d'ennemis et de défenses, éditeur de carte…).

Modélisation

Vous modéliserez les Défenses, les Projectiles et les Ennemis par des classes, en exploitant au mieux mécanismes de l'héritage, du polymorphisme et des fonctions virtuelles.

La scène de jeu sera représentée par un objet dérivant de la classe Qt *QGraphicsScene*. Les objets composant une *QGraphicsScene* dérivent de la classe *QGraphicsItem*. Enfin, cette scène pourra être affichée dans un widget *QGraphicsView*.

(Voir à ce sujet la partie *Examples/Graphics View Examples* de la documentation Qt)

L'intérêt d'utiliser ces classes est qu'elles implémentent un mécanisme de gestion des événements au niveau des objets composant la scène (clic de souris, survol). Un mécanisme de mise à jour pour des animations simples est également prévu via le slot *advance()* de la *QGraphicsScene*, qui appelle la fonction membre *advance* de tous les objets composant la scène (attention, *advance* est appelée 2 fois lors d'une mise à jour, consultez la documentation).

Pour l'animation graphique des ennemis, consultez *Examples/Graphics View Examples/Ported Asteroids*

La gestion du temps sera confiée à un *QTimer* cadencé à 50 Hz (période de 20 ms).

Vous appliquerez le design pattern *Fabrique* (*Factory*) à la création des ennemis.

Rendu

Votre code sera commenté en suivant la convention de Doxygen. Vous pourrez ainsi générer automatiquement un document présentant les différentes classes et leurs fonctionnalités.

Vous complèterez cette documentation automatique par quelques pages présentant le sujet, la modélisation du problème et sa réalisation.