Sujet Projet Informatique:
Jeu vidéo: bataille de tanks.

2012

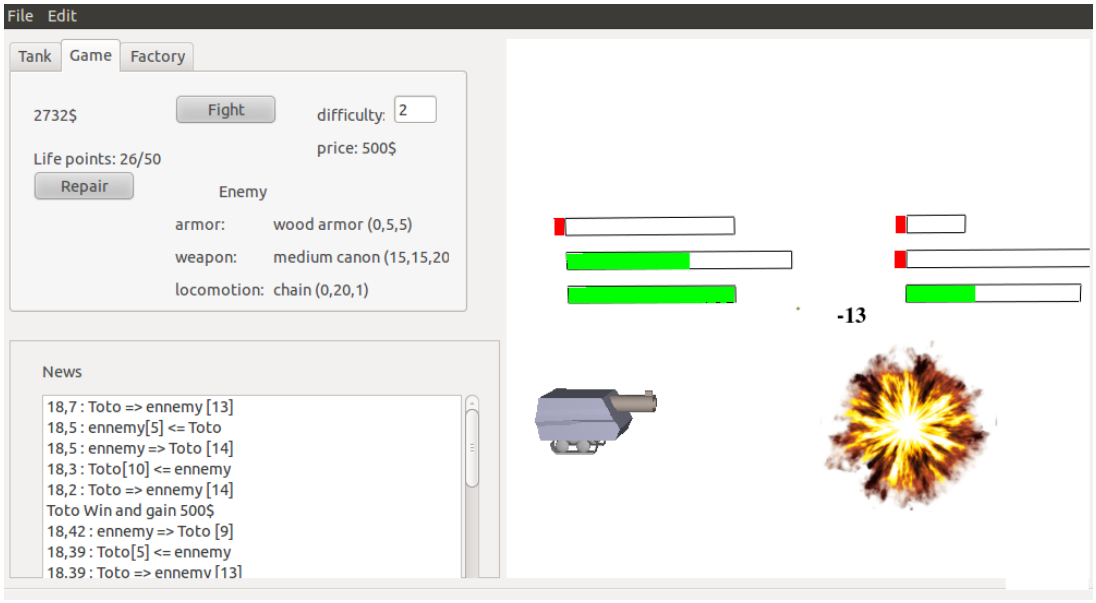

FIGURE 1 – Exemple d'image de résultat possible du projet. La partie droite visualise un combat entre deux tanks dont les barres de points de vie indiquent l'état du combat. La partie gauche rassemble différentes informations telles que les points de vie, la prime du combat, ainsi que l'historique du combat et des actions du joueur.

1 But

Le but de ce projet est de réaliser un jeu de bataille entre tanks se jouant suivant des lois de jeu de rôle. Le jouer pouvant acheter des composants individuels afin d'améliorer celui-ci.

2 Principe général du jeu

Le jeu consiste à faire affronter le tank du joueur face à celui généré par le programme (voir exemple en fig. 1). En fonction des composants spécifiques du tank, celui-ci frappera plus ou moins fort, et pourra résister plus ou moins longtemps. Lorsqu'une bataille est gagnée, le joueur gagne un prix sous forme d'argent qui lui permet d'acheter différentes améliorations pour son tank. Plus la difficulté du combat est importante, plus le gain d'argent l'est également.

Le jeu se comporte donc en deux temps :

1. Le combat entre deux tanks qui se déroule au tour par tour et dont les règles suivent une loi de jeu de rôles qui seront précisés ultérieurement dans l'énoncé. Le joueur n'a pas la main sur le combat, celui-ci dépend des paramètres de l'armement de chaque tank ainsi que d'un facteur chance. Une fois le combat débuté, celui-ci ne prend fin qu'une fois que l'un des deux tanks est détruit.
2. La part d'amélioration du tank. Lorsque le joueur possède suffisamment d'argent, il peut se rendre dans l'usine de fabrication afin d'acheter l'une d'entre elles en fonction de ces moyens et choix. Suivant les choix stratégiques du joueur lors de ses achats, celui-ci pourra affronter des adversaires de mieux en mieux équipés.

Le joueur perd la partie si son tank est totalement détruit et qu'il ne possède plus d'argent.

2.1 Constitution d'un tank

Un tank est constitué de 3 parties fondamentales (voir fig. 2) :
– Une carrosserie
– Une arme
– Un moyen de locomotion

Chacune de ces parties pourra indépendamment faire l'objet d'une personnalisation par le joueur parmi les composants disponibles. On pourra ainsi imaginer que le joueur pourra choisir entre un pistolet, une arme automatique, ou un canon comme arme. De même, la carrosserie pourra tout d'abord être consistuée de bois avant de pouvoir acquérir une carrosserie en métal, ou en kevlar. Chacune de ces spécialisations doit être achetée à un certain prix.

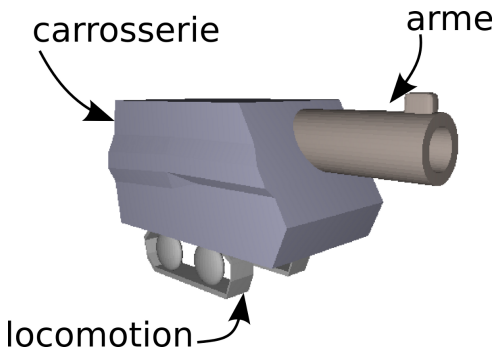

FIGURE 2 – Les 3 parties constitutives d'un tank.

Chacune de ces parties contiennent elle-même 3 attributs qui les différenties :
– Des points d'attaques
– Des points de vies
– Un poids

La somme des points d'attaques, de vies, et les poids permettent d'obtenir les caractéristiques du tank.
– Plus les points d'attaques sont importants, plus le tank pourra frapper fort.
– Plus les points de vies sont importants, plus le tank pourra résister aux attaques.
– Plus celui-ci est léger, plus il est efficace en ripostant.

Un tank en bon état sera également plus efficace en attaque qu'un tank avec des parties non fonctionnelles.

Enfin, un tank pourra également contenir des objets spéciaux qui permettront d'influencer le combat tel que la possibilité de tirer deux fois, de placer des mines sous le tank adverse, ou bien encore de réduire la puissance de feux de l'ennemi.

2.2 Déroulement d'un combat

Un combat se déroule au tour par tour : Le tank du joueur attaque le tank adverse, puis celui-ci attaque le tank du joueur. Ce processus se succède jusqu'à ce que l'un des deux tanks n'ai plus aucun point de vies.

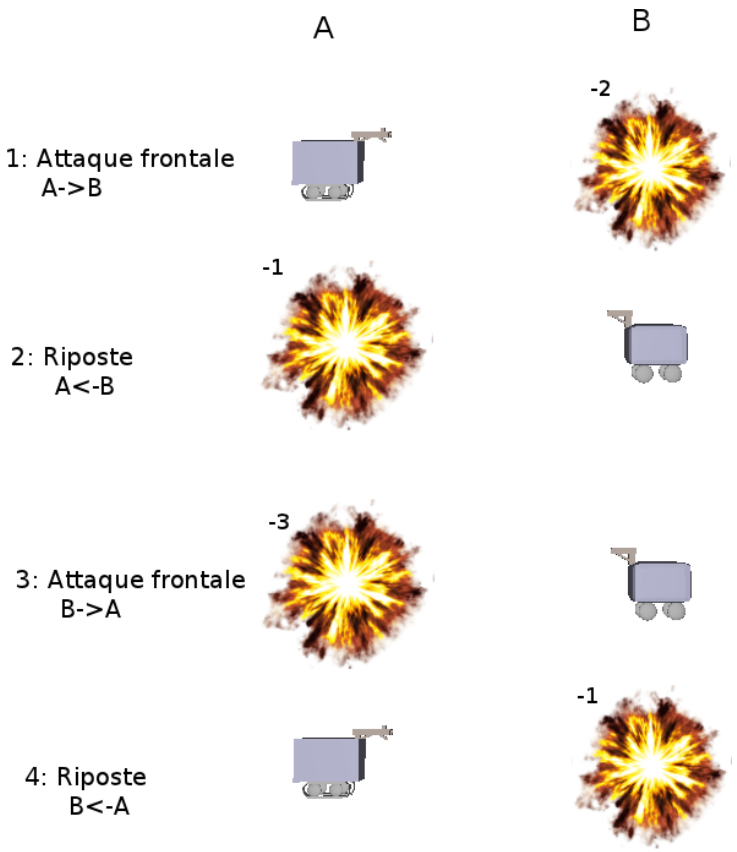

FIGURE 3 – Les 4 parties du déroulement d'un tour complet de combat.

Chaque attaque se déroule elle-même en deux parties (voir fig. 3) :
- L'attaque du premier tank sur l'adversaire qui utilise les points d'attaques à leur pleine puissance.
- La partie de riposte rapide du tank adverse qui a subi l'attaque. Plus celui-ci est puissant et léger, plus il pourra faire de dommages en riposte à l'attaquant.

L'algorithme du déroulement d'un combat peut donc être schématisé sous la forme :

```
A<- tank du joueur
B<- tank ennemi

Tant que tank non détruit

  A.attaque(B);
  B.riposte(A);

  B.attaque(A);
  A.riposte(B);

Fin tant que
```

Notez qu'une fois le combat débuté, celui-ci ne peut s'arrêter qu'après la destruction d'un des deux adversaires. Il n'est pas possible de se retirer, ni de se réparer ou d'acheter d'autres éléments lors du combat.

3 Details du jeu

3.1 Règles de combats

Les règles de bases que devra suivre le jeu lors de la phase de combat sont les suivantes :

3.1.1 Puissance d'attaque

La puissance d'attaque Pa est calculée comme étant la somme des points d'attaques de l'ensemble des éléments composants le tank. Si un élément est détruit (point de vie=0), alors ses points d'attaques correspondants sont divisés par 2.

```
Pa=0;
if armure.point_de_vie >0
  Pa += armure.point_attaque
else
  Pa += 0.5*armure.point_attaque

if arme.point_de_vie >0
  Pa += arme.point_attaque
else
  Pa += 0.5*arme.point_attaque

if locomotion.point_de_vie >0
  Pa += locomotion.point_attaque
else
  Pa += 0.5*locomotion.point_attaque
```

3.1.2 Attaque frontale

Lorsqu'un tank attaque directement un adversaire, les points de dommages infligés à l'adversaire sont donnés par $Pa \pm 15\%$. La variation de 15% étant modélisée par une valeur aléatoire de distribution uniforme.

3.1.3 Riposte

Lorsqu'un tank riposte face à un adversaire, les points de dommages infligés à l'attaquant sont donnés par $\mu/2 \times (Pa \pm 15\%)$, où μ est le rapport entre le poids du tank qui a attaqué et le poids du tank qui riposte.

3.1.4 Attribution des dommages

Le cas de l'attaque frontale ou de la riposte aboutit à l'encaissement de points de dommages. Ceux-ci sont déduits des points de vie du tank qui les subits.

Dans un premier temps, l'armure est le premier élément qui absorbe les dommages. Si celle-ci est détruite, c'est ensuite la partie locomotion qui perd des points de vies. Enfin, si celle-ci est également détruite, c'est enfin l'arme qui perd ses points de vies. Un exemple est illustré en fig. 4.

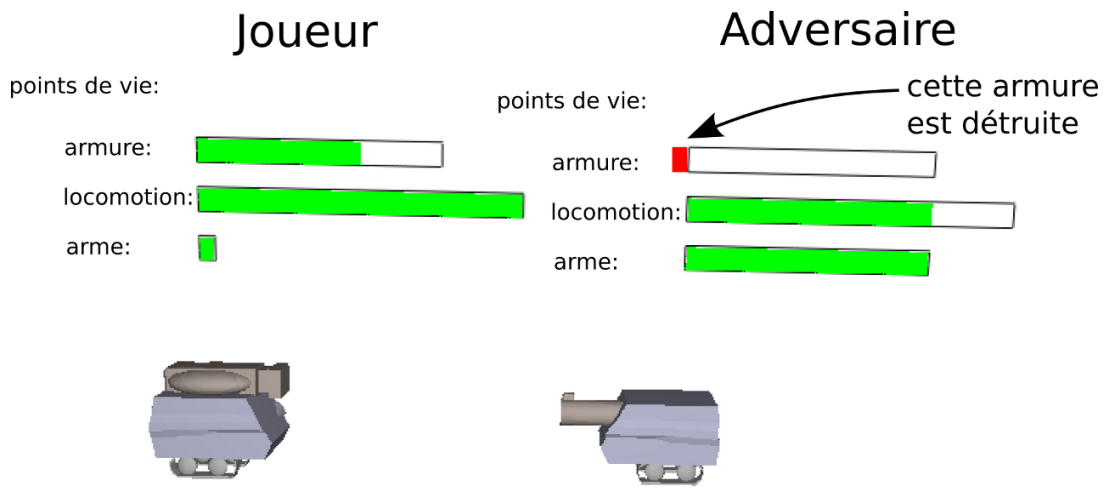

FIGURE 4 – Exemple de points de vie restants pour chaque partie des tanks lors du déroulement d'un combat.

3.1.5 Destruction d'un tank

Un tank est considéré comme détruit lorsque tous ses points de vies sont à 0. C'est-à-dire que sont armure, sa locomotion ainsi que sont armure sont détruites.

3.1.6 Poids d'un tank

Le poids d'un tank est donné par la somme des poids de chacun des éléments qui le composent.

3.1.7 Elements spéciaux

Les éléments spéciaux d'un tank agissent avant que celui-ci n'attaque de manière frontale. Leur comportement peut agir sur le tank adverse en tant que malus ou sur le tank actuel en tant que bonus.

3.1.8 Réparation

Une fois un combat terminé, le tank du joueur a potentiellement subi des dommages qu'il est possible de réparer avant de lancer un nouveau combat. La réparation d'un élément coute

au joueur l'équivalent des points de vies à restaurer. Lorsqu'un élément est totalement détruit (points de vie à 0), sa réparation coute plus cher et chaque point de vie à restaurer coute alors le double.

Chaque élément du tank est restauré séparément dans les possibilités monétaires du joueur. Notez que le joueur ne peut pas prendre crédit, et d'autre ne peut pas posséder une somme d'argent négative. Si le contenu monétaire du joueur ne permet pas de réparer totalement la pièce du tank, alors celle-ci sera réparée au prorata de l'argent restant. Les éléments du tank seront réparés dans cet ordre prioritaire : Armement, puis locomotion, et enfin armure.

3.2 Difficulté de combat et prime

L'inscription à une bataille d'une difficulté donnée aboutit à la génération automatique d'un tank adverse dont le niveau dépend de la difficulté choisie.

Par exemple, au niveau 0 les tanks adverses possèderont une armure faible ainsi qu'une puissance d'attaque très limitée. Par exemple, le seul armement disponible sera un couteau ou un pistolet. Au niveau 2 par contre, les tanks possèderont des armures et armements beaucoup plus lourds.

La génération d'un adversaire doit se réaliser avec une part d'incertitude. Par exemple, au niveau 0, nous pourrons définir que l'adversaire aura 60% de chance d'être équipé d'un couteau, et 40% de chance d'être équipé d'un pistolet.

Le joueur n'aura pas connaissance de son adversaire potentiel avant de débuter un combat. Une fois celui-ci débuté, le joueur devra attendre la fin de celui-ci sans avoir la possibilité de se retirer.

Plus le niveau de difficulté croit, plus la prime emportée par le joueur est importante. Mais également, le risque de perdre, et donc de devoir débourser la moitié de la prime.

3.3 Achat

Le joueur possède la possibilité d'acheter des éléments d'amélioration ou de choix stratégique de son tank parmi une liste d'ensemble d'éléments (voir fig. 5). Ceux-ci sont classés suivant 4 catégories :

- Les objets d'armement (pistolet, fusil, canon, etc)
- Les armures (bois, plastique, métal, etc)
- Les moyens de locomotion (roues, chaines, etc)
- Les objets spéciaux (ex. mines, brouilleurs, pointeur laser, etc)

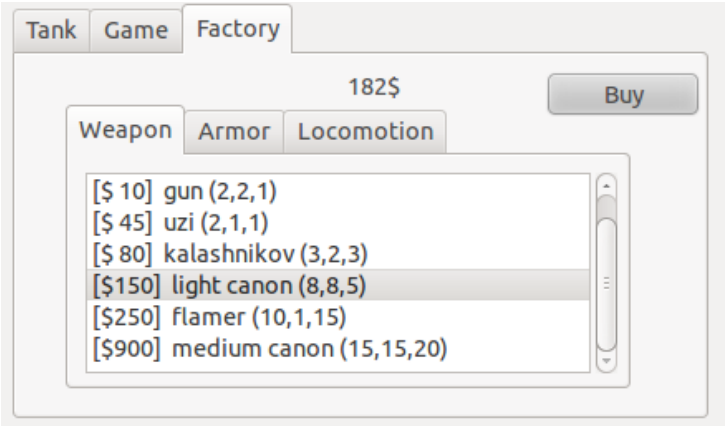

FIGURE 5 – Exemple de possibilité d'achat de la partie armement.

Chaque élément possède un prix. Pour pouvoir acquérir un élément, le joueur doit posséder au moins la somme de celui-ci (le crédit n'est pas possible). Un tank possède nécessairement un unique objet des trois premières catégories. Lorsque le joueur achète l'un d'entre eux, il vient alors remplacer l'élément précédent. Les objets spéciaux quant à eux viennent s'accumuler. Le joueur pouvant posséder autant d'objets spéciaux qu'il en achète.

Au combat suivant, les objets qui ont été achetés par le joueur viennent alors équiper son tank.

4 Travail demandé

4.1 Diagramme de classe

Réfléchissez dans un premier temps au diagramme de classe permettant de réaliser ce jeu. **Vous devrez faire valider ce diagramme avant de débuter le projet.**

Aide sur le diagramme :
- Un joueur possède un tank, de l'argent, et potentiellement un nom.
- Un tank est formé d'une armure, d'une arme, et d'un moyen de locomotion. Il contient également un ensemble d'objets spéciaux.
- Un élément constitutif d'un tank possède des points de vies qui peuvent évoluer en fonction du combat, des points d'attaque et un poids.
- Tout élément pouvant être acheté doit posséder un prix.
- Tout élément pouvant être affiché doit posséder une méthode *affiche*.
- Tout élément qui possède des actions spéciales doit pouvoir agir sur le tank courant ou adverse. Quelle particularité de programmation objet doit posséder les objets possédant des actions spéciales ?
- L'enregistrement à un tournoi doit aboutir à la génération d'un tank adverse dépendant du niveau de difficulté.
- Les objets finaux vus par l'utilisateur : couteau, canons, armures en métal, roues, mines, etc. doivent posséder des propriétés par défaut que l'on peut documenter (prix, points d'attaques, poids, etc).
- On doit pouvoir visualiser le déroulement d'un combat lorsqu'il a lieu (en mode texte dans un premier temps ou graphique).

Définissez bien quelles sont les relations entre objets : *Est-contenu dans*, *Hérite de*, ou *Connait*. Quels sont les objets qui doivent posséder un comportement polymorphe ?

4.2 Déroulement du projet

On distinguera la partie coeur du jeu (déroulement d'un combat, gestion d'un tank et de ses éléments) de la partie graphique (affichage, information). Dans un premier temps, votre projet devra tourner entièrement en mode texte, les affichages étant réalisés sur la ligne de commande. Lorsque le coeur de ce projet sera réalisé et validé, vous pourrez alors passer à l'étude d'une réalisation graphique.

Votre projet devra satisfaire au déroulement suivant, et chaque étape devra être validée par un enseignant avant de continuer.

1. Réfléchissez et établissez un diagramme de classes présenté à la section précédente. Faite valider ce diagramme.
2. Construisez un dossier où vous placerez les éléments de votre projet. Écrivez un programme principal (main) qui ne fait rien et créer le Makefile associé. Assurez vous que vous avez activé les *Warnings* et le mode *Debug*.
3. Générez les en-têtes des classes essentielles au coeur du projet. Ecrivez le nom des méthodes, les arguments qu'elles doivent posséder, les attributs privés qu'elles possèdent et réalisez la hiérarchie de classe. Ces en-têtes doivent être la représentation d'une partie de votre diagramme de classes.
   Notez qu'on ne s'intéressera pas aux classes d'affichage dans cette étape. On pourra également ne pas considérer les objets spéciaux dans un premier temps.
   – Après l'ajout de chaque en-tête *.h* construisez le fichier *.cpp* correspondant (sans implémenter de code à l'intérieur) et ajouter la ligne du makefile qui lui correspond.
   – Vérifiez que l'ensemble compile après chaque inclusion.
   Faites valider cette structure de code.
4. Commencer à compléter le code des classes de bases de composants d'un tank.
5. Compléter vos classes jusqu'à obtenir un tank formé d'une arme, d'une armure et d'un moyen de locomotion viable. On ne commencera pas à spécialiser des objets spécifiques (canon, pistolet, etc) avant d'avoir validé cette étape.
   – Testez le comportement d'un tank que vous génèrerez dans votre fonction *main*. Équipez-le d'éléments de bases et vérifiez chacune des méthodes d'accès. Pensez aux méthodes nécessaires lors du déroulement d'un combat.
   – Faite valider cette étape.
6. Implémentez le déroulement d'un combat entre deux tanks que vous créez en suivant les spécifications de l'énoncé. Affichez le déroulement sur la ligne de commande. Implémentez séparément l'attaque frontale et la riposte dans des méthodes différentes. Vérifiez en détail le déroulement du combat. Faites valider cette étape en montrant les vérifications que vous avez effectuées.
7. Implémentez la notion de joueur avec une quantité d'argent. Mettez en place les méthodes d'achats et de réparations.
8. Réalisez la spécialisation de quelques éléments typiques d'armement, d'armure et de moyen de locomotion. Vérifiez les achats de ces éléments définitifs et le bon déroulement des combats avec ceux-ci.
9. Implémentez la notion d'inscription à un tournoi avec la génération d'un adversaire dépendant de la difficulté, ainsi que la gestion du gain.
10. Procédez à un test global du bon déroulement de votre programme. Vous devez, à partir de cette étape pouvoir réaliser des combats, vous inscrire à ceux-ci en gagnant/perdant de l'argent, et acheter des éléments. Faites valider cette étape, il s'agit du fonctionnement de base du jeu.

Toutes les étapes qui suivent ne devront pas être implémentées avant d'avoir scrupuleusement vérifié le bon comportement des étapes précédentes.

1. Implémentez la présence d'objet spéciaux (sans spécifier leur comportement).
2. Spécialisez un plus grand nombre d'objets
   – Plus grand choix de types d'armes, d'armures, de moyens de locomotions.

– Objets spéciaux avec comportement : ex. Une mine possède une chance sur cinq de faire exploser le tank adverse à chaque tour, mais ne peut être utilisée qu'une seule fois, et une seule par tour. Un viseur laser permet d'améliorer la précision en limitant le malus du facteur chance des 15% initiaux.

4.3 Améliorations

Les améliorations ne seront mises en place que si les étapes précédentes du projet fonctionnent, ont été testées et validées.
Voici un certain nombre d'améliorations du projet qui peuvent être mises en place dans l'ordre que vous souhaitez.

– Mise en place d'un menu permettant de visualiser les caractéristiques de votre tank, et d'acheter du materiel. On pourra utiliser Qt pour cela.
– Mise en place d'une représentation graphique, soit 2D (images), soit 3D, de votre tank et de l'adversaire.
– Amélioration stratégique de votre jeu : Affinez les règles et agrémentez les objets de choix stratégiques qui rendront le jeu plus intéressant : ex. Consensus entre poids et puissance/résistance, etc.

5 Annexe random

Ce jeu demande l'utilisation de nombres aléatoire, voici un code permettant de générer une valeur de type aléatoire avec une distribution proche d'être uniforme entre 0 et 1 :

```
#include <cstdlib>
#include <ctime>
#include <iostream>

double genere_rand()
{
  int grand_nombre=4e3;
  int r=rand();
  return (r%grand_nombre)/static_cast<double>(grand_nombre);
}

int main()
{
  //initialise le generateur aleatoire
  srand ( time(NULL) );

  double nombre=genere_rand();
  std::cout<<nombre<<std::endl;

  return 0;
}
```