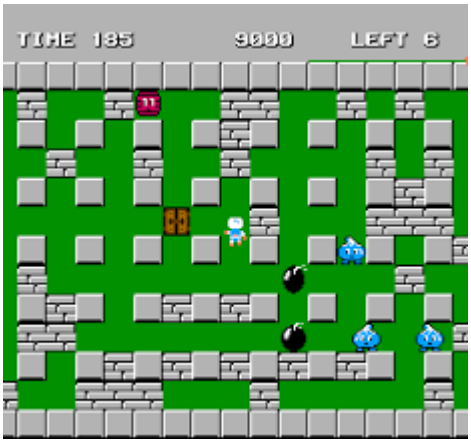
BomberBUT

Sujet de SAE 1.01 2024-2025

Résumé

Vous allez développer un jeu vidéo en tour par tour nommé BomberBUT. Il s'agit d'une adaptation du célèbre jeu vidéo Bomberman dont la première version date de 1985.

L'histoire : *dans un futur lointain, des fonds monétaires ont enfin été débloqués pour rénover le bâtiment Maryse Bastier de l'IUT, aussi connu sous sous le nom de halle informatique. L'UVSQoca-Cola, un consortium industrio-universitaire, fait alors appel à une entreprise de démolition, qui doit rapidement casser tous les murs de la halle. Mais tout ne sera pas si simple car des fantômes d'anciens étudiants ne voulant pas quitter les lieux adorés errent dans la halle pour empêcher sa démolition...*



Le joueur, dont le personnage est appelé Bomber, essaie de rester en vie le plus longtemps possible en faisant exploser un maximum de murs et de fantômes, ce qui lui rapporte des points.

Pour ce faire, il pose des bombes, se met à l'abri le temps qu'elles explosent, et ramasse des upgrades.

Dans la SAE 1.01, vous allez développer la version à 1 joueur, en mode graphique. Dans la SAE 1.02, vous devrez développer une IA qui joue à une version du jeu à 4 joueurs.

Présentation des règles (version à 1 joueur)

Le jeu

Le jeu se présente sous la forme d'une fenêtre graphique, avec laquelle le joueur interagit par le clavier et/ou la souris. Le jeu est en tour par tour et non en temps réel, ce qui implique en particulier que le joueur dispose du temps qu'il veut entre chaque déplacement ou action. Après chaque action, l'environnement réagit, et ainsi de suite jusqu'à la fin de la partie.

La carte

Le jeu se déroule sur une grille rectangulaire de taille variable, dont les cases sont repérées par des coordonnées (x,y). Une case de la carte peut être vide, ou contenir un des éléments fixes suivants:

- un mur (M)
- une colonne (C)
- une prise ethernet (E)

Les murs empêchent de passer mais on peut les faire exploser. Les colonnes, elles, sont faites de metal et industructibles. Les prises ethernet sont des générateurs de fantômes (ces prises archaïques n'existent plus depuis longtemps).

Les cases vides peuvent contenir les éléments suivants :

- le bomber (P)
- un fantôme (F)
- un upgrade (U)
- une bombe (B)

Les cases contenant des murs, des colonnes, une prise ethernet, le bomber ou un fantôme sont considérées *bloquantes*, on ne peut pas entrer dessus (valable pour le bomber et les fantômes). Les upgrades ne sont pas bloquants (on peut entrer sur la case pour les prendre), et les bombes non plus (on peut passer sur leur case, à ses risques et périls). Les seuls éléments pouvant être superposés sont les bombes qui peuvent se trouver sur la même case qu'un fantôme ou que le bomber.

Au début du jeu, un fichier de scénario décrit l'état de la map en dbut de jeu avec le codage des lettres ci-dessus, par exemple :

```
CCCCCCCCC
C   P     C
CMCMCMCMC
CMMMM E C
CCCCCCCCC
```

Ce fichier précise également le *timer global* (nombre de tours restant) ainsi que le *timer fantôme* (qui détermine la fréquence de leur apparition).

En général au premier tour il n'y a pas encore de fantômes et d'upgrades. Comme dans le jeu bomberman, les cases d'ordonnée et abcsisse paires seront en général des colonnes indestructibles, ainsi que tout le tour de la map.

Le bomber

Il est caractérisé par sa position dans la carte ainsi que les attributs suivants :

- ses points de vie (PV), 3 au départ, qui peuvent fluctuer au cours de la partie. Il n'y a pas de maximum, et le minimum est de 0. Si 0 est atteint, la partie est terminée.
- son niveau (Niv), 0 au départ.
- La portée de ses bombes, égale à 1+ Niv/2

Le temps

Le jeu se passe en une série de tours, de la façon suivante :

1. décision de l'action du bomber
2. résolution de l'action
3. déplacement des fantômes
4. attaque des fantômes
5. apparition de nouveaux fantômes
6. réduction des timers et les explosions

1,2. Actions du Bomber

Les actions disponibles sont :

- les déplacements dans les 4 directions, qui sont possibles si la case de destination est non-bloquante.
- poser une bombe, possible si la case du bomber n'en contient pas déja une.
- ne rien faire

La conséquence d'un déplacement est que :

- le bomber se déplace sur la case
- si la case contient un upgrade, il est consommé.

La conséquence du fait de poser une bombe place une nouvelle bombe sur la case, avec son timer fixé à 5.

3. Déplacement des ennemis

Un fantôme se déplace chaque tour d'une case, aléatoirement, *sauf s'il a un bomber adjacent, auquel cas il reste sur place*. Il se déplace uniquement sur les cases non-bloquantes. Pour déterminer dans quelle direction se déplace un fantôme, on regarde les cases voisines non bloquantes.

- S'il n'y aucune case voisine non-bloquante, le fantôme ne bouge pas
- S'il y a une seule case voisine non-bloquante, le fantôme se déplace sur cette case
- S'il y a au moins deux cases non bloquantes, on retire de ces cases la case occupée par le fantôme au tour précédent, afin d'éviter qu'il revienne sur ses pas, et on tire au hasard uniformément parmi les cases restantes.

Les bombers ne ramassent pas les upgrades, ils passent dessus sans les prendre.

4. Attaque des fantômes

Si à cet instant, un fantôme est adjacent au bomber, ce dernier perd un point de vie.

5. Apparition de nouveaux fantômes

Les fantômes sont générés par les prises ethernet. Le scénario chargé donne le timer d'apparition des fantômes, par exemple 20. Ceci signifie que de nouveaux fantômes doivent apparaître tous les 20 tours, à compter du tour 1 (donc les fantômes apparaissent aux tours 20, 40, 60 etc) même s'il y a déjà des fantômes en jeu.

Lorsque les fantômes doivent apparaître, on détermine les cases non bloquantes au voisinage de chaque prise ethernet, et on en choisit une au hasard pour faire apparaitre un fantôme (soit un par prise ethernet). S'il n'y a pas de cases non-bloquantes disponibles, aucun fantôme n'apparait pour cette prise cette fois-ci (ce sera le cas si la map est saturée de fantômes).

6. Timers et explosions

Le timer global est décrémenté de 1. S'il arrive à 0, le jeu est fini et le score est donné.

Ensuite, le timer de chaque bombe est décrémenté de 1. Si le timer d'une bombe arrive à zéro, la bombe explose. Quand une bombe explose, la case de la bombe ainsi que les 4 cases voisines subissent l'explosion. L'effet d'une explosion dépend de ce qui se trouve dans la case:

- sur une colonne ou une prise ethernet, aucun effet
- sur un mur, il est détruit (la case devient vide) et le bomber marque 1 point
- un upgrade est détruit
- un fantôme est détruit et à sa place on place un upgrade
- un bomber perd un point de vie
- une autre bombe qui subit une explosion va exploser à son tour, une fois tous les effets ci-dessus résolus pour la première bombe. On peut ainsi avoir des *explosions en chaîne*.

Upgrades

Quand un bomber entre sur une case contenant un upgrade, l'upgrade disparait et le bomber gagne un niveau et marque 1 point. L'effet des niveaux est le suivant

- niveau 0 : c'est le niveau de départ, rien de spécial
- niveau 1 : le bomber gagne 1 PV
- niveau 2 : le bomber gagne 1 en portée
- niveau 3 : +1 PV
- niveau 4 : +1 portée (etc) L'effet de la portée est que toutes les bombes posées par le bomber vont exploser plus loin que 1 case. Par exemple, avec une portée 3, les bombes vont exploser suivant le schéma ci-dessous :

```
   E
   E
   E
EEEEEEE
   E
   E
   E
```

Remarquez que les bombes n'explosent pas en diagonale. Attention : les colonnes sont bloquantes pour les explosions et l'explosion se propagera sur la portée complète uniquement si elle n'est pas bloquée par une colonne. Par exemple dans la situation suivante, avec une portée 2 :

```
 .
 .
CB.C
 C
```

alors la zone d'explosion de la bombe est

```
 E
 E
CEEC
 C
```

Objectifs d'implémentation

L'objectif final idéal de votre projet est d'avoir une version graphique du jeu, qui implémente de façon exacte toutes les règles, avec des options plus ou moins avancées de votre choix. Des objectifs plus raisonnables sont détaillés par les niveaux suivants. Essayez de réaliser complètement un niveau (modèle ou graphique) avant de passer au suivant, et de façon générale donnez priorité au modèle avant le graphique.

Modèle niveau 1

On peut déplacer le personnage sur la map. Détection des cases bloquantes ou non. Pour cette version et les versions ci-dessous, si vous n'avez pas fait les graphiques, cela peut fonctionner dans le terminal et l'affichage se fait dans le terminal avec un système de lettres.

Modèle niveau 2

On ajoute la génération des fantômes, leur déplacements et leurs attaques.

Modèle niveau 3

On ajoute la gestion des explosions, des upgrades et autres détails des règles.

Options du modèle

Une fois les trois niveaux réalisés entièrement, vous pouvez ajouter des options. Par exemple :

- d'autres types de bombes
- d'autres types d'ennemis avec des comportements variés (par exemple, qui fonce directement sur le perso
- d'autres upgrades
- un système de téléportation

Vous pouvez ajouter ce que vous voulez. Attention, ajouter ces options implique que le modèle niveau 3 soit réalisé et qu'on puisse lancer le jeu en mode *vanilla*, c'est à dire sans les options.

Graphique niveau 1

Une interface graphique basique et fonctionnelle avec des formes de tkiteasy.

Graphique niveau 2

Vous ajoutez la vision des timers, du score.

Graphique niveau 3

Vous utilisez des images plus avancées et réalisez un jeu qui claque un peu visuellement.

Options graphiques

A partir du niveau 2, vous pouvez ajouter des options, par exemple une interface graphique permettant de charger le scénario voulu, l'accès aux high-scores, une progression dans une campagne avec des scénarios qui s'enchaînent etc.

Recommandations

La partie graphique

- Le moteur de jeu ou modèle (la partie logique et données) doit être séparé de la partie graphique au niveau des fichiers.
- Pensez ergonomique dans les contrôles du jeu. Pensez aussi que le jour du test on doit arriver rapidement au jeu sans entrer des informations au clavier dans le terminal comme le nom du joueur, etc.
- N'utilisez pas d'autre bibliothèque graphique que tkiteasy ou tkinter. Le code doit tourner directement sur les machines de l'IUT.
- Respectez l'ordre des niveaux. Un objectif basique est d'atteindre le niveau 2 ou 3 en modèle et 1 en graphique.

Le style du code

- Utilisez des variables globales constantes pour les dimensions de la fenêtre, la taille des cases etc.
- Votre code doit être découpé en fonctions, et hormis les constantes indiquées ci-dessus ne doit pas utiliser de variable globale. *Si du code est dupliqué, c'est qu'il fallait utiliser une fonction avec un paramètre pour indiquer ce qui change*. Idéalement, une fonction fait moins de 15 lignes.
- Vos variables et fonctions doivent avoir des noms intelligents qui permettent de comprendre.
- Commentez les grandes étapes du code sans le paraphraser, et mettez toutes les spécifications (docstring) des fonctions.

Votre travail

- Le travail en binôme est obligatoire. Le travail en projet est une occasion de beaucoup progresser, aussi ne faites pas tout, et ne laissez pas l'autre tout faire ! Vous seriez également pénalisés lors de l'évaluation où nous nous assurons que **chacun des deux maîtrise l'ensemble du projet** par des questions (la réponse "ce n'est pas moi qui ai développé cette partie" *ne sera pas acceptée* vous pénalisera individuellement).
- Vous suivrez la progression par niveaux proposée dans ce sujet afin que votre programme soit bien structuré. Nous préférons un programme bien écrit, stable, et qui va moins loin dans les niveaux, plutôt qu'un programme très sophistiqué mais mal conçu, pas clair et et plein de bugs. La note dépend largement autant de votre maîtrise et la clarté/conception du code que de l'application en elle-même.
- S'aider d'un groupe à l'autre est possible, recopier un code tel quel est interdit. Nous avons des moyens informatiques de comparaison des codes (JPlag, entre autres). Le code de votre binôme doit être le vôtre, et chacun.e des membres du binôme doit être capable de l'expliquer intégralement, que ce soit « votre partie » ou non, comme déjà dit (mais on insiste).
- De même, si chatgpt ou autre IA vous fournit du code, vous avez intérêt à bien le maîtriser.

Les échéances

- Vous déposez votre projet sur le dépot e-campus sur le cours *Initiation au Développement* avant le **LUNDI 6 JANVIER 2025 à 20h00**. Il doit être accompagné d'un README de quelques pages permettant de le prendre en main (comment le lancer, quelles sont les commandes, etc).
- Des soutenances de projet auront lieu dans la ou les semaines suivantes. Environ 10 à 15 minutes par binôme, nous regardons les projets, le code des projets, et votre maîtrise de ce code.

Les compétences qui seront évaluées sur ce projet seront les suivantes (il peut y avoir des changements mineurs) :

- Respect du cahier des charges de dépôt (deadline, README, archive nettoyée et au bon format, programme qui fonctionne directement sans erreurs et sans rien installer de nouveau sur les machines de l'IUT)
- Respect du cahier des charges du programme : jusqu'où vous êtes allés dans les niveaux.
- Spécifications et tests: chaque fonction doit documenter sa spécification, et les fonctions critiques du modèle doivent être accompagnées de leur fonction de test (test unitaire) dans un fichier séparé.
- La qualité du code : commentaires, découpage en fonctions, constantes, nommage correct…
- Considérer tous les cas particuliers d'un programme (ex: tel paramètre fait-il planter le programme?). Votre programme doit être robuste.
- Les options ajoutées, modèle et graphiques.
- Incorporer des solutions techniques innovantes : vous avez utilisé des algorithmes ou structures de données malignes.

Et maintenant ?

Go ! N'hésitez surtout pas à questionner vos enseignant.e.s si vous avez des questions. De plus, un salon dédié à la SAE permettra de dialoguer sur Discord. Bon travail ! Amusez-vous bien…