Projet : Casse-briques

Le but de ce projet est de réaliser un jeu de casse-briques (si vous ne savez pas de quoi il s'agit, Google est votre ami) où l'utilisateur contrôle une raquette horizontale qui doit renvoyer une balle vers un mur de briques qui se cassent lorsque la balle les touche. Chaque niveau consiste en un mur différent, et le joueur réussit un niveau lorsqu'il a éliminé toutes les briques. Il perd la partie s'il n'arrive pas à rattraper la balle.

1 Le programme à réaliser

1.1 Le jeu de base

Dans sa version de base, le jeu doit permettre de jouer un unique niveau de dimension fixe, dans lequel les briques sont générées aléatoirement (leur taille s'adapte de façon à remplir toute la largeur de la fenêtre et environ le tiers supérieur en hauteur). Les briques doivent également avoir un niveau de *résistance* (entre 1 et 3) correspondant au nombre de fois qu'il faut toucher la brique pour la détruire. Le but est de finir ce niveau sans laisser la balle s'échapper. La raquette est déplacée uniquement au clavier.

Pour réaliser cette partie, voici l'ordre des étapes à suivre :

1. Commencez par créer une fenêtre dont les dimensions (`hauteur` x `largeur`) devront être facilement modifiables, et dans laquelle une balle va être lancée vers le bas et devra rebondir correctement sur les 4 bords. Attention, pour la balle, vous allez devoir choisir sa vitesse de déplacement, c'est-à-dire définir de combien de pixels elle se déplace entre deux rafraîchissements de l'écran de jeu (c'est le vecteur vitesse de la balle), afin que le jeu soit jouable sans être trop difficile. **Cette vitesse doit absolument être aisément paramétrable**, car l'effet réel peut être très différent d'une machine à l'autre. Vous choisirez donc une vitesse par défaut et laisserez la possibilité à l'utilisateur d'en choisir une autre en paramètre de la ligne de commande.
2. Ensuite, ajoutez une raquette ainsi que les déplacements latéraux de cette raquette : la raquette devra se déplacer vers la gauche quand l'utilisateur appuiera sur la touche ←, et vers la droite quand il appuiera sur la touche →. La raquette ne peut pas sortir de la zone de jeu, et il faut que la balle puisse rebondir dessus. Désormais, la balle ne peut plus rebondir sur le côté inférieur de la fenêtre et un message "Perdu" doit s'afficher quand la balle s'échappe par ce côté.
3. Ajoutez maintenant un mur de briques aléatoire "plein" à votre zone de jeu. **Ce mur doit obligatoirement être encodé à l'aide d'une *liste* (et non d'une matrice !) de briques.** Le format de représentation des briques est libre, mais prévoyez qu'il faudra être capable de représenter la position d'une brique ainsi que plusieurs propriétés, comme leur résistance, une couleur, un nombre de points, le fait qu'elles libèrent ou non des bonus lors de leur destruction, ...
4. Ajoutez la gestion des rebonds sur ces briques.
5. Ajoutez maintenant la gestion de la destruction des briques à votre programme. La *résistance* d'une brique indiquera le nombre d'impacts qu'elle doit subir avant de disparaître ; il faudra indiquer le changement de résistance de chaque brique dans la zone

de jeu lors des impacts (par exemple : un changement de couleur, ou un éclaircissement de la brique). En outre, il faudra indiquer à l'utilisateur qu'il a gagné la partie quand la dernière brique aura été détruite.

6. Vérifiez que la balle se comporte correctement dans les espaces libres créés par la destruction des briques.
7. Ajouter un mode où l'ordinateur joue tout seul, que l'on peut lancer avec une option spéciale de la ligne de commande. Cela permet de tester plus facilement, avec la raquette qui suit la balle et une vitesse plus élevée, par exemple. Cette étape peu être réalisée plus tôt, pour permettre des tester facilement en cours de développement.

1.2 Les bonus et améliorations

Une fois que le jeu de base fonctionne parfaitement (et pas avant), vous pourrez commencer à ajouter des bonus et améliorations parmi ceux de la liste suivante (si vous voulez en faire d'autres, consultez votre enseignant avant) :

1. Ajouter des "bonus/malus" contenus dans les briques. Ce sont des objets qui tombent depuis l'endroit où une brique est détruite et qui doivent être attrapés par la raquette pour être activés. Les effets peuvent être :
   — faire des bonus de points si on calcule un score ;
   — agrandir/rétrécir la raquette (en largeur) ;
   — donner des balles supplémentaires (lorsque l'on perd la balle, une nouvelle balle est redonnée) ;
   — donner une multi-balle (plusieurs balles simultanément dans la fenêtre de jeu) ;
   — supprimer une balle ;
   — rajouter/détruire des briques ;
   — augmenter/diminuer la vitesse de déplacement de la balle ;
   — ...
2. Ajouter une zone d'information à la fenêtre de jeu pour afficher diverses informations : un score, un temps écoulé, le fait d'avoir droit à des balles supplémentaires, etc...
3. Ajouter la possibilité de mettre de l'effet sur la balle lorsque l'on déplace la raquette rapidement quand elle la touche, ou qu'elle touche le coin de la raquette, par exemple. Remarque : pour cette optimisation, il sera probablement nécessaire de pouvoir déplacer la raquette avec la souris plutôt qu'avec les touches du clavier.

1.3 La version avancée du jeu : édition de niveau, sauvegardes, ...

Dans un deuxième temps, vous devrez fournir une version du jeu permettant de créer un nouveau niveau de jeu à partir d'un fichier texte contenant la description des briques dans leur état initial. Vous choisirez un encodage pour les briques et leurs propriétés et on écrira dans le fichier les codes des briques dans l'ordre de leur position dans la fenêtre.

Par exemple, le fichier texte de gauche ci-dessous pourra correspondre à un mur de 11 briques de large et 4 briques de haut avec une brique sur deux seulement sur la ligne la plus basse. Les numéros indiquent la résistance des briques. Le fichier de droite montre un second exemple, où l'on laisse une colonne vide de part et d'autre du mur représenté :

```
11
4

1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3 3
3 . 3 . 3 . 3 . 3 . 3
```

```
13
4

. 1 1 1 1 1 1 1 1 1 1 1 .
. 2 2 2 2 2 2 2 2 2 2 2 .
. 3 3 3 3 3 3 3 3 3 3 3 .
. 3 . 3 . 3 . 3 . 3 . 3 .
```

Remarque : si vous avez implémenté la possibilité d'avoir des bonus/malus associés à certaines briques, vous devrez rajouter de quoi les encoder dans le fichier texte.

Vous pourrez ensuite implémenter une ou plusieurs des fonctionalités avancées parmi les suivantes :

- Rajouter un écran de sélection de niveau et la possibilité d'enchaîner plusieurs niveaux.
- Rajouter de quoi mémoriser/afficher des *high scores*.
- Rajouter la possibilité de mettre en pause et de sauvegarder l'état du jeu à ce moment-là, ainsi que la possibilité de charger une partie sauvegardée en début de jeu.
- Rajouter un fichier de configuration pour le jeu afin de fixer différents paramètres (taille de la fenêtre, vitesse de la balle, type de déplacement (souris/clavier), score ou non, ...)

2 Consignes de rendu

Le projet se déroulera en deux grandes phases : un premier rendu aura lieu fin novembre 2017, et un second rendu début janvier 2018. Pour les deux rendus, il sera impératif de suivre les consignes précisées ci-dessous, sans quoi vous perdrez des points. En particulier :

- Ce projet est à faire en binôme (s'il y a besoin de faire une exception, contactez les enseignants). Vous devrez selectionner votre binôme sur e-learning dans la semaine suivant la publication du sujet.
- **Vous DEVEZ utiliser** `upemtk`. L'utilisation de modules autres que les modules standards de Python est interdite ; en cas de doute, n'hésitez pas à poser la question.
- **Votre programme devra impérativement s'exécuter sans problème sur les machines de l'IUT.** Prévoyez donc bien de le tester sur ces machines en plus de la vôtre et d'adapter ce qui doit l'être dans ce cas (par exemple, les vitesses de déplacement). Il sera donc utile de permettre à l'utilisateur de préciser certaines options auxquelles vous attribuez des valeurs par défaut plutôt que de devoir modifier le code à chaque exécution.

2.1 Premier rendu

L'objectif général à atteindre pour le premier rendu est d'avoir un jeu fonctionnel et robuste avec :

- Le jeu de base (section 1.1) complet.
- Au moins un bonus de la liste donnée en section 1.2.

Remarque : vous ne devez pas commencer les améliorations de la section 1.3 avant d'avoir parfaitement rempli l'objectif ci-dessus.

La date limite pour ce premier rendu est le **vendredi 24 novembre 2017 à 23h55**. Passé ce délai, la note attribuée à votre projet sera 0. Les séances de TP de la semaine suivante seront consacrées à la vérification de votre rendu et à amorcer la seconde partie. Vous déposerez sur la plate-forme e-learning une archive contenant :

1. **les sources de votre projet**, commentées de manière pertinente (quand le code s'y prête, pensez à écrire les doctests). De plus :
   — les fonctions doivent avoir une longueur raisonnable ; n'hésitez pas à morceler votre programme en fonctions pour y parvenir ;
   — plus généralement, le code doit être facile à comprendre : utilisez des noms de variables parlants, limitez le nombre de tests, et veillez à simplifier vos conditions ;
   — quand cela se justifie, regroupez les fonctions par thème dans un même module ;
   — chaque fonction et chaque module doit posséder sa *docstring* associée, dans laquelle vous expliquerez le fonctionnement de votre fonction et ce que sont les paramètres ainsi que les hypothèses faites à leur sujet.
2. un **fichier texte** `README.txt` expliquant :
   — les optimisations qui ont été implémentées : on s'attend à ce que vous en réalisiez au moins une ;
   — l'organisation du programme,
   — les choix techniques et
   — les éventuels problèmes rencontrés.
   Vous pouvez y ajouter tous les commentaires ou remarques que vous jugez nécessaires à la compréhension de votre code.

Si le code ne s'exécute pas, la note de votre projet sera 0. Vous perdrez des points si les consignes ne sont pas respectées, et il va sans dire que si deux projets trop similaires sont rendus par 2 binômes différents, la note de ces projets sera 0.

2.2 Second rendu

L'objectif pour le second rendu est toujours d'avoir un jeu fonctionnel et robuste avec le jeu de base et au moins un bonus. Vous devez, en plus avoir rajouté
— La lecture d'un niveau de jeu dans un fichier (section 1.3).
— Au moins une des fonctionalités avancées proposées dans la liste de la section 1.3.

Les consignes données pour le premier rendu restent valables pour le second rendu. La date limite pour le second rendu est le **7 janvier 2018 à 23h55**. En outre, des mini-soutenances seront organisées quelques jours après le rendu, pendant lesquelles vous ferez une démonstration sur une machine des salles de TP et serez interrogés sur le projet (les choix techniques/algorithmiques que vous avez faits, les difficultés rencontrées, l'organisation de votre travail, ...). Les contributions de chaque participant seront évaluées, et il est donc possible que tous les participants d'un même groupe n'aient pas la même note.