mickael.hoerdt@hesge.ch 2015/2016

Mini-projet d'ingénierie de protocoles réseaux informatiques

Un jeu de snake multi-joueur en réseaux
Étape 3: Le protocole du jeu, le serveur et les clients

Une fois qu'on dispose du protocole SnakeChan et SnakePost implémentés en étapes 1 et 2, nous pouvons implémenter le protocole du jeu du Snake multiplayer en réseau. Celui-ci sera transporté comme le payload des deux protocoles implémentés dans les étapes précédentes, comme l'indique la figure 1. Les messages du protocole de jeu seront sérialisés/désérialisés en json (voir le package json en python).

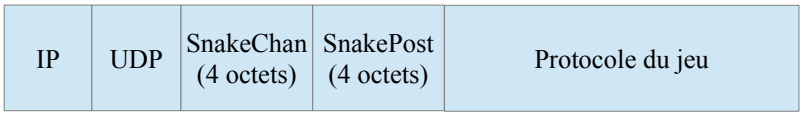

Figure 2 : Encapsulation du protocole de jeu dans SnakeChan et SnakePost

1. Phase de connexion

Les informations d'un nouveau joueur seront indiquées directement dans le protocole SnakeChannel par l'ajout de deux nouvelles clef valeur par le client : `color` et `nickname`. `nickname` est une chaine de caractère, `color` est une couleur correspondant à l'une des clefs du dictionnaire `pygame.color.THECOLORS` et est décodée à l'instanciation d'un nouveau snake par le client.

Exemple de communication :

Client :

```
SnakeChan    :OUT   -GetToken 2661843323 Snake
SnakeChan    :IN    -Token 2721555251 2661843323 19
SnakeChan    :OUT   -Connect
"\challenge\2721555251\protocol\19\color\yellow\nickname\mickael"
SnakeChan    :IN    -Connected 2721555251
SnakeChan    :CONNECTED!
```

Serveur :

```
SnakeChan    :IN    -GetToken 2661843323 Snake
SnakeChan    :OUT   -Token 2721555251 2661843323 19
SnakeChan    :IN    -Connect
"\challenge\2721555251\protocol\19\color\yellow\nickname\mickael"
SnakeChan    :OUT   -Connected 2721555251
```

Le serveur devra s'assurer que le nickname est unique car il est utilisé par les clients pour indexer l'état des autres joueurs. Si le nickname est déjà pris, le serveur corrigera l'erreur de lui-même avec une solution de votre choix.

2. Messages "foods"

Les messages "foods" contiennent la position la plus récente de toutes les pommes sur le terrain sous forme de liste de coordonnées [x,y]. Cette liste est envoyé de manière fiable par le serveur à tous les clients dans 3 situations :

- À chaque fois que celui-ci rajoute une pomme dans le jeu.
- Lorsque un nouveau joueur arrive dans la partie, il reçoit immédiatement la position des pommes actuelles.
- Lorsque le serveur détecte qu'un snake a mangé une pomme.

Exemple de format : `{'foods': [ [6, 24], [17, 32], [25, 8], [13, 17] ] }`
Transport snakepost : fiable, du serveur à tous les clients

3. Messages "body_p"

Les messages "body_p" contiennent la liste la plus récente des coordonnées [x,y] des éléments du corps du snake d'un joueur. Le premier élément de la liste corresponds à la tête du snake, le dernier élément à sa queue. Un message "body_p" est envoyé de manière non fiable d'un client au serveur à chaque nouveau déplacement du snake de ce client. A réception, le serveur met à jour cette position pour le snake qui corresponds au canal d'où le message à été reçu.

Exemple de format : `{'body_p':[ [20,10],[20,11],[20,12],[20,13],[20,14] ] }`
Transport snakepost : non fiable, d'un client vers le serveur.

4. Messages "snakes"

Les message "snakes_p" contiennent la liste la plus récente des positions du corps de tous les snakes dans la partie, préfixée par l'identifiant du joueur. Ces messages sont envoyés à période fixe du serveur à tous les clients, qui mettent à jour à la position de chaque joueur indiqué dans la liste à la réception.

Exemple de format :

```
{
   'snakes': [

              [ 'mickael', [ [20, 9], [20, 10], [20, 11], [20, 12], [20, 13] ] ],
              [ 'florent', [ [20, 3], [20, 4], [20, 5], [20, 6], [20, 7]] ]

              ]
}
```

Transport snakepost : non fiable, du serveur à tous les clients.

5. Messages "player_info"

Les messages "player_info" contiennent l'information la plus récente sur l'état des joueurs dans la partie. Ils contiennent dans l'ordre : le nom du joueur, sa couleur, son score, et un booléen qui indique si le joueur est prêt à jouer ou pas (voir le message "ready"). Un client qui reçoit un message player_info mets à jour les scores affichés sur l'interface graphique ainsi que l'état "ready" des snakes indiqués. Ces messages sont envoyés de manière fiable du serveur à tous les clients dans

trois situations :

- A la connexion d'un nouveau client, ou bien à son timeout (voir partie 9).
- Lorsque le score change, tel que calculé par le serveur
- Lorsque l'état "ready" d'un joueur passe de False à True

Exemple de format : `[['mickael','yellow', 4, False], ['florent','red', 3, False]]`
Transport snakepost : fiable, du serveur à tous les clients

6. Messages "game_over"

Le message "game_over" est envoyé par le serveur à tous les clients lorsque celui-ci détecte qu'un snake est entré en collision avec lui-même ou avec un autre : le score de ce snake est décrémenté de 1. Si le snake est rentré dans un autre snake, l'autre snake a un score incrémenté de 1. Ce message contient simplement le nom du joueur pour lequel c'est la fin de la partie. Un client qui reçoit un message "game_over" ne quitte pas le jeu, il recommence juste la partie depuis le début.

Exemple de format : `{'game_over': 'florent'}`
Transport snakepost : fiable, du serveur à tous les clients.

7. Messages "grow"

Le message "grow" est envoyé du serveur à tous les clients lorsque le serveur détecte qu'un snake est rentré dans une pomme. A réception, le client concerné augmente la taille du snake d'une constante.

Exemple de format: `{'grow': 'florent'}`
Transport snakepost : fiable, du serveur à tous les clients.

8. Messages "ready"

Le message "ready" est envoyé d'un client au serveur pour indiqué lorsqu'il est prêt à commencer la partie. Il contient le nom du joueur et un booléen. Tant qu'un joueur n'a pas son état « ready » à True, il ne peut ni manger des pommes, ni provoquer de collisions. Lorsque le serveur reçoit ce message, il met à joueur l'état du joueur est envoit un nouveau message "player_info" pour indique le changement aux autres joueurs.

Exemple de format : `{'ready': True}`
Transport snakepost : fiable, d'un client vers le serveur.

9. Déconnexion d'un joueur

Le protocole SnakeChannel ne dispose pas de message indiquant une deconnexion. Pourtant, les joueurs doivent pouvoir venir et partir du jeu à tout moment, le serveur mettant à jour cette information. Pour celà, le serveur de jeu considérera qu'un joueur n'est plus dans une partie après deux secondes sans avoir reçu de messages de mis à jour de sa part et renverra un message player_info en conséquence.

Travail à faire :

Implémenter le protocole du jeu qui permettent au serveur et aux clients de communiquer les différents états du jeu. Implémenter le serveur de jeu, ainsi que le client. Pour le moteur graphique et la gestion des états du jeu sur le client, on utilisera la version single player fournie[1].

Forme du rendu final des étapes 1,2 et 3 :

A rendre impérativement avant le **8/1/2016 23h59**. Tout retard sera sanctionné sur la note finale. Une archive nommée Snake_multiplayer_<nom1>_<nom2>.tar.gz envoyée par mail sur mickael.hoerdt@hesge.ch avec le sujet « Rendu final Réseau I snake ».

Le mail devra contenir l'url d'une vidéo de présentation de maximum **10 min** qui démontre le fonctionnement de votre implémentation, en particulier lorsqu'il y a des pertes de paquets sur le réseau.

L'archive en pièce jointe devra contenir :

- L'ensemble du code avec un fichier README qui explique comment l'exécuter.
- Un rapport de maximum **10 page**s au format pdf qui documente la partie technique l'implémentation des étapes 1, 2 et 3 et les bugs connus, ainsi u

Bonus : Sécurisation du protocole Snakechannel

L'inconvénient du protocole SnakeChannel est qu'il est très facile de forger un packet pour insérer des fausses données dans le canal. Documenter et implémenter une solution qui permette de remédier à ce problème en utilisant le Token du protocole comme une clé partagée sur le canal.

1https://infolibre.ch/hepia/reseauxI-2015-2016/snake_single_player.tgz