Remote Snake API

May 8, 2019

1 General Description

- All transmissions will be based on UDP
- All UDP packet will contain plain json data

2 Communications

2.1 Initialisation

1. Server wait for a client
2. Client can send:

```
{
    "type": "new-game"
}
```

3. Server can reply:

```
{
    "type": "state",
    "syn": 1,
    "game-id": 1,
    "game-over": false,
    "snake": [(1,2),(1,3)],
    "food": [(6,7)]
}
```

Note that, syn entry is used to keep packet ordering consistent and detecting packet inversion on the network. Thus, syn entry indicate the expected syn that the client should send on the next UDP packet.

2.2 Gameplay

2.2.1 Change Direction

1. When client is playing a game it can ask to the server to change snake direction:

```
{
    "type": "update",
    "syn": 1,
    "game-id": 1,
    "direction": "left",
}
```

2. Server can reply

```
{
    "type": "state",
    "syn": 2,
    "game-id": 1,
    "game-over": false,
    "snake": [(0,2),(1,2)],
    "food": [(6,7)]
}
```

2.2.2 Refresh Screen

1. When no key are press (the snake is simply going straigth forward). So, client can send:

```
{
    "type": "update",
    "syn": 2,
    "game-id": 1,
    "direction": null
}
```

2. Server can reply:

```
{
    "type": "state",
    "syn": 3,
    "game-id": 1,
    "game-over": false,
    "snake": [(1,2),(0,2)],
    "food": [(6,7)]
}
```

2.2.3 End Game

- When game is over server will send the following state message (switch game-over to true):

```
{
    "type": "state",
    "syn": null,
    "game-id": 1,
    "game-over": true,
    "snake": [(0,2),(1,2)],
    "food": [(6,7)]
}
```

- No reply is expected from the client and server will be in charge to free local memory. Note that syn=null.