Rétrojeux synchrones en HTML5

Pierre-Évariste Dagand

Les rétrojeux (*retrogaming* en anglais) étaient joués dans les salles d'arcade du millénaire précédant. En exploitant quelques kilooctets de mémoire vive et un processeur cadencé à quelques megahertz, il était alors possible de jouer à des jeux désormais classiques comme Pong [1], Snake [2] ou Space Invaders [3]. Ces jeux étaient écrits en assembleur afin d'obtenir les meilleures performances possibles et de minimiser la taille des structures de données.

Aujourd'hui, un simple téléphone portable dispose de mégaoctets de mémoire vive et d'un voire deux processeurs cadencés à plusieurs gigahertz. Cette profusion de ressources nous permet de revisiter la conception de tels jeux : les performances passent au second plan tandis que la facilité de développement et de déploiement passe au premier plan.

Le standard HTML5 [4] simplifie le déploiement d'applications dynamiques. Les navigateurs Web modernes permettent de faire interagir des programmes écrits en Javascript avec le contenu d'une page Web. En particulier, l'interface de `canvas` [5, 6] offre la possibilité de dessiner des objets, les animer et d'intercepter des entrées (clavier, souris) dans une page Web.

L'API `canvas` impose une programmation ***réactive*** : afin de traiter la mise à jour du `canvas` et traiter les entrées, le programmeur doit enregistrer une fonction de rappel (*callback* en anglais) par le truchement de la méthode `requestAnimationFrame` [7]. Cette fonction sera appelée à chaque évènement, duquel elle pourra réagir par une action.

Ce modèle réactif est couramment utilisé dans le domaine des ***systèmes de contrôle-commande***. C'est ainsi que sont programmés les commandes de vol des avions de ligne : l'ordinateur de bord est sollicité périodiquement par les mesures des capteurs de l'avion et les commandes des pilotes. En retour, il effectue des actions sur les gouvernes et les réacteurs.

Afin de simplifier le développement de ces applications (dont la correction est critique), des langages spécialisés ont été créés. Ceux-ci reposent sur la théorie des ***langages flots de données synchrones*** : abstraitement, on considère qu'un tel programme traite un flot de données (mesures des capteurs et commandes de l'utilisateur) et produit un flot de données qui correspondent à des instructions qui seront transmises aux actuateurs.

Objectif : Dans ce stage, nous nous intéresserons à l'application de ce formalisme pour la programmation de rétrojeux. Tout d'abord, on se familiarisera avec l'API HTML5 en développant un ***clone du jeu Snake*** directement dans le langage Javascript. Fort de cette expérience, on définira ensuite un langage flot de données synchrone intégrant l'API `canvas` du standard HTML5. On s'intéressera alors à la ***compilation*** de ce langage vers Javascript. Enfin, on validera notre approche en implémentant un second rétrojeu (dont le choix est laissé libre à l'étudiant).

Profil recherché : Le candidat devra être intéressé par les aspects pratiques et théoriques liés à la programmation et à la conception de langages de programmation. Une certaine aisance dans un langage fonctionnel (Javascript, OCaml, Scheme, …) est attendue.

Encadrement : Ce stage aura lieu au LIP6, au sein de l'équipe Whisper (Upmc). Il sera encadré par Pierre-Évariste Dagand, chargé de recherche au Cnrs.

Références

[1] Wikipedia. Pong, November 2016. URL https://en.wikipedia.org/wiki/Pong.

[2] Wikipedia. Snake, November 2016. URL https://en.wikipedia.org/wiki/Snake_%28video_game%29.

[3] Wikipedia. Space invaders, November 2016. URL https://en.wikipedia.org/wiki/Space_Invaders.

[4] Mozilla Developper Network. Html5 developer guide, November 2016. URL https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5.

[5] Mozilla Developper Network. Canvas api, November 2016. URL https://developer.mozilla.org/en-US/docs/Web/HTML/Element/canvas.

[6] Mozilla Developper Network. Canvas tutorial, November 2016. URL https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial.

[7] Mozilla Developper Network. requestAnimationFrame api, November 2016. URL https://developer.mozilla.org/en-US/docs/Web/API/window/requestAnimationFrame.