TP2 : Dart et Polymer

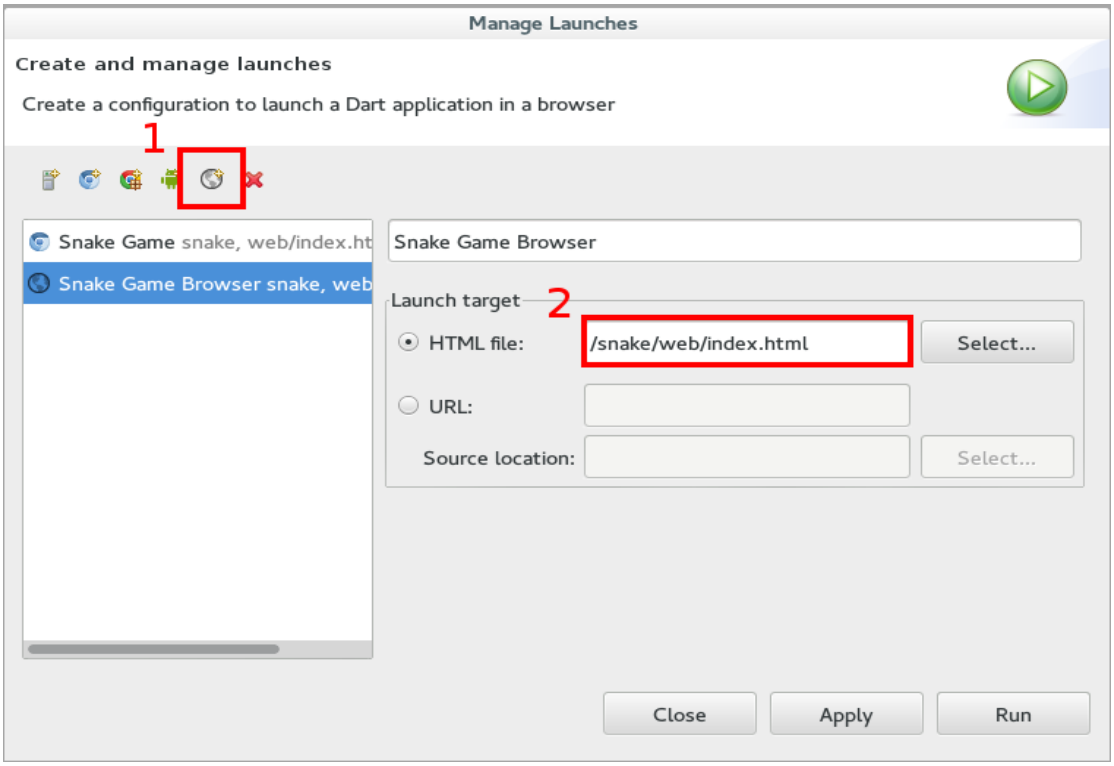
L'objectif de ce TP est de découvrir deux technologies web: Dart et Polymer. Dart est un langage de programmation développé par Google qui a pour objectif de remplacer Javascript. Polymer est un framework Javascript permettant la création et l'utilisation de composants web.

Mise en place

Pour ce TP, nous utiliserons l'éditeur proposé par Dart qui est disponible à cette adresse : https://www.dartlang.org/.

Le code de base du TP se trouve sur github : https://github.com/gbecan/teaching-jxs-tp2

Pour pouvoir exécuter le projet, il faut créer une nouvelle configuration (Create a new browser launch) dans le menu Run /Manage Launches. Ensuite on sélectionne le fichier HTML principal du projet (index.html). **Cette opération n'est à faire qu'une seule fois.**



Dart

Le but de ce TP est d'implémenter une version du célèbre jeu Snake (http://playsnake.org/) avec Dart. Contrairement à JavaScript, Dart propose un langage orienté objet à base de classes avec vérification de types. Il est important de noter que Dart reste un langage dynamique. Ces informations de typages sont optionnelles et utilisées uniquement lors de la compilation ou le debug. Lors de l'exécution, les types sont oubliés.

Le code de base de ce TP est composé des fichiers suivants :

- *index.html* décrit la structure de l'application
- *main.dart* contient le code lancé au démarrage de l'application
- *SnakeGame.dart* contient la méchanique du jeu. Il est completé par les fichiers *snake-game.html* et *snake-game.css* qui encapsulent le jeu dans un composant Polymer.

La première étape de l'implémentation du jeu est de modéliser le serpent ainsi que son déplacement. L'affichage ainsi que le déplacement du serpent se feront sur une grille dont la taille des cases est définie par l'attribut *grid* de la classe SnakeGame.

Q1 : Créer une classe *SnakePart* pour représenter les coordonnées d'une partie du corps du serpent.

Q2 : Créer une classe *Snake* contenant une liste de SnakePart afin de modéliser le serpent. Modifier aussi la classe SnakeGame afin d'initialiser le serpent avec une longueur de 3.

Q3 : Ajouter une méthode *move* à la classe Snake pour gérer son déplacement. L'interaction utilisateur se fait via la méthode *changePosition* de la classe SnakeGame. Cette méthode est à compléter pour répercuter les commandes de l'utilisateur sur le serpent. Compléter aussi la méthode *updateGame* dans la classe SnakeGame afin d'appeler la méthode *move* à chaque acutalisation du jeu.

Q4 : Ajouter une méthode *draw* à la classe Snake pour dessiner le serpent dans le canvas.

À cette étape, le joueur doit pouvoir déplacer le serpent dans l'espace de jeu matérialisé par le canvas. Nous allons continuer l'implémentation avec la modélisation de la nourriture qui permet au serpent de grandir ainsi que l'implémentation des règles du jeu.

Q5 : Créer une classe *Food* pour représenter les coordonnées de la nourriture à attraper.

Q6 : Ajouter une méthode *relocate* qui déplacera la nourriture de manière aléatoire sur l'espace de jeu. On prendra soin de ne pas déplacer la nourriture sur le serpent. Modifier aussi la classe SnakeGame afin d'initialiser la position de la nourriture.

Q7 : Ajouter une méthode *grow* à la classe Snake afin de gérer l'agrandissement du serpent à chaque

fois qu'il attrape de la nourriture.

Q8 : Implémenter les règles du jeu dans la méthode *updateGame* de la classe SnakeGame. Pour rappel, la tête du serpent ne doit pas sortir de l'espace de jeu dont la taille est défini par les attributs *width* et *height* de la classe SnakeGame. De plus, le serpent ne doit pas rentrer en collision avec son propre corps. À chaque nourriture attrapée, le serpent grandit et le score est augmenté de 1.

Polymer

Polymer permet de définir de nouvelles balises HTML (composant Polymer). Par défaut, la structure et le style ces balises ne peuvent pas être ni accédées ni modifiées depuis l'extérieur. Cela garantit le même comportement sur n'importe quelle page et permet de réutiliser le composant sans se préoccuper des effets de bords d'autres parties de la page.

Le jeu que vous venez de développer est déjà encapsulé dans un composant Polymer. En s'inspirant de ce composant, nous allons développer un nouveau composant qui affichera le dernier score ainsi que les 5 meilleurs scores d'un jeu. Ce composant sera générique afin de pouvoir le réutiliser pour d'autres jeux.

Pour implémenter le composant suivant, nous exploiterons les capacités de data-binding de Polymer avec en particulier la directive *repeat*. Pour faire fonctionner ces capacités avec Dart, il faut annoter l'attribut que l'on souhaite afficher avec @observable. Plus de précisions sont disponibles dans le lien suivant :

https://www.polymer-project.org/0.5/docs/polymer/databinding.html

Q9 : Créer un composant score-panel affichant le dernier score ainsi que les 5 meilleurs scores. *Ce* composant proposera une unique méthode *newScore* qui permettra de renseigner le score d'une nouvelle partie.

Afin de faire communiquer notre jeu de snake avec le composant score-panel, nous allons utiliser un évènement. Les évènements sont très utiisés en programmation web et permettent par exemple de détecter que l'utilisateur à appuyer sur un bouton (évènement onclick). Polymer facilite l'envoi d'évènements personnalisés via la méthode *fire(nomDÉvènement, détails).*

Q10 : Modifier la classe SnakeGame afin de lancer un évènement à chaque fin de partie.

Q11 : Dans le fichier main.dart, lier l'évènement précédemment créé au composant score-panel afin de déclencher la mise à jour du score.

Bonus

Q11 : Créer un nouveau type de nourriture bonus faisant augmenter le score de 5 au lieu de 1. Ce nouveau type de nourriture apparaîtra de manière aléatoire et pour un temps limité. Utiliser l'héritage en Dart pour réaliser l'implémentation de cette nouvelle classe.

Q12 : Créer plusieurs niveaux de difficultés. Par exemple, le niveau facile ne prendra pas en compte la collision du serpent avec lui même et la vitesse sera diminuée. Le niveau difficile aura une vitesse élevée et ne comportera pas de nourriture bonus. Ici aussi, nous utiliserons l'héritage.