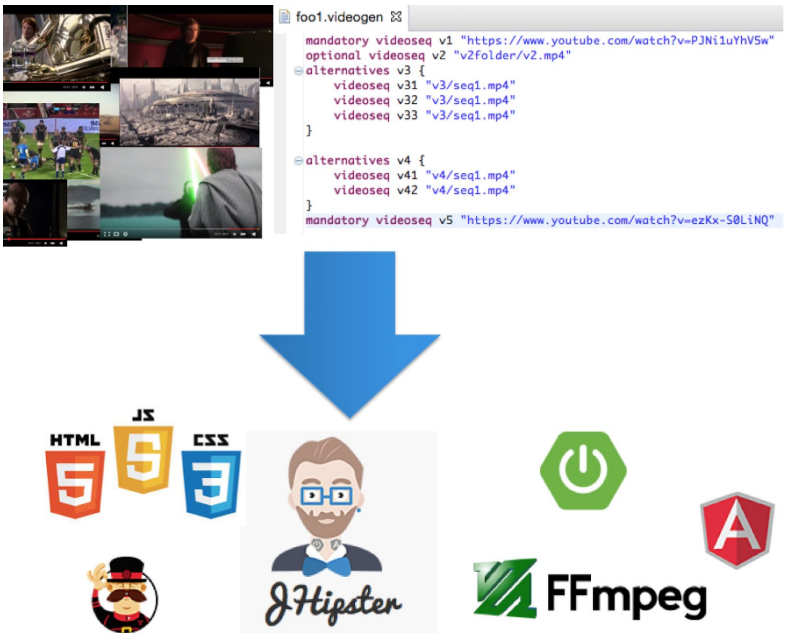
Ingénierie des Modèles

TP (projet) –*
Métamodélisation, DSL, Transformations (2018)

Aide sur ffmpeg: http://slhck.info/ffmpeg-encoding-course/#/42

L'objectif du projet est d'offrir une solution logicielle pour déployer un générateur ou configurateur Web de vidéos à partir d'une spécification textuelle (VideoGen). Les utilisateurs du site Web pourront par exemple visualiser des variantes de vidéos générées aléatoirement ou avec des probabilités. Les créateurs de générateurs de vidéos pourront utiliser le langage VideoGen ainsi que quelques outils pour raisonner sur la durée et taille des vidéos.
Le projet peut être vu comme une généralisation du générateur de "Bref 30 ans" (http://bref30ans.canalplus.fr/). Outre la généralisation de cette idée, nous incorporons d'autres fonctionnalités (configurables) à notre générateur de générateur.



Concrètement, trois tâches principales sont à réaliser:

- Développement d'un générateur de vidéos "feature complete"
  - ~ finir les exercices des différents TP1-4
- Développement d'un site Web pour jouer des variantes de vidéos
  - ~ utiliser JHipster https://www.jhipster.tech/ ou SparkJava http://sparkjava.com/ ou tout autre framework Web[1] pour intégrer une partie de vos fonctionnalités Java|Xtend côté serveur (+ un peu de HTML/JavaScript côté client pour opérationnaliser le rendu)
- Conduite d'une étude empirique sur la robustesse de votre générateur

Ce qui est à rendre:

1. Le code Xtend ou Java des fonctionnalités du générateur de vidéo (incluant des "tests" et un minimum de documentation)
2. Le code du site Web (basé sur JHipster ou n'importe quel framework Web[2] et intégrant le code Xtend/Java précédent) ainsi qu'un "screencast" de démonstration
3. Un rapport succinct mais argumenté pour présenter les résultats et conclusions de l'étude empirique sur la robustesse de votre solution
4. Un générateur de vidéos pour le concours (bonus de 3 points sur la note du projet)

Comment rendre le travail?

Via un **repository git** (e.g., avec Github, Gitlab, ou Bitbucket)

Evaluation

Le travail est à rendre pour le **17 février 2019**. Vous pouvez vous partager le travail: les groupes de 2 sont autorisés.

Barème sur 20:

- Fonctionnalités du générateur de vidéo => 5
- Site Web => 5
- Test et étude empirique => 10
- Bonus? => 3

Il y a également une deadline intermédiaire le **10 février 2019:**

- envoyer un email à mathieu.acher@irisa.fr mentionnant l'adresse de votre **repository git**
- uploader votre fichier .videogen et vos videos sur Google drive, possiblement avec une archive zip ou tar.gz (cf ci-dessous):

[1] L'utilisation de JHipster n'est pas obligatoire. Vous pouvez utiliser n'importe quelle technologie/framework Web pour accueillir votre code Java/Xtend côté serveur

[2] Ou le framework Web de votre choix

https://drive.google.com/drive/folders/1yfePC5mClq8o0NPkuO-HrmBBSYJ9dHJc?usp=sharing

Développement d'un générateur de vidéos "feature complete"

Liste des fonctionnalités à implémenter:

- concaténation des séquences vidéos avec ffmpeg (TP2)
- gestion des probabilités lors du tirage aléatoire des variantes (TP1/TP2)
- export GIF des variantes vidéo (TP3)
- traitement/filtres sur les vidéos (TP4)
- outil d'analyse pour produire les durées des variantes (TP2)
- outil d'analyse pour produire les tailles des variantes (TP3)
- gestion des images: à vous de jouer!

Le code sera écrit en Xtend ou Java (libre à vous). Il faut également fournir des cas de tests pour chaque fonctionnalité précédemment listé. A minima, chaque cas de test peut être vu comme un programme exécutable et une démonstration de comment utiliser votre solution. Vous pouvez inclure quelques assertions intéressantes pour démontrer/vérifier que votre fonctionnalité est correctement implémenté.
Il faut nécessairement fournir des spécifications VideoGen (fichiers .videogen + vidéos) en marge des cas de tests.

Il est également recommandé d'organiser correctement votre code (en regroupant les appels systèmes à ffmpeg dans une classe utilitaire; en regroupant les outils d'analyse dans une classe dédiée; etc.)

Développement d'un site Web pour jouer des variantes de vidéos (putting all together)

Grâce aux fonctionnalités précédentes, on est en mesure de fabriquer des variantes de vidéos. Cependant, on ne veut pas appeler notre générateur avec un Eclipse, Maven, ou en ligne de commande. On veut plutôt fournir une interface Web pour les utilisateurs "finaux"[3].
On utilisera Jhipster (http://jhipster.github.io) ou le framework Web de votre choix pour développer le site Web.
Techniquement, il faut configurer une stack Web particulière puis intégrer une partie les fonctionnalités Java|Xtend côté serveur. Ensuite, on produira du JavaScript côté client pour communiquer avec le serveur, et enfin du HTML/CSS pour présenter une interface graphique et jouer les variantes de vidéos. Il y a deux modes d'utilisation dans le site Web

Mode aléatoire/probabilité

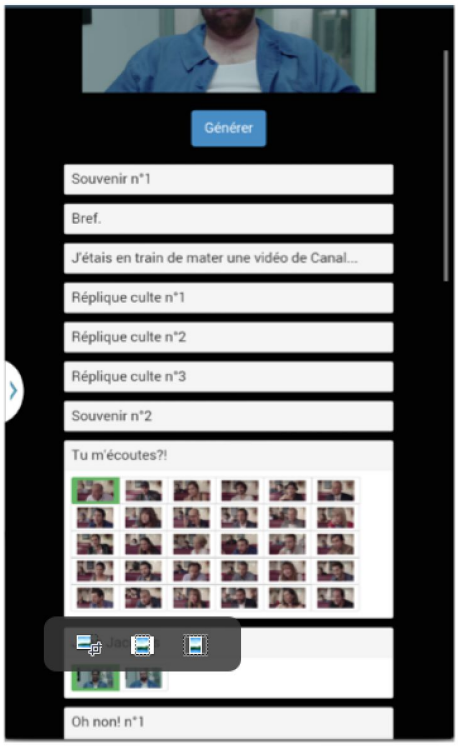
Un utilisateur clique sur un bouton "Generate" pour lancer la génération d'une variante de vidéo. A chaque fois qu'il clique sur "Generate" une nouvelle variante est produite et jouée.
Il peut sauvegarder la vidéo au format GIF

Mode configurateur

L'utilisateur effectue des choix en sélectionnant des vignettes de vidéos. L'ordre d'affichage des vignettes sera déterminé par l'ordre de la spécification VideoGen. Pour les vidéos "alternatives" les vignettes seront affichées dans un même bloc (cf exemple ci-dessous). Pour les "obligatoires", on se contentera d'afficher la vignette (sans interaction possible de l'utilisateur). Pour les "optionnelles", montrer la vignette et faire en sorte que l'utilisateur puisse inclure (ou pas) la vidéo. A la fin de chaque configuration, on joue la variante de vidéo correspondante (l'assemblage de la vidéo se fera côté serveur).

Précision importante: en entrée du mode configurateur, il y a un fichier ".videogen" qui sera traité côté serveur… Le but du "mode configurateur" n'est pas de générer un nouveau ".videogen" mais simplement de permettre à un utilisateur de pro-activement sélectionner les séquences de vidéos à inclure dans la variante souhaitée (autrement dit: l'utilisateur configure/exploite un ".videogen")

[3] A noter que les créateurs de générateur (i.e., les personnes en charge de spécifier des .videogen et de bénéficier des outils d'analyse sur les tailles/durées) n'auront pas de service en ligne. Le site Web s'adresse exclusivement aux utilisateurs finaux (internautes souhaitant visualiser des vidéos).



Test large échelle de votre générateur

Vous avez implémenté une solution complète pour générer aléatoirement une variante de vidéos. La question qu'on souhaite adresser est la suivante: **est-ce que votre procédure marche pour n'importe quel fichier videogen et n'importe quel ensemble de vidéos?**

Pour répondre à cette question, vous allez utiliser votre procédure et lui demander de générer, pour chaque fichier videogen, 10 variantes aléatoire. Idéalement, votre procédure ne plante pas et génère des vidéos de qualité pour n'importe quel .videogen.

Suite de tests: La promotion a programmé différentes solutions pour implémenter le générateur de vidéos, en utilisant une diversité de vidéos et de spécifications videogen. Nous allons profiter ce cette opportunité pour réutiliser ces vidéos et ces .videogen!
Il est demandé que chaque groupe d'étudiants qui suit la matière IDM de rendre disponible ("upload") dans un dossier unique un fichier videogen et les vidéos associés, possiblement avec une archive zip ou tar.gz.
Nous utiliserons Google drive:
https://drive.google.com/drive/folders/1yfePC5mCIq8o0NPkuO-HrmBBSYJ9dHJc?usp=sharing

Précision importante: veillez à ce que les chemins des videos dans .videogen soient ***relatifs au dossier.***
Exemple d'un mauvais videogen:
mandatory videoseq v1 "/Users/macher/v1.mp4"
car le chemin "/Users/macher/v1.mp4" ne sera pas accessible
Un bon videogen:
mandatory videoseq v1 "v1.mp4"
optional videoseq v2 "v2.mp4"
(note: on évitera l'utilisation des sous-dossiers et l'utilisation de "slash" pour être interopérable Linux/MacOS/Windows)

Ainsi chaque groupe pourra tester sa procédure dans différentes situations en téléchargeant les .videogen et les vidéos associés.

Automatisation des tests: Evidemment on ne va pas lancer à la main notre générateur… on va automatiser la tâche. Après avoir téléchargé le dossier sur Google drive, il faut écrire un programme qui parcourt chaque sous-dossier; dans chaque sous-dossier, générer aléatoirement 10 vidéos à partir du videogen qui s'y trouve.
Ecrire un fichier de "log" en sortie qui détaille

- les cas d'erreur: quel(s) videogen sont concernés? quelles sont les combinaisons de séquences vidéos qui ont causé l'erreur? toutes? quelles sont les messages d'erreurs?
- les succès (quel(s) videogen sont concernés?)

Analyse des résultats: Il est attendu un rapport succinct (~2 pages) mais argumenté (basé sur votre fichier de "log") pour présenter les résultats des tests et des conclusions qu'on peut en tirer:

- Dans combien de cas votre procédure fonctionne/échoue?
- Quels sont les cas difficiles?
- Pourquoi votre procédure "plante"?

Le rapport, le code, et la documentation seront mis à disposition dans un sous-dossier de votre dossier unique (sur votre repository Git).

En résumé:

- Upload d'un fichier videogen et des vidéos associés, dans un dossier unique, sur le Google drive (facile)
- Programmation d'une procédure qui considère le .videogen de chaque dossier et exécute la procédure aléatoire de génération de vidéos
- Ecriture d'un rapport

Concours

Divers points bonus seront attribués pour promouvoir les meilleures générateurs de vidéos. Pour gagner, il semble indispensable de développer des fonctionnalités riches du générateur (e.g., pour supporter les filtres sur les vidéos). Et il faut de la créativité!

A noter que le meilleur générateur de vidéos ne produira pas forcément la meilleure vidéo. Un “bon” générateur est certainement un générateur qui produit de multiples variantes de vidéos (à chaque fois de bonne qualité), qui surprend l’auditoire, etc.

Le gagnant du concours aura 3 points de bonus, le deuxième 2 points de bonus, et le troisième 1 point de bonus. Le jury sera composé de quelques enseignants-chercheurs de l’ISTIC.

Boîte à idée (hors évaluation)

Social vidéo: possibilité de “liker” des variantes de vidéo, de commenter, etc.
(Multi-)objective problem: Java solvers (compromis taille/durée)
Un langage plus riche avec répétition, variable, “composite”, mécanismes pour modulariser + contraindre l’assemblage des vidéos
Syntaxe/éditeur graphique pour spécifier/visualiser VideoGen
Composer des génériques de dessins animés; peut-être en coupant le son de certains pour avoir une bande son “uniforme”
Mode “jeu” (configuration multi-step avec accès à des vidéos selon certaines étapes) ou mode shoot’em up (apparition de vignettes qu’il faut shooter pour activer les séquences)