L3 2014

Projet de langage naturel

Jean-Vincent Hay &
Stanislas Mur

[Reconnaissance et synthèse vocale dans les jeux vidéo]

Compte rendu du projet de Langue Naturelle de Jean-Vincent Hay et Stanislas Mur, Jeu vidéo en reconnaissance vocale et synthèse vocale.

POURQUOI LA RECONNAISSANCE ET LA SYNTHESE VOCALE DANS LES JEUX VIDEO ?

Les jeux vidéo sont de plus en plus présents dans notre quotidien, et possèdent chacun leurs langages et leurs codes, tout comme le cinéma. Ils sont aussi un média où l'interaction est un point critique : la manière de jouer est très influencée par la façon dont l'utilisateur, le joueur, pourra interagir avec son environnement.

Ainsi, un jeu de courses sera joué différemment selon que le joueur utilise une manette ou un volant et un pédalier, cette dernière option étant naturellement celle avec laquelle le joueur interagira le mieux, et donc la plus immersive. De même, il est plus naturel pour certains jeux, comme les jeux de stratégie ou d'énigmes, d'utiliser un écran tactile et des gestes des doigts, que d'utiliser une souris et un clavier, qui est plus naturel de jouer avec une manette.

Sur certains jeux encore, la manière de jouer est largement influencée aussi par la communication. Traditionnellement au clavier, à l'aide de textes tapés à la main ou de commandes rapides, la communication peut devenir un outil très efficace, notamment dans des jeux à plusieurs, où la coordination d'une équipe devient capitale. Cette communication peut encore être améliorée, rendue plus immersive et efficace à l'aide de l'usage de la voix, plus naturelle, complète, directe et rapide que tous les types de communication au clavier étudiées jusque-là.

Toutefois les moyens évoqués ne sont que des moyens d'améliorer l'expérience de jeu en la rendant plus réaliste, plus naturelle, immersive... Elles partent toutes du postulat que le joueur est capable de jouer à un jeu vidéo à l'aide des moyens dont il dispose déjà, et des sens dont est doté le joueur, à savoir l'ouïe (entendre les actions et les réactions de l'environnement, les informations sonores, les musiques...), le toucher (contrôle de la manette par exemple, retours de force et retours haptiques...), et plus particulièrement la vue. Ainsi, à l'instar du cinéma, les jeux vidéo sont un média très peu accessible pour les personnes malvoyantes et aveugles.

Ainsi, des projets sont nés pour palier à ce manque d'accessibilité, explorant au passage des manières d'interagir avec le jeu inouïes jusque-là. D'abord dédiés à un public de personnes âgées, ces jeux commencent peu à peu à l'aide des développeurs indépendants à trouver un autre public et d'autres possibilités. On trouve désormais par exemple quelques jeux de rôle textuels tels que Zork, dont les textes sont dictés par l'ordinateur, ou des jeux de stratégie contrôlables par commandes vocales.

http://reflexfaite.wordpress.com/2012/12/13/des-jeux-video-pour-les-aveugles/
http://www.les-rpg.com/dragonium-chance-inouie-aveugles/
http://www.gentside.com/jeu-vid%E9o/a-10-ans-seulement-il-cree-un-jeu-video-pour-aveugles_art39946.html
Une console de jeux à destination principalement des séniors malvoyants : http://www.odimo.fr/
Un jeu de stratégie en temps réel contrôlable à la voix :
http://www.jeuxvideo.com/articles/0001/00010437-tom-clancy-s-endwar-test.htm

LES LACUNES DU TRAITEMENT AUTOMATIQUE DES LANGUES DANS LES JEUX VIDEOS

Il n'existe qu'un seul système capable d'affranchir les joueurs de leur clavier ou de leur écran, lequel est axé pour les seniors malvoyants, et ne propose que des jeux « basiques » comme des jeux de mots croisés.

Il n'existe à l'heure actuelle aucun jeu à visée grand public, et principalement jeune public, capable de fonctionner sans utiliser de clavier ni d'écran.

TECHNOLOGIES

Plusieurs systèmes d'exploitation intègrent des outils divers et variés, plus ou moins accessibles par d'autres outils.

Tout d'abord, il convient de séparer les moteurs des systèmes de reconnaissance et de synthèse vocale.

Un moteur est un programme constitué de grammaires, de modèles acoustiques, et d'un décodeur. L'ensemble de ces trois éléments permet d'analyser des données acoustiques pour les transformer en données exploitables par d'autres ressources.

Un système quant à lui est une application de ce moteur, de la même manière que l'ensemble {habitacle, volant, roues, carrosserie, moteur} forme le système "voiture" par exemple. C'est un programme, ou un une suite de programmes, prêt à être utilisé par l'utilisateur final et à réaliser ce pourquoi il a été fait.

Microsoft par exemple, intègre dans Windows des librairies et des applications de reconnaissance et de synthèse vocale. D'abord utilisés comme outil d'accessibilité pour les personnes malvoyantes en tant que système de synthèse vocale, ces outils ont été intégrés à l'environnement .NET, en tant que moteur, puis utilisés dans Windows pour contrôler l'interface par la voix, en tant que système complet.

http://msdn.microsoft.com/fr-fr/library/gg145021%28v=vs.110%29.aspx

Mono, une implémentation libre de .NET comprend ces fonctionnalités, et permet d'exécuter les programmes conçus à l'aide de .NET dans un environnement Linux.

http://go-mono.com/status/status.aspx?reference=4.0&profile=4.5&assembly=System.Speech

Les moteurs et les systèmes de synthèse et de reconnaissance vocale ne sont pas une idée ancienne, et de nombreuses librairies open-source ont pu voir le jour dans de nombreux langages. Toutefois la reconnaissance et la synthèse vocale sont généralement considérées comme deux technologies suffisamment différentes pour qu'elles soient séparées. Ainsi il est difficile de trouver un projet open-source par exemple qui intègre les deux technologies.
CMUSphinx et VoxForge font partie de ces outils de reconnaissance vocale.

CMU Sphinx est un moteur open-source de reconnaissance vocale. Il est conçu pour la recherche, mais aussi pour développer d'autres applications comme des systèmes.

VoxForge quant à lui est un projet d'agrégation de modèles acoustiques libre. En effet, bien que l'écriture de grammaires spécifiques à certaines langues soit réalisable à taille humaine dans des projets libres, la création de modèles acoustiques, qui seront réutilisés par les moteurs, est une chose plus compliquée. Le principe de VoxForge est donc de recueillir des données de la part d'utilisateurs afin de compléter chaque fois un peu plus ces modèles.

Un exemple de coordination de ces deux moteurs est le logiciel Simon, qui est un système de reconnaissance vocale permettant de contrôler son ordinateur par la voix, disponible sur Windows et Linux.

http://cmusphinx.sourceforge.net/
http://www.voxforge.org/
http://simon-listens.org/index.php?L=1

De même que l'on peut trouver de nombreux moteurs et systèmes de reconnaissance vocale sous licence libre, on peut aussi trouver de nombreux moteurs et systèmes de synthèse vocale sous le même type de licence.

Le projet

SUJET

Nous cherchons à créer un jeu de rôle par texte adapté pour aveugles en python permettant de s'affranchir totalement de l'écran et du clavier, en utilisant la synthèse et la reconnaissance vocale.

Technologies employées

Nous avons décidé de programmer notre jeu en Python, principalement pour sa portabilité et sa simplicité d'utilisation.

Pour ce projet, nous avons choisi d'utiliser deux moteurs de Langue Naturelle :

- Pour la partie Speech-to-Text, nous utilisons l'api Google speech, qui a les avantages de renvoyer la plupart du temps des résultats corrects ou compréhensibles par l'analyseur, et d'être très léger car, et cela est son seul désavantage, il ne fonctionne que par internet. On enregistre un fichier son en format flac avec le logiciel SOX (Linux), qui est envoyé aux serveurs Google speech, qui enfin renvoie le résultat en format json. Ce résultat est ensuite analysé pour extraire le texte final.
- Pour la partie Text-to-Speech, nous avions précédemment prévu d'utiliser la librairie python pyttsx, utilisant elle-même le logiciel libre espeak, disponible sur toutes plateformes, couplé ensuite avec MBROLA pour donner un rendu plus naturel. Finalement nous avons opté pour l'API Google Speech, utilisée par Google Translate, qui présente les mêmes avantages et inconvénients que la solution STT précédemment citée (nécessite une connexion à internet, mais permet aussi d'être beaucoup plus léger, et de meilleure qualité que la plupart des autres solutions libres, pour une implémentation bien plus simple).

http://doc.ubuntu-fr.org/synthese_vocale
https://github.com/gillesdemey/google-speech-v2
https://pypi.python.org/pypi/pyttsx
http://espeak.sourceforge.net/
http://tcts.fpms.ac.be/synthesis/mbrola.html

ANALYSEUR DE COMMANDES EN LANGAGE NATUREL

Grace à l'api Google speech, nous récupérons une entrée texte issue d'une reconnaissance vocale.

Cette entrée texte est analysée pour créer une commande compréhensible par le logiciel. Pour cela, on recherche des mots clefs dans le texte, en prenant en compte une « pseudo-grammaire hors contexte »

Chaque mot de l'entrée sera comparé aux différents mots clefs attendus en mesurant la distance de levenshtein. Le mot clef ayant la plus petite distance sera donc ajouté à la commande si cette distance est inférieure à un seuil égal à la taille du mot analysé.

Les mots clefs attendus dépendent du résultat des analyses de mots précédentes.

En premier lieu, le logiciel attend une commande (Attaquer, aller à, prendre, etc.)

Ensuite, selon la commande reconnue, le logiciel attendra un argument.

Les différents types d'arguments sont contenus dans plusieurs tableaux. Les arguments peuvent donc être des directions (aller à), des objets (prendre), des ennemis (attaquer), ou des personnages non joueurs (parler)

En début de partie, le joueur passera par un tutoriel pour assimiler les différentes commandes.

LISTE DES MOTS CLEFS

Commande	Mot clef	Type d'argument
MOVE	Va	Direction
	Aller	
ATTACK	Attaque	Monstre
	Frappe	
TAKE	Prends	Objet
	Attrape	
TALK	Parle	Personnage non joueur
	Parler	
SEE WEAPON	Arme	N/A
SEE ARMOUR	Armure	
SEE HP	Vie	
	Pv	
QUIT	Arrêter	N/A
	Stop	
PASS	Passe	N/A
REPEAT	Recommence	N/A
	Pareil	
	Encore	
	Idem	

SYNTHETISEUR VOCAL

La synthèse vocale, tout comme la reconnaissance vocale, est effectuée à partir de l'API Google Speech. Cela permet notamment de réduire la charge du programme. Bien sûr il existe d'autre solutions que cette API, mais c'est avant-tout celle-là que nous avons préféré utiliser de par sa simplicité. Toutefois, elle nécessite une connexion constante à internet pour fonctionner.

Lorsque des informations ou du texte devraient être affichés pour être lus par l'utilisateur, ils sont à la place envoyés à l'API. Cette dernière renvoie un fichier MP3 prêt à être lu. Etant donné que nous utilisons SoX pour la lecture et l'enregistrement, nous utilisons ensuite l'encodeur/décodeur LAME, qui nous permet de passer d'un fichier MP3 à un fichier WAV, lisible par SoX. Le texte est enfin lu.

Cette API pose certains problèmes. Par exemple le nombre de caractères à envoyer est limité. De plus elle n'est pas aussi paramétrable que eSpeak, ce qui pose des problèmes au niveau de la clarté de la locution, de la ponctuation et des accents. Avec eSpeak, il serait possible de moduler la vitesse et la hauteur du son. Toutefois, eSpeak est sensiblement moins performant que Google concernant la qualité de synthèse du français, du fait du manque de travail effectué, par l'un comme par l'autre, sur cette langue.

De plus, eSpeak utilise plus de ressources machine que d'envoyer du texte, télécharger un mp3 et le décoder, ce qui présente un avantage supplémentaire en faveur de Google, notamment si l'on désire porter ce type d'application sur des appareils portables, où l'utilisation des ressources est intimement liée à l'autonomie de l'appareil. Enfin, l'API Google ne nécessite pas de fonctionner sur une plateforme spécifique, contrairement à eSpeak, qui peut nécessiter selon les systèmes des installations et des configurations différentes. C'est aussi un problème qui a été rencontré avec SoX, l'outil utilisé pour l'enregistrement et la lecture des fichiers audio, dont l'installation sur windows nécessite que l'utilisateur modifie des variables d'environnement. Toutefois, le code est conçu de manière à-ce que l'utilisateur puisse paramétrer les outils utilisés pour la lecture et l'enregistrement.