Lausanne janvier 2011

I] Présentation générale

Nous sommes partis d'un constat : quasiment tous les élèves possèdent chez eux un ordinateur, très peu l'utilisent pour autre chose que les jeux, les messageries instantanées et les réseaux sociaux. L'idée est donc de leur faire découvrir une autre utilisation possible, la programmation et plus précisément la programmation orientée objet (POO).

Pourquoi apprendre la programmation ?

L'apprentissage de la programmation est un domaine pédagogique particulièrement intéressant permettant de donner aux élèves l'expérience de systèmes complexes en développant un mode de pensée structuré, méthodique et rigoureux qu'ils pourront ensuite mettre en œuvre dans de nombreuses autres disciplines.

La pratique de la programmation permet de se trouver confronté à des problèmes plus ou moins complexes nécessitant à la fois une démarche logique basée sur l'analyse, une approche par hypothèses, essais successifs et élimination, une capacité imaginative, un esprit d'initiative propice à l'autoapprentissage et enfin le développement de qualités plus génériques telles que la patience, l'ouverture d'esprit, l'adaptabilité, la précision et la rigueur.

Enfin, la programmation s'appuie sur des langages qui ont leur vocabulaire et leur syntaxe, elle renforce donc la formation de l'esprit aux pratiques linguistiques.

Les activités proposées n'ont aucunement la prétention de faire des élèves des programmeurs confirmés, juste de leur donner l'envie de découvrir cette nouvelle discipline, en essayant de la démystifier, autant que faire se peut !

II] Alice en classe de seconde

Pour qui ?

Les élèves de seconde (15/16 ans) ayant choisi de suivre l'enseignement d'exploration PSN (pratique scientifique et numérique, 3h/semaine) suivent une initiation à la POO. Il faut noter que d'autres activités leurs sont proposées durant ces 3 h : construction d'un altimètre "électronique" avec test sur le terrain et "physique et physiciens du XXe siècle" (mécanique quantique, relativité, physique des particules) avec visite préparée du LAPP d'Annecy le Vieux. La programmation est aujourd'hui omniprésente dans beaucoup de domaines de la physique (électronique : programmation d'interfaces complexes, physique théorique (nous avons été surpris d'apprendre de la bouche même de certains chercheurs du LAPP que la programmation de classe en C++ était leur lot quotidien). Tous les thèmes abordés dans cet enseignement d'exploration permettent donc aux élèves d'appréhender un peu mieux les enjeux de la physique du XXIe siècle.

Alice ?

L'apprentissage de la programmation est potentiellement porteur de frustrations !
Les tâtonnements pour obtenir un résultat satisfaisant peuvent être longs et stériles, la patience trop malmenée, les problèmes jugés trop complexes, leurs résolutions trop obscures et la syntaxe trop difficile à apprendre. D'où l'utilisation du logiciel Alice.

Alice est un logiciel « open source » qui a été réalisé à l'Université de Carnegie Mellon (Pittsburgh – Pennsylvanie – USA), dans le cadre d'une démarche pédagogique visant à motiver des élèves de collèges ou de lycées à s'initier à la programmation.
Le principe d'Alice est de permettre la construction d'une scène 3D incluant des décors, des personnages (humains ou humanoïdes), des animaux, des objets, du texte, du son ou d'autres éléments issus de bibliothèques fournies avec le logiciel.

Alice permet d'aborder beaucoup de notions utilisées en POO : les méthodes, les fonctions, les paramètres et même la notion d'héritage.

L'enseignement de la POO avec Alice permet la mise en place d'un "apprentissage par essais successifs" (j'ai une idée, j'essaye, cela ne fonctionne pas, j'essaie de comprendre pourquoi, puis de trouver mon erreur et de la corriger).

Il ne faut pas se fier au côté ludique d'Alice, le résultat est une chose, la façon d'y parvenir en est une autre.

Enseignement d'Alice au lycée G Fichet

L'année dernière (2009-2010), un cours "classique" était dispensé aux élèves: présentation des nouvelles notions à l'aide d'un vidéoprojecteur, suivi d'exercices d'applications.
Cette méthode a très vite montré ses limites. L'intervention de l'enseignant, trop longue, entraînait inévitablement un "décrochage" de l'attention des élèves.
Pour notre 2e année, nous avons donc décidé de modifier notre méthode d'enseignement. Chaque élève progresse à son rythme. Un élève (ou plutôt un binôme) commence par se confronter à un problème (les problèmes proposés s'inspirent grandement des exercices proposés dans le document "Initiation à la programmation orientée objet avec Alice" (voir l'annexe : "exemple de situations-problèmes")). Pour résoudre ce problème, les élèves recherchent dans ce même document les nouvelles notions qui leur permettront d'arriver à leur fin, l'enseignant étant là pour plus ou moins les orienter dans leurs recherches et réexpliquer de vive voix (et individuellement) les notions les plus complexes.
L'évaluation est 100% pratique, les élèves ont à concevoir un programme qui devra respecter les consignes données par l'enseignant (un exemple de contrôle est disponible dans l'annexe). Cette évaluation est bien évidemment individuelle.

Premier bilan

Dès l'année dernière, nous avions noté que certains élèves peu motivés dans les autres disciplines montraient un intérêt certain pour l'apprentissage de la POO (un élève "décrocheur" a décidé de se

réorienter dans une filière professionnelle "bac pro informatique et réseau" et selon ses dires, son choix n'était pas totalement étranger à sa découverte de la POO avec Alice). Alice n'est pas la solution miracle qui permettra à des élèves en difficulté de retrouver le chemin du succès, mais, c'est une des approches possibles pour l'apprentissage de la POO, apprentissage de la POO qui, d'après nous, à sa place dans les "enseignements d'exploration" proposés aux élèves de seconde.
De plus, Alice est aussi une expérience pédagogique nouvelle pour l'enseignant.
"La résolution d'une situation problème" est de plus en plus pratiquée, Alice est un bon moyen de mettre en oeuvre cette méthode pédagogique.
Autre élément nouveau, il arrive parfois que les élèves, à force de pratique, atteignent un niveau de compétence équivalent à celui des enseignants, ce qui normalement n'arrive jamais dans les disciplines plus "classiques". Il faut apprendre à gérer ce genre de situation.

III] Atelier scientifique en classe de première

Pourquoi ? Pour qui ? Comment ?

À la fin de l'année scolaire 2009-2010, certains élèves nous ont demandé de poursuivre leur découverte de la POO pendant leur année de première. Après avoir discuté avec notre hiérarchie de la faisabilité du projet, la forme "atelier" a été retenue. Les élèves volontaires se réunissent avec un enseignant toutes les semaines pendant une heure. Cet atelier n'est pas un cours au sens classique du terme, mais plutôt un espace d'échanges propice aux séances de "questions réponses". En effet, les sujets abordés sont beaucoup trop vastes pour être uniquement travaillés une heure par semaine. Les élèves travaillent donc en autonomie grâce aux documents mis à leurs dispositions ("D'Alice à Java", "De Java à Andoid" ou d'un futur document sur GWT) et se retrouvent une fois par semaine pour poser des questions à l'enseignant sur les points mal compris (pour être tout à fait honnête, certains élèves travaillent uniquement pendant l'heure hebdomadaire, il n'y a rien de choquant à cela, au vu de leur emploi du temps, mais cela se ressent sur leur progression).

"D'Alice à Java"

Ce document essaie d'établir une passerelle entre les notions vues en seconde avec Alice et la "vraie" programmation en Java. Il rappelle en permanence les notions de base de la POO vue avec Alice avant de passer à Java.
Voici un exemple tiré du document "D'Alice à Java" : l'instruction "while"

Voir l'Annexe, exemple 1

Ce document essaye au maximum "de laisser la théorie pour plus tard" (il faut toujours avoir à l'idée que le but ici encore est de faire découvrir aux élèves une nouvelle "discipline", pas de faire des élèves des programmeurs professionnels), en se basant au maximum sur des exemples.

Voir l'Annexe, exemple 2

Le document ne fait pas mention de certaines notions pourtant importantes en java. Dès le départ, il était clair que le but n'était pas de faire une "bible du java", mais de donner les notions de base permettant d'aborder la programmation sous Android (et plus tard le développement web avec GWT).

Voir le plan du document en annexe

"De Java à Android"

Android est un système d'exploitation développé par l'open handset alliance (le principal contributeur est Google), reservé à l'origine aux smartphones, mais que l'on trouve de plus en plus sur les tablettes (Archos, Samsung, Toshiba,...). Le document "De Java à Android" initie les élèves au développement d'applications sous Android. Les applications sous Android sont écrites en Java (avec une machine virtuelle un peu différente de la classique JVM).

Pourquoi Android ?

Les smartphones sous Android sont souvent des appareils tout-en-un (appareil photo, GPS, connexion à internet.....). La programmation de ces composants est "relativement" simple et permet donc d'écrire des applications élaborées. De plus, les outils de développement (IDE Eclipse + plugin) sont gratuits ce qui n'est pas négligeable dans le cadre d'une utilisation scolaire.

Voir le plan du document en annexe

Après un premier chapitre consacré à l'installation, les trois chapitres suivants sont consacrés à la programmation de l'interface graphique (utilisation de XML et de Java).

Voici un extrait sur la mise en place des boutons

Voir l'Annexe, exemple 3

Le chapitre 5 aborde l'interaction utilisateur-machine

Voir l'Annexe, exemple 4

Enfin, le chapitre 6 s'intéresse à l'appel des activités ou des services.

À l'origine il était prévu une deuxième partie : "Pour aller plus loin" (la première partie, les chapitres 1 à 6, étant intitulée "les bases"). Aujourd'hui, je ne sais pas si cette 2e partie verra le jour !
Des notions plus complexes telles que les bases de données SQLite, les fournisseurs de contenus, le réseau ou bien encore les parseurs XML devaient alors être abordées. En commençant à rédiger cette deuxième partie, je me suis vite rendu compte qu'il était impossible (en tout cas pour moi !) de "rester simple" dans mes explications, ce qui allait à l'encontre de mon objectif de départ.
Il existe d'excellents ouvrages sur le développement d'applications sous Android, trop difficile (toujours d'après moi !) pour des débutants, ils pourront en revanche être très utiles pour les élèves ayant déjà fait l'effort d'acquérir les bases en étudiant "de Java à Android". Si à la fin de la lecture de ce document, l'élève est pris d'une "irrésistible" envie de se procurer un de ces ouvrages, pour "creuser" la question, j'aurais atteint mon objectif !

"De Java à GWT" ?

Depuis quelques semaines, certains élèves ont débuté l'étude de Google Web Toolkit.
GWT est un framework fourni par Google permettant d'écrire des applications de type RIA.

"Une *rich Internet application* (RIA), ou application Internet riche, est une application Web qui offre des caractéristiques similaires aux logiciels traditionnels installés sur un ordinateur. La dimension interactive et la vitesse d'exécution sont particulièrement soignées dans ces applications Web."

Source wikipedia

Pour écrire ce type d'application le couple Javascript + XML, aussi appelé AJAX (pour être tout à fait rigoureux, l'***Asynchronous Javascript and XML*** est la combinaison de Javascript, XML, CSS, DOM et XMLHttpRequest), était quasiment incontournable.

Problème, Javascript est un langage compliqué à maîtriser (en tout cas pour moi !), Google a donc mis à la disposition des développeurs web GWT, qui permet de "faire" de l'AJAX en écrivant du Java à la place du Javascript

Comme l'idée de présenter aux élèves GWT ne m'est venue que courant décembre, je n'ai pas encore écrit de document, les élèves utilisent donc pour l'instant des "tutorials" trouvés sur internet et le site officiel de Google, qui est très bien fait, mais en Anglais....

Pourquoi GWT ?

Pour commencer, parce que GWT utilise Java, c'est donc (comme pour le développement sous Android), un bon moyen de réinvestir le travail fait précédemment. Mais cela ne s'arrête pas là. En effet, c'est aussi l'occasion d'aborder des notions très importantes souvent très peu connues des élèves :
Qu'est-ce qu'un réseau ? Qu'est-ce que l'Internet ? Que signifie HTTP ? Qu'est-ce qu'un client ? Qu'est-ce qu'un serveur ?

Au jour d'aujourd'hui les élèves ont le choix de travailler sur Android ou sur GWT. En fin d'année leurs réalisations (s'il y'en a !) devraient être présentées au public dans le cadre d' "Expo Science Annecy 2011".

Certains pourront sans aucun doute me reprocher de me disperser, de "courir trop de lièvres à la fois" et de proposer un contenu trop riche aux élèves. Cette critique est sans aucun doute justifiée, mais je rappellerai (pour ma "défense") que ce projet est expérimental (autant pour les enseignants que pour les élèves) et qu'il évoluera dans les mois qui viennent.

IV] Liens

Alice :
www.alice.org
www.animanum.com/alice
http://www.epi.asso.fr/revue/articles/a1006d.htm
Java :
http://www.epi.asso.fr/revue/articles/a1009b.htm
contact :
projetalicejava@gmail.com

IV] Questions de l'auditoire

V] Atelier pratique : découverte d'Alice

1) Exemple tiré du document "Initiation à la POO avec Alice"

p 66 : **Méthodes, paramètres, Events et if/else**

2) Exercices

Exercice 1

Créer une scène avec un bonhomme de neige et 4 sphères (shapes/sphere) de 4 couleurs différentes (rouge, vert, bleu et jaune). Écrire un programme permettant au bonhomme de neige de prendre la couleur de la sphère choisie (on choisira la sphère en cliquant dessus). Un clic sur le bonhomme de neige doit le faire redevenir blanc. À chaque changement de couleur, le bonhomme de neige doit "dire" sa couleur.

Exercice 2

Nous allons créer un programme permettant de travailler son vocabulaire en Anglais. Créer un monde avec un chat (Class Sitting_cat). Placer devant le chat 3 mots en 3D (les mots en 3D sont des objects comme les autres) : cat, dog et bird. Au cas où l'élève clique sur la bonne réponse (le bon mot en 3D), le chat devra dire :"Bravo tu es le meilleur". Si l'élève clique sur n'importe quoi d'autre, le chat devra hocher la tête et dire "Non, essaie encore une fois". Pour arriver à vos fins, vous devrez créer une fonction "cestbon" qui renverra true en cas de bonne réponse et false en cas de mauvaise.

Exercice 3

Vous allez créer un "jeu" pédagogique destiné aux enfants de l'école primaire :
L'enfant verra apparaître sur l'écran un animal (un chameau, une vache ou un pingouin), au-dessus de l'animal, l'enfant aura 3 choix possibles (texte en 3D) : "Camel", "cow" ou "penguin"

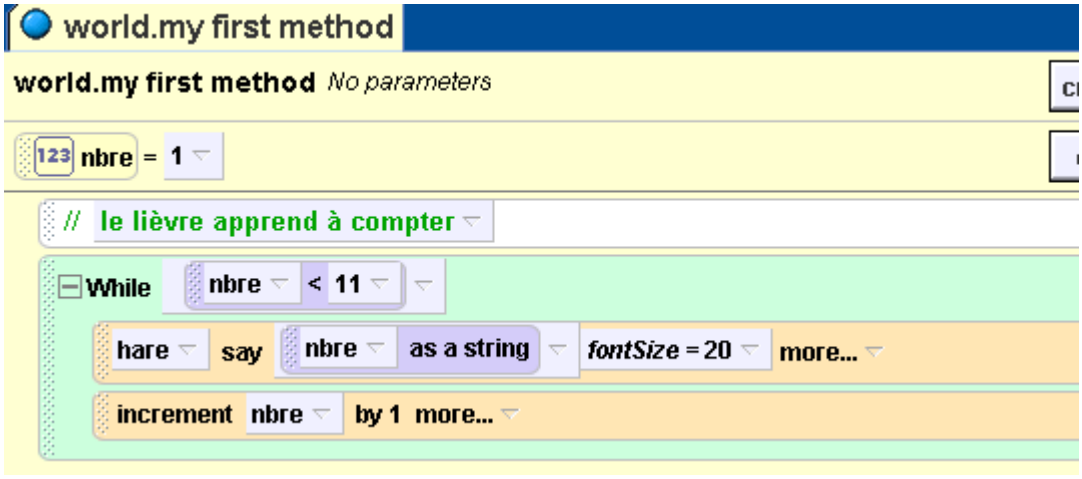
Si l'enfant clique sur le bon mot (celui qui correspond à l'animal affiché), on verra apparaître "Bravo" et un nouvel animal s'affichera à la place du précédent. Dans le cas contraire, on verra apparaître "Faux essaye encore une fois".
Attention le choix de l'animal à afficher (vache, chameau ou pingouin) devra être aléatoire (géré par le hasard)

Annexes

exemple 1

While

Dans Alice nous avons déjà eu l'occasion d'utiliser l'instruction *while* à plusieurs reprises.
Voici un exemple pour vous rafraîchir la mémoire (le lièvre compte jusqu'à 10)



while signifie « tant que ». Dans l'exemple « Alice » (ci-dessus), nous avons créé une variable locale *nbre*. Tant que *nbre* reste inférieur à 11, le lièvre affiche la valeur de *nbre*, à la fin du bloc *while*, la variable *nbre* est augmentée d'une unité.

Voici l'équivalent Java de ce programme :

Ex 4.3

```
public class Principal {
	public static void main(String[] args) {
		//Création de notre variable locale
		int nbre = 1;
		while (nbre<11){
			System.out.println (nbre);
			nbre=nbre+1;

		}

	}

}
```

Rien de très difficile dans cet exemple, vous pouvez constater que l'utilisation du *while* en Java est assez simple.

exemple 2

Surdéfinition (ou surcharge) d'une méthode

En java (et dans tous les langages orientés objets) deux méthodes peuvent porter le même nom à condition qu'elles n'aient pas les mêmes paramètres (ce n'est pas tout à fait vrai, elles peuvent avoir les mêmes paramètres dans certaines situations, on parle alors de redéfinition, nous verrons cela dans le chapitre sur l'héritage). On parle de surdéfinition ou de surcharge d'une méthode.
Prenons tout de suite un exemple pour bien vous faire comprendre l'intérêt de la surcharge d'une méthode.
Imaginons qu'il existe 2 types de personnages: les soldats qui combattent (les forces du mal !) et les civils. Les soldats ont des points de force, mais pas les civils. Nous allons donc écrire 2 méthodes *initialisation;* une pour les soldats avec 3 paramètres (*vie* (de type *int*), *force* (de type *int*), *nom* (de type *String*)) et une pour les civils avec uniquement 2 paramètres (*vie* (de type *int*) et *nom* (de type *String*)).

Ex 2.9

Classe Personnage :

```
public class Personnage {
	private int point_de_vie;
	private int point_de_force;
	private String nom_perso;
	public void initialisation (int vie, int force, String nom){
		point_de_vie = vie;
		point_de_force = force;
		nom_perso = nom;
		System.out.println ("Vous venez de créer un nouveau personnage (un
soldat)");
		System.out.println ("Il se nomme " + nom_perso);
		System.out.println ("point de vie : " + point_de_vie);
		System.out.println ("Point de force :" + point_de_force);
	}
	public void initialisation (int vie, String nom){
		point_de_vie = vie;
		nom_perso = nom;
		System.out.println ("Vous venez de créer un nouveau personnage (un
civil)");
		System.out.println ("Il se nomme " + nom_perso);
		System.out.println ("point de vie : " + point_de_vie);
	}

}
```

Classe Principal :

```
public class Principal {
	public static void main(String[] args) {
		Personnage bilbo = new Personnage ();
		bilbo.initialisation(200,100,"Bilbo");
		Personnage gollum = new Personnage ();
		gollum.initialisation(150, "Gollum");
		}
}
```

Résultat :

Vous venez de créer un nouveau personnage (un soldat)
Il se nomme Bilbo
point de vie : 200
Point de force :100
Vous venez de créer un nouveau personnage (un civil)
Il se nomme Gollum
point de vie : 150

Plan du document : d'Alice à Java

Chapitre 1 : Introduction

Chapitre 2 : Les classes, les champs et les méthodes

Chapitre 3 : Commentaires et première fenêtre

Chapitre 4 : Les structures de contrôle

Chapitre 5 : Notion d'héritage et de polymorphisme

Chapitre 6 : Les bases de la programmation graphique
1ère partie : fenêtres et boutons

Chapitre 7 : Les bases de la programmation graphique
2e partie : Gestion des évènements, les listeners

Plan du document de Java à Android

chapitre I	Introduction et installation
chapitre II	Première application
chapitre III	Les layouts
Chapitre IV	Les widgets
chapitre V	Les listeners
chapitre VI	Les intents

exemple 3

Les widgets sont eux aussi des objets au sens POO du terme. Chaque type de widgets est une classe qui dérive de la classe "View" (d'ailleurs, la classe "Viewgroup" hérite elle aussi de la classe "View"). Nous placerons systématiquement nos widgets dans des layouts. Commençons donc à étudier quelques widgets.

Les Boutons

Voici un fichier main.xml qui permettra d'afficher un bouton

ex4.1 : main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Un Bouton"
    />
</LinearLayout>
```

Nous avons une balise "Button" : les attributs "layout_width" et "layout_height" ne devraient pas vous poser de problème (rien de nouveau). L'attribut "text", vous permet d'afficher du texte dans votre bouton.

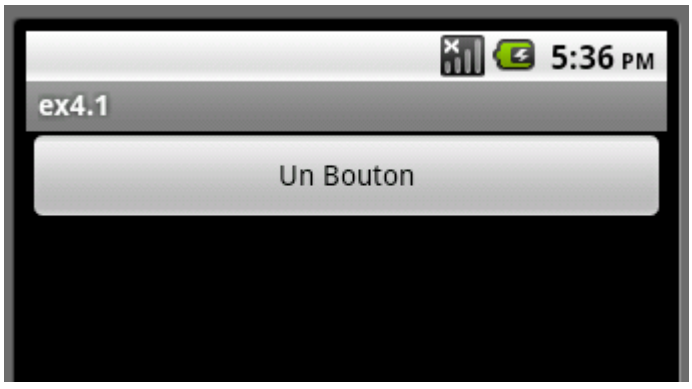
ex4.1 : Principale.java

```
package org.education.lgf.atelier;

import android.app.Activity;
import android.os.Bundle;

public class Principale extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

ex4.1 : résultat



exemple 4

Comment gérer la présence de 2 boutons ?

Il suffit de tester l'objet "view" passé à la méthode "onClick", grâce, par exemple, à un "if/else"
Prenons un exemple:

ex5.4 : main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:gravity="center"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<Button
      android:id="@+id/boutonA"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Bouton A"
    />
<Button
      android:id="@+id/boutonB"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Bouton B"
    />
</LinearLayout>
```

Rien à signaler pour le fichier "main.xml"

ex5.4 : Principale.java

```
package org.education.lgf.atelier;

import android.app.Activity;
import android.os.Bundle;
import android.widget.Button;
import android.widget.Toast;
import android.view.View;

public class Principale extends Activity implements View.OnClickListener {
    /** Called when the activity is first created. */
      Button monboutonA;
      Button monboutonB;
      @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        monboutonA =(Button)findViewById(R.id.boutonA);
        monboutonA.setOnClickListener(this);
        monboutonB =(Button)findViewById(R.id.boutonB);
        monboutonB.setOnClickListener(this);
    }
    public void onClick(View view){
      if (view==monboutonA){
            Toast.makeText(this,"Bouton A", Toast.LENGTH_SHORT).show();}
      else {Toast.makeText(this,"Bouton B", Toast.LENGTH_SHORT).show();}
    }
}
```

L'exemple est, je pense, suffisamment parlant, il suffit de comparer l'objet qui vient de subir un clic (l'objet "view") avec "monboutonA". Si "view = = monboutonA" renvoie "true" alors le "toast" "Bouton A" sera affiché, sinon c'est le "toast" "Bouton B" qui sera affiché.

Exemples de situations-problèmes :

Programme n°1 :
La scène comprend un bonhomme de neige et 4 sphères (shapes/sphere) de 4 couleurs différentes (rouge, vert, bleu et jaune).
Vous allez créer sous Alice un programme permettant au bonhomme de neige de prendre la couleur de la sphère choisie (on choisira la sphère en cliquant dessus). Un clic sur le bonhomme de neige doit le faire redevenir blanc. À chaque changement de couleur, le bonhomme de neige doit "dire" sa couleur.

Programme n° 2
Vous allez créer un programme permettant de travailler son vocabulaire en Anglais.
Voici le déroulement souhaité :
Le monde contient un chat (Class Sitting_cat). Devant le chat sont écrits 3 mots en 3D (les mots en 3D sont des objects comme les autres) : cat, dog et bird.
Au cas où l'élève clique sur la bonne réponse (le bon mot en 3D), le chat devra dire :"Bravo tu es le meilleur". Si l'élève clique sur n'importe quoi d'autre, le chat devra hocher la tête et dire "Non, essaie encore une fois".

1°) Écrivez sur papier l'organigramme correspondant à ce programme.
2°) Après accord du professeur, programmez cet organigramme sous Alice.

(Aide donnée si nécessaire : Pour arriver à vos fins, vous créerez une fonction "cestbon" qui renverra true en cas de bonne réponse et false en cas de mauvaise.)

Programme n°3
Vous allez créer un "jeu" pédagogique destiné aux enfants de l'école primaire :
L'enfant verra apparaître sur l'écran un animal (un chameau, une vache ou un pingouin), au-dessus de l'animal, l'enfant aura 3 choix possibles (texte en 3D) : "Camel", "cow" ou "penguin"
Si l'enfant clique sur le bon mot (celui qui correspond à l'animal affiché), on verra apparaître "Bravo" et un nouvel animal s'affichera à la place du précédent. Dans le cas contraire, on verra apparaître "Faux essaye encore une fois".

Attention : le choix de l'animal à afficher (vache, chameau ou pingouin) devra être aléatoire (géré par
le hasard)

1°) Écrivez sur papier l'organigramme correspondant à ce jeu.
2°) Après validation de cet organigramme par le professeur, programmez-le sous Alice.

Exemple d'évaluation

Contrôle PSN Alice

1) Créer une scène avec une tortue (Class Tortoise), un lapin (Class Bunny) et un pingouin (Class Penguin).
2) Le lapin, la tortue et le pingouin devront regarder la caméra tout en étant parfaitement alignés.
3) Écrire **une seule** méthode (pour les 3 personnages) permettant au lapin, à la tortue et au pingouin de dire :
   « Bonjour, je suis », une pause d'une seconde, « un lapin » pour le lapin, « un pingouin » pour le pingouin et « une tortue » pour la tortue
4) Arrangez vous pour que (dans cet ordre) :
   - le lapin dise : « Bonjour, je suis », une pause d'une seconde, « un lapin »
   - le pingouin dise : « Bonjour, je suis », une pause d'une seconde, « un pingouin »
   - la tortue dise : « Bonjour, je suis », une pause d'une seconde, « une tortue »
5) Écrire une méthode permettant au lapin de sauter en l'air d'exactement sa taille (si le lapin mesure 70 cm, il devra sauter en l'air de 70 cm).
6) Écrire une méthode permettant à la tortue d'avancer vers la caméra de la distance séparant le lapin et le pingouin.