BOLLET Jacqueline
GRENIER Thierry

Projet de Programmation Orientée Objet

Rapport Final

Projet encadré par Marc Daniel

Année 2000-2001

Introduction

Notre projet porte sur la mise en place d'un Tamagoshi (écrit dans un langage orienté objet : le C++), animal virtuel réagissant aux interactions de son environnement et de son maître qui suivra une évolution dans le temps.

Nous présenterons, dans un premier temps, les changements du cahier des charges, puis l'automate d'états finis, ensuite un Diagramme de Contexte Final (qui n'est pas définitif malgré son nom) et nous terminerons par un peu mot sur l'affichage en présentant l'aspect général qu'aura le jeu.

I- Présentation du programme – Mode d'emploi

1) Présentation générale

Un tamagoshi est, à la base, un animal virtuel qui a les mêmes besoins qu'un animal réel. Nous avons trouvé plus plaisant de faire un tamagoshi capable de « comprendre » ce que nous lui disons et de « répondre » au lieu de faire une « machine à manger et dormir » qui obéirait par des clics sur des boutons. Il est bien évident qu'en trente-deux heures de projet, il n'est pas pensable de lui apprendre un langage complet.

Notre animal virtuel comporte donc un dictionnaire de mots auxquels il réagira différent. Il comporte aussi un dictionnaire de réponses. Pour simplifier, nous n'avons implémenté qu'une seule réponse à un ordre donné.

Au départ nous pensions intégrer notre projet à un programme déjà existant qui se connecte sur l'Internet Relay Chat, nous n'avions donc pas besoin d'interface graphique. Depuis, le tamagoshi est devenu un programme autonome et indépendant de l'IRC, c'est pourquoi une fenêtre s'ouvre au lancement pour signaler à l'utilisateur l'état du jeu.

En tant qu'animal virtuel, le tamagoshi a besoin de dormir et de manger mais aussi de sortir, et surtout que l'on s'occupe de lui d'une manière générale. Bien entendu, si nous nous étions limités à ces fonctions, le programme aurait été très vite ennuyant. C'est pourquoi il possède aussi d'autres fonctions comme les jeux, la lecture ou la cueillette.

Enfin, il nous a semblé qu'un animal virtuel qui vivrait à notre rythme n'aurait quasiment jamais de besoins. Sans compter les moments où il dort ! C'est pourquoi nous avons choisit de le faire évoluer dans un temps virtuel. Sachant que chaque utilisateur aura sa propre façon d'utiliser ce programme, nous lui laissons le choix de la durée d'une journée virtuelle. Ceci étant dit, pour que le tamagoshi ait des réactions normales, nous déconseillons fortement de choisir une durée inférieure à cinq minutes. De plus, s'il n'y a pas de borne supérieure, nous conseillons de définir une durée de trente minutes, voire une heure au maximum. Il faut bien se rendre compte que plus la journée virtuelle sera longue, plus l'utilisateur sera tenté de délaisser son animal.

Un autre point de vue aurait été de définir une journée réelle et de signaler à l'utilisateur un besoin par un bip ou une fenêtre qui s'afficherait au moment adéquat…

2) Mode d'emploi

En préambule, nous pouvons signaler que le programme est fait pour tourner sous un « UNIX » ayant le compilateur g++ (v3 conseillée) avec les librairies standards (ainsi que la librairie X11 pour l'interface graphique).

Malgré cela, à la librairie X11 près, le programme devrait être très facilement portable sous un système d'exploitation tel que Windows.

D'une manière classique les sources du programme se trouvent dans un fichier compressé : Bollet_Grenier.tar.gz. Pour le décompresser, il suffit de taper dans une fenêtre shell « tar –xvvzf Bollet_Grenier.tar.gz ». Ceci créera un répertoire « tmg/ » contenant les sources.
La commande « make » permet de créer l'exécutable du nom de « tamagoshi » ; la commande « ./tamagoshi » permet donc de lancer l'exécution du programme.

Il est à noter que la commande « make clean » permet d'effacer les fichiers créés lors de la compilation, devenus alors inutiles. De plus, la commande « make propre » permet d'effacer aussi l'exécutable et les fichiers qu'il aura pu créer et de revenir à la configuration avant compilation.

Le programme lance une nouvelle fenêtre à l'écran et se sert du shell de départ: la nouvelle fenêtre n'est que de l'interface sur laquelle est dessiné le tamagoshi. Elle ne sert qu'à voir l'état actuel de son animal. Le shell de départ est la fenêtre destinée au dialogue entre l'utilisateur et le tamagoshi.

Avant toute chose, le programme demande à l'utilisateur la saison de départ. Les saisons seront visibles quand le tamagoshi sera dans le jardin. Ensuite, l'utilisateur pourra choisir le nom de son animal, et son propre nom. On peut noter que les noms composés devront être entrés avec un séparateur autre que l'espace, sinon seule la première partie du nom sera prise en compte.

Enfin, la durée d'une journée virtuelle lui sera demandée d'être entrée en nombre de minutes (cf. la présentation générale pour la durée conseillée).

Le jeu commence alors. Commençons par expliquer la fenêtre de l'interface graphique :

Deux environnements sont pris en compte : la maison et le jardin.

- *La Maison* : c'est l'endroit où le tamagoshi évolue par défaut. C'est aussi là où il va manger. On y trouve par conséquent son pouf (qui lui sert aussi de chaise et de lit) et une table. Une ampoule indique aussi s'il fait jour ou nuit. C'est ici qu'il peut manger (biberon, purée, pâté, boulette, viande, bonbon, gâteau), dormir, se faire guérir (aspirine, Rennie, piqûre ou antibiotique), lire et jouer à cache-cache (pour plus de détails sur l'enchaînement des activités voir l'Automate d'Etats Finis).

- *Le Jardin :* On y trouve une fleur (en été ou printemps) et un pommier qui indique la saison (pas de pomme ni de fleur : hiver, fleurs : printemps, pommes roses : été, pommes rouges : automne). On représente le temps qu'il fait simplement : un soleil, un soleil caché par un nuage, un nuage… C'est ici qu'il peut jouer à la balle, manger une pomme et cueillir une fleur.

Comme expliqué précédemment, l'essentiel de l'intérêt du jeu n'est pas dans l'interface graphique mais dans la partie texte :

Les phrases du tamagoshi sont écrites en vert, ses besoins en bleu ; celles de l'utilisateur sont en noir. Le tamagoshi peut alors deux sortes d'actions : les actions automatiques et celles commandées.

Actions automatiques

Dès le lancement effectif du tamagoshi, celui-ci commencera à se mettre dans une «attente active ». Ceci veut dire que même si vous ne vous occupez pas de lui, il ne manquera de vous rappeler à l'ordre.

- « faim » : donnez-lui à manger ! (voir l'annexe Nourriture)
- « sommeil » : couchez-le !
- « ennui » : Il s'ennuie ; jouez avec !
- « malade » : soignez-le !
- « rentrer à la maison » : Il est dans le jardin mais à une envie… rentrez le à la maison !

Ces envies sont calculées en fonction du temps. Si l'envie n'est pas satisfaite, le tamagoshi continue à se plaindre, de plus en plus en souvent.
Note : Pour le sommeil, si vous attendez trop avant de le coucher, l'heure sera passée et tamagoshi n'aura plus sommeil… En revanche, il aura perdu beaucoup de points de santé et de joie.

Régulièrement (et pendant son sommeil ou à son réveil), vous aurez d'autres informations :

- ➔ Numéro du jour : permet de voir où en est le tamagoshi dans son évolution. On peut noter qu'il devient adolescent au cinquième jour, adulte au vingtième jour et qu'il meurt le quarantième jour.
- ➔ Numéro du changement de saison. La fonction est quasi la même sauf qu'en plus (en allant dans le jardin), vous verrez quelques modifications.
- ➔ Points santé, édu et joie : Désignent l'état global du tamagoshi. Il faut savoir que votre tamagoshi est dans un état normal quand ces points sont proches de 100. Le but est d'avoir ces points le plus haut possible (200). Si les points deviennent trop faibles, le tamagoshi meurt (cf. Annexe Points en quotas pour plus de détails).

Actions commandées

Soit en réponse à une demande de tamagoshi, soit de votre propre chef, vous pouvez effectuer un certain nombre de choses. Ces commandes peuvent être insérer dans n'importe quelle phrase (tamagoshi regarde dans chaque ligne à son attention si elle détient un mot clé, et réagit en conséquence le cas échéant sauf si la ligne contient plus d'un ordre).

- ➔ Pas de mot reconnu : tamagoshi gagne un point éducation par tranche de 5 mots.
- ➔ « cache-cache » : cf supra.
- ➔ « balle » : cf supra.
- ➔ « dodo » ou « dormir »: vous le couchez pour qu'il dorme.
- ➔ « reveil » ou « reveiller » ou « reveille » : vous le réveillez avant qu'il ne fasse de lui-même … il se réveille alors grognon !
- ➔ un mot parmi (« biberon », « pate », « puree », « boulette », « viande », « bonbon», « gateau ») : tamagoshi mangera sans trop se poser de question. A vous de faire attention. (voir l'annexe Nourriture)
- ➔ « antibiotique », « Rennie », « piqure » ou « aspirine » : cf Médicaments.
- ➔ « fleur » : tamagoshi cueille la fleur et vous l'offre ou la met dans la maison (selon son humeur).
- ➔ « pomme » : Il mange une pomme s'il est dans le jardin et qu'il y en a au moins une.

- ➔ « livre » : Il lit un livre durant une durée aléatoire qui dépend aussi de son âge.
- ➔ « maison » ou « jardin » : le fait changer d'environnement.
- ➔ « travail » : le programme s'arrête après avoir effectué les sauvegardes nécessaires.

Deux jeux sont proposés :

Un message est affiché pour indiquer le début et la fin d'un jeu.

cache-cache : Le tamagoshi se cache dans une pièce de la maison. Le but est de trouver où. On énumère un lieu. Si c'est l'endroit de la cachette, c'est gagné et tamagoshi gagne des points de joie et se cache à nouveau. Sinon, on tente un autre lieu. Il est à noter que si le jeu dure trop longtemps, tamagoshi se lasse et perd des points de joie.

jeu de la balle : Vous lancez la balle, et tamagoshi tente de la rattraper. Pour ce faire, il suffit d'envoyer un caractère. La direction sera déterminée aléatoirement selon le caractère. S'il la rattrape, il gagne un point de joie. On peut répéter l'opération un certain nombre de fois jusqu'à ce que tamagoshi ou vous-même en ayez assez (envoyer « . » alors). Si tamagoshi a gagné trop peu de points de joie, il demandera de recommencer.

II- Automate d'Etats Finis (AEF)

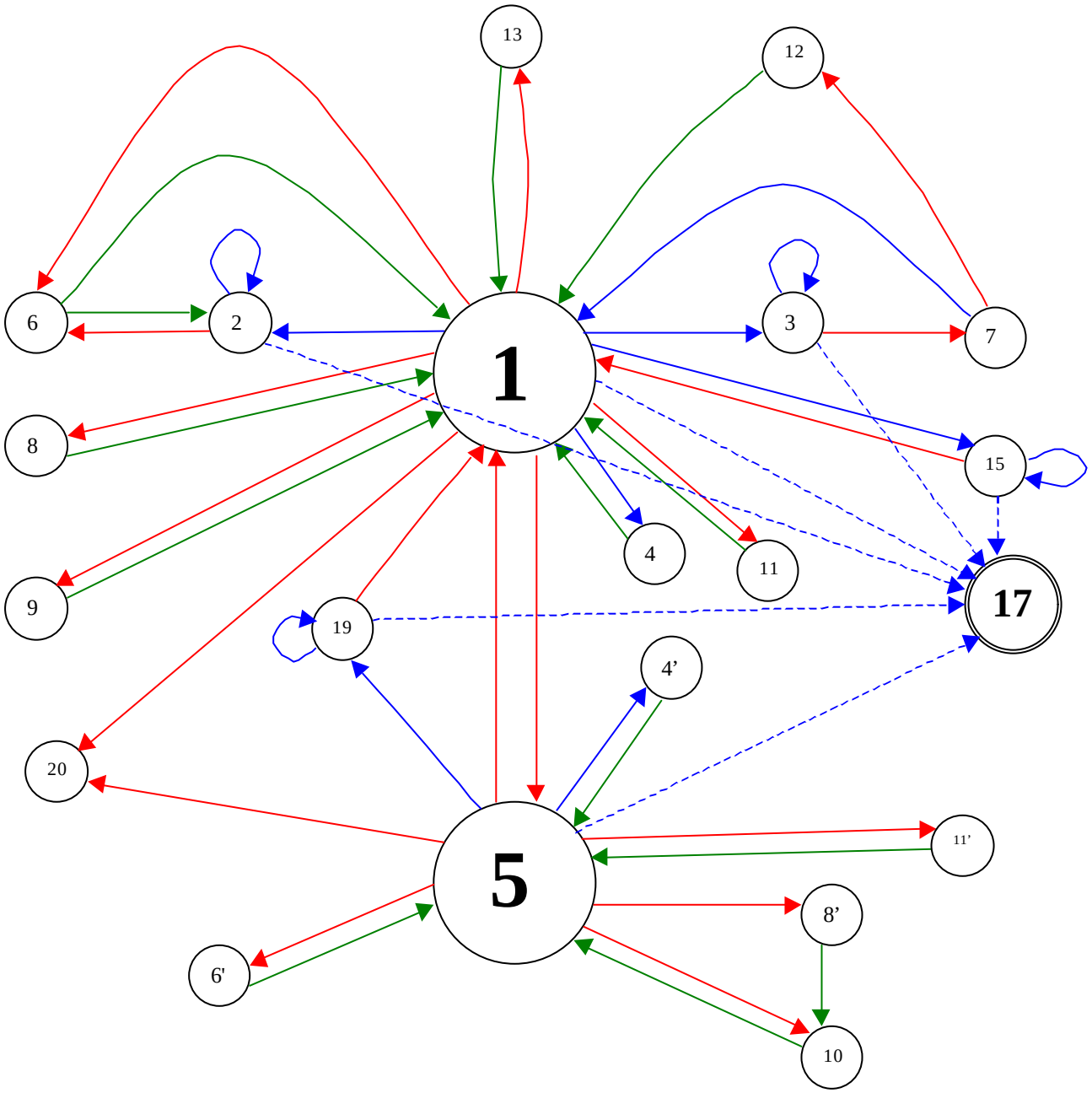
La première étape, une fois le cahier des charges fini, a été de définir avec soin les différentes classes que nous voulions implémenter. Nous avons donc défini tous les attributs qui nous semblaient nécessaires (ainsi qu'un petit nombre de méthode de base).
La grande difficulté a été ensuite de trouver les méthodes et surtout leur enchaînement. Pour nous aider, nous avons eu l'idée d'établir un AEF afin de bien planifier les événements possibles ainsi que leur séquencement.

Définition des états :

- 1 : Attente dans la maison
- 2 : faim
- 3 : sommeil
- 4 : ennui
- 5 : Attente dans le jardin
- 6 : Manger
- 6' : Manger une pomme
- 7 : Dormir
- 8 : Jouer à cache cache
- 8' : Jouer à la balle
- 9 : Lire
- 10 : cueillir une fleur
- 11 : Ecouter / Parler
- 12 : Perte de point de santé due à un réveil prématuré
- 13 : Réponse négative : impossible d'accomplir l'ordre demandé
- 15 : Malade
- 17 : Mort
- 19 : Envie de rentrer à la maison car faim ou sommeil
- 20 : sauvegarde de la partie

Les flèches correspondent aux transitions. Il y a trois types de transitions suivant l'événement :

- ➔ Ordre ou parole : flèche rouge
- ➔ Action automatique du tamagoshi : flèche bleue
- ➔ Action effectuée : flèche verte


13
12
6
2
1
3
7
8
15
4
11
9
19
17
4'
20
5
11'
8'
6'
10

III- Diagramme de Contexte Final (DCF)

Une fois l'AEF établi, beaucoup de méthodes nous sont apparues. De plus, nous avons pu trouver un algorithme pour les deux méthodes principales (Synchronisation et Analyse lexicale).
Nous pensons d'abord finir les algorithmes principaux avant de mettre un point final au DCF.

Voici la spécification de chaque classe :

Jeu_tmg : Il s'agit d'une sorte « d'arbitre ».
Tout d'abord, c'est ce module qui gère les différents objets. Instancier plusieurs objets de cette classe reviendrait à créer plusieurs tamagoshis (ce que nous n'avons pas choisi de faire). Il comporte, entre autres, un affichage, un tamagoshi, une maison et un jardin . C'est lui qui créé la nourriture, les médicaments et les activités.
Ensuite, c'est lui qui gère l'interactivité entre l'utilisateur et le tamagoshi. Il se charge de reconnaître les mots tapés par l'utilisateur et activer la méthode (ou les) adéquate(s). Mais c'est aussi lui qui déclenchera les actions automatiques du tamagoshi (comme la faim, le sommeil…) et changera l'état de ce dernier.
Enfin, c'est ce module qui se charge du déroulement correct du jeu : il initialise le jeu, rafraîchit l'affichage et se charge des sauvegardes.

Tamagoshi : Il s'agit de l'animal virtuel en lui-même.
Le tamagoshi évolue par héritage. On pourra donc distinguer trois formes de tamagoshi : bébé, adolescent, adulte. Chacune des formes héritant du moule tamagoshi ; le constructeur recopie les données de la précédente forme. Les méthodes qui diffèrent suivant la forme sont déclarées en virtuel dans le moule. Ceci permet d'avoir des réactions différentes et un vocabulaire différent selon l'âge.

Environnement : Il s'agit du milieu dans lequel le tamagoshi évolue. On notera qu'il s'agit ici de la maison et du jardin (classes qui dérivent de Environnement) mais que l'on pourrait en imaginer d'autres. L'intérêt peut apparaître comme surtout graphique, mais la classe Jardin possède une fleur et des pommes qui font partie intégrante du jeu. On peut aussi noter que certaines activités sont limitées à un seul environnement (on ne peut pas jouer à la balle dans la maison : on risque de casser quelque chose ; les livres sont dans la maison…).

Nourriture et *Médicaments* : Le principal intérêt de ces classes est de pouvoir associer un affichage différent à chaque type de nourriture et de médicament (ce qui n'est pas fait en pratique).

Activités : Elle permet d'ajouter des fonctionnalités au tamagoshi simplement. Il faut quand même bien penser à ajouter la commande dans Jeu_Tmg. Si l'activité change le tamagoshi, il faut aussi penser à ajouter une méthode dans la classe correspondante. Pour donner un exemple concret, nous avons implémenté une classe lire_un_livre et cueillir_une_fleur dérivant directement d'Activités, ainsi que jouer_balle et jouer_cache_cache qui dérivent d'abord d'une classe Jeu (qui, elle-même, dérive d'Activités). Bien entendu plus cette classe sera étoffée, plus le jeu sera intéressant.

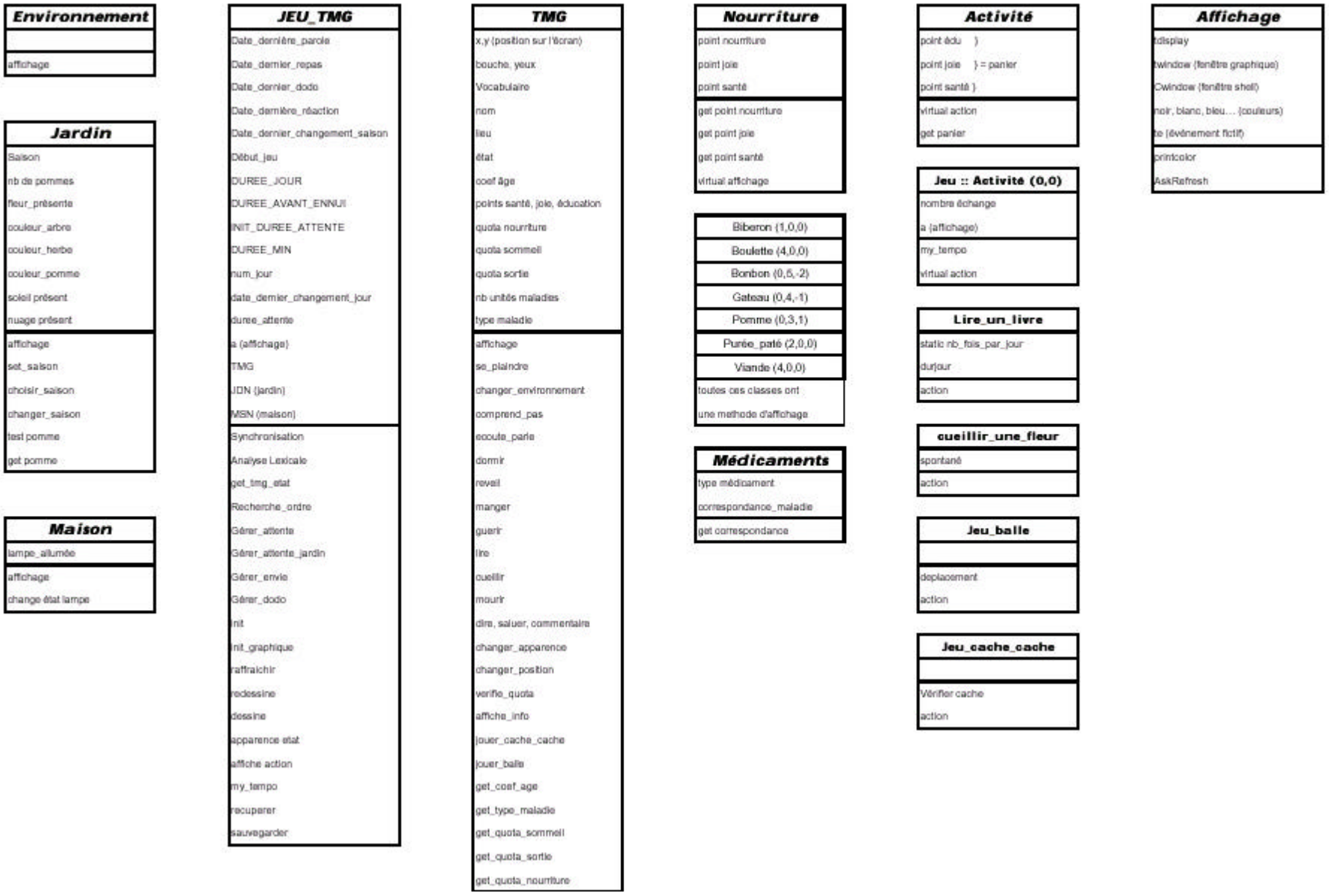
La classe *Affichage* sert surtout à définir les différentes couleurs, la fenêtre graphique, etc…

La communication entre les classes se passe de la manière suivante :

La classe Jeu_tmg comporte, au moment adéquat, les objets voulus. Entre autres, il possède un objet de type Tamagoshi (qui est en fait un tmg_bébé, tmg_ado ou tmg_adu). Il accède et modifie régulièrement aux points (santé, édu et joie) et à l'état du tamagoshi. Les autres classes font de même. Nous avons donc préféré mettre ces variables en public même s'il aurait été plus correct de faire des assesseurs. Nous avons procédé de même pour les variables très utilisées.
En revanche, la forme du tamagoshi n'appartient qu'à lui-même et il est seul à être susceptible d'accéder. Ils sont en protected dans le moule pour des questions d'héritage.
Pour ce qui est des variables dont on ne voulait des accès qu'en lecture ou qu'en écriture, des assesseurs ont été créés.

Enfin, les méthodes (hors de celles de Jeu_tmg) qui ont besoin des objets créés se voient passer les objets concernés en paramètre par Jeu_tmg.

DIAGRAMME DE CONTEXTE FINAL



IV- Travail effectué et problèmes rencontrés

La répartition du travail a été plutôt floue : en effet, nous avons beaucoup travaillé ensemble et en même temps sur quasi toutes les phases de notre projet. On peut noter que jusqu'au Diagramme de Contexte Final, nous avons toujours travaillé ensemble. Nous avons aussi élaboré la plus grosse partie de l'algorithme des méthodes ensemble.

Après cette phase, nous nous sommes un peu plus répartis le travail. On notera que Jacqueline Bollet s'est beaucoup appliquée à mettre en place l'interface graphique à l'intérieur du projet. Thierry Grenier s'est surtout occupé des constructeurs (et destructeurs), ainsi que des formules gérant les diverses attentes (celles du genre (6*TMG->get_coef_age() - 3)*(DUREE_JOUR/24)).

Le débogage a aussi été fait conjointement avec de nombreux lancements du jeu. Ceci n'est pas très étonnant vu la durée d'une partie. C'est entre autre pour faire ces tests que nous n'avons pas mis de limite supérieure ni inférieure à la durée virtuelle d'un jour.

Les problèmes ont été multiples et de sources variées. Le premier problème est surtout venu du fait que les cours de Conception Orientée Objet étaient en retard sur notre avancement.

Ensuite, est venu le problème de la méconnaissance du C++. Il a joué à plusieurs niveaux : nous avons du ajouté un bout de code C pour les fonctions dont nous n'avons pas trouvé l'équivalence en C++ ; nous avons aussi perdu presque une semaine à cause d'un problème simple que nous n'avions pas compris en cours (nous ne pouvions avancer car ce problème survenait lors de la phase d'édition des liens, ce qui nous empêchait de tester notre programme).

Enfin, comme pour tout projet, certains aléas se sont ajoutés :

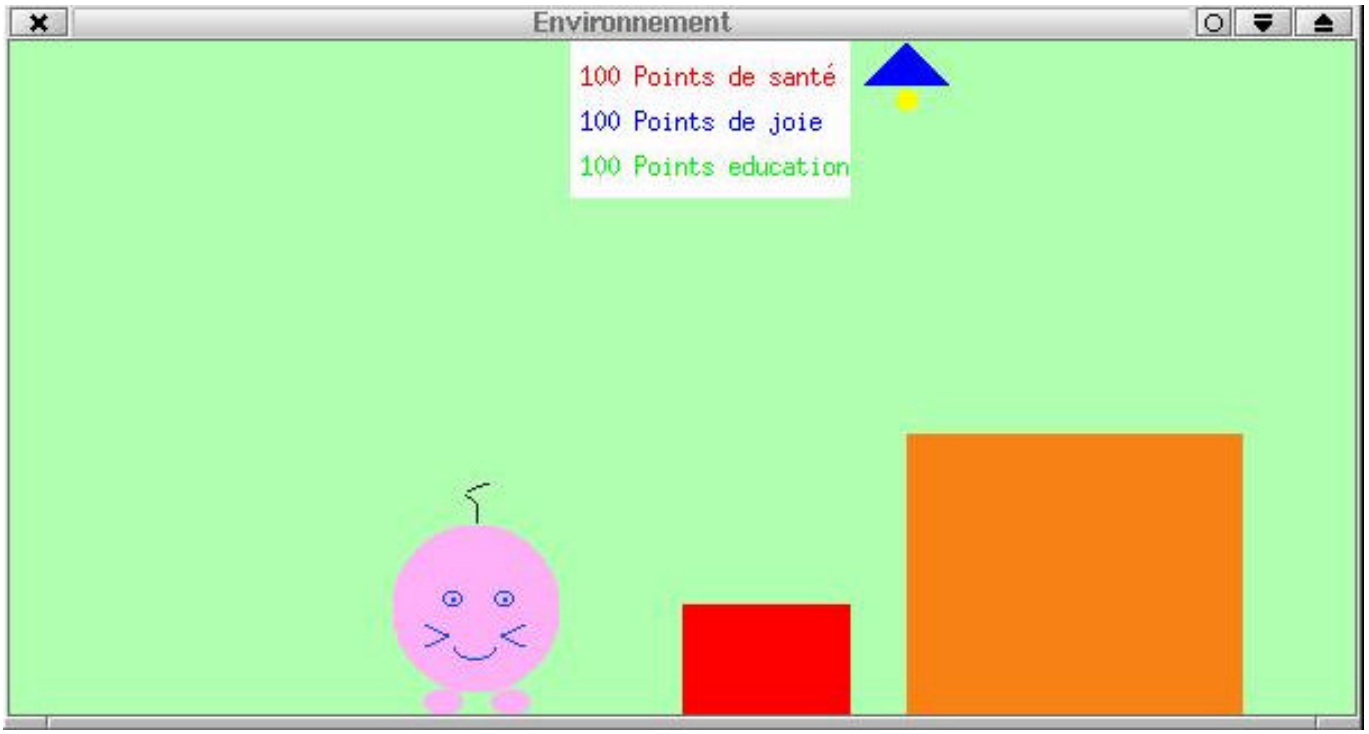
Problème de Makefile (qui fonctionnait correctement sur les « vieilles » machines mais qui ajoute une erreur sur les nouvelles).

Problème de graphisme (qui a été ajouté après que le tamagoshi fonctionne normalement, ceci a occasionné certaines modifications qui ajoutent une trop grande dépendance entre les modules…). Notamment, nous n'avons pas pu faire afficher les objets comme nous l'avions prévu car cela demandait trop de temps pour la définition.

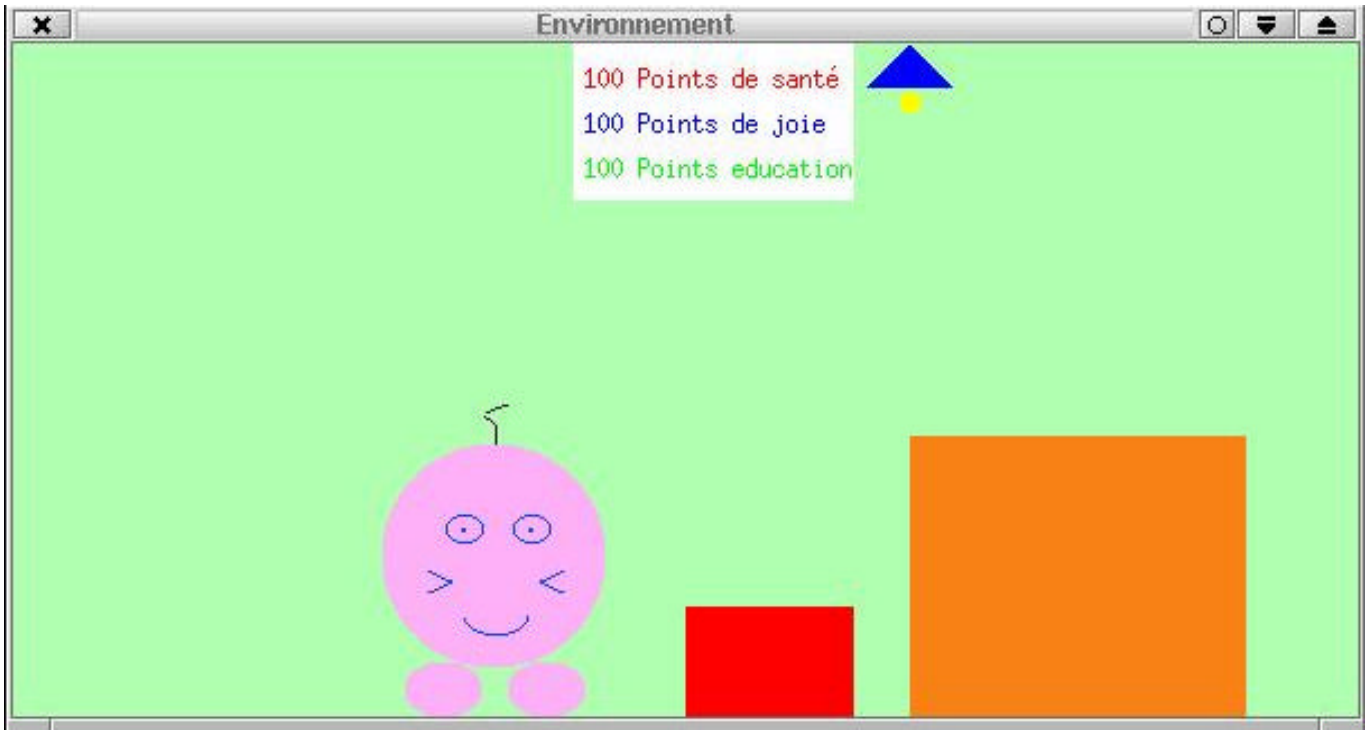
V- Affichage

Voilà quelques exemples d'affichages de la fenêtre graphique :

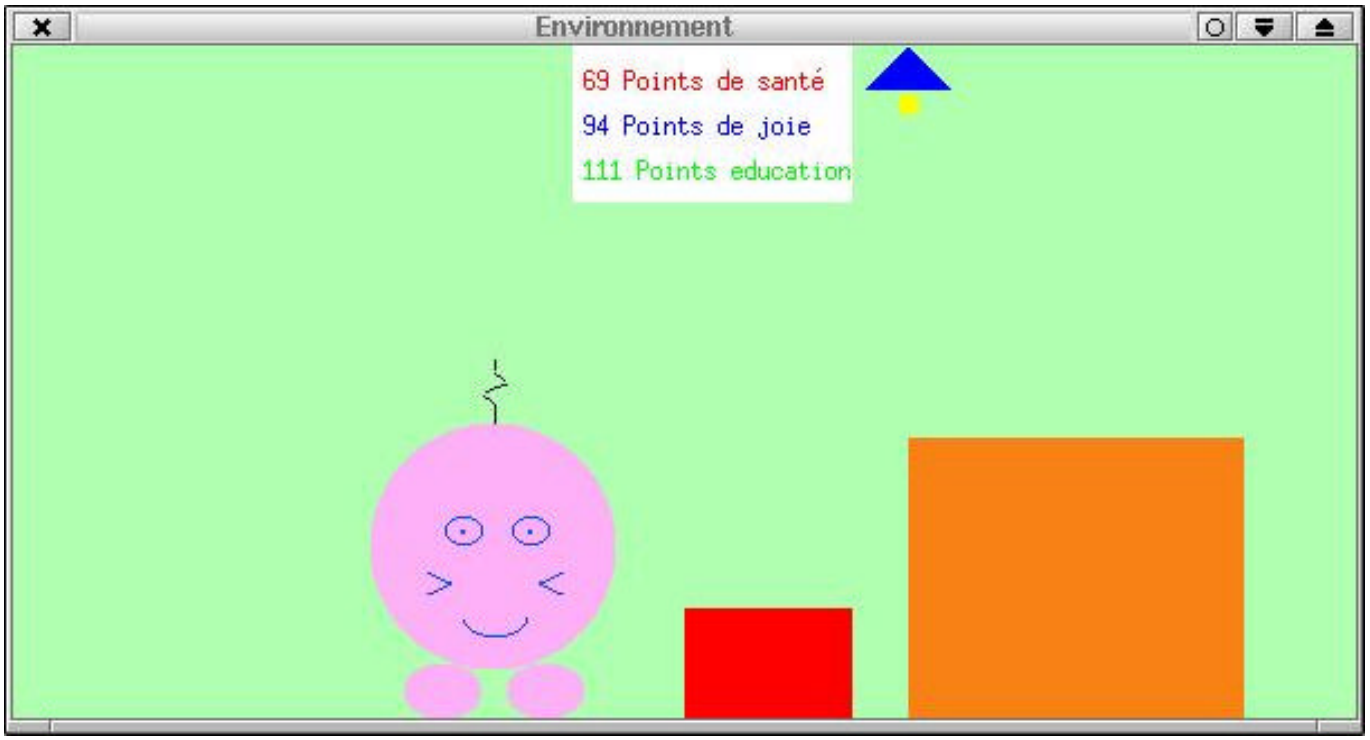
Tamagoshi à l'état bébé :



tamagoshi à l'état adolescent :

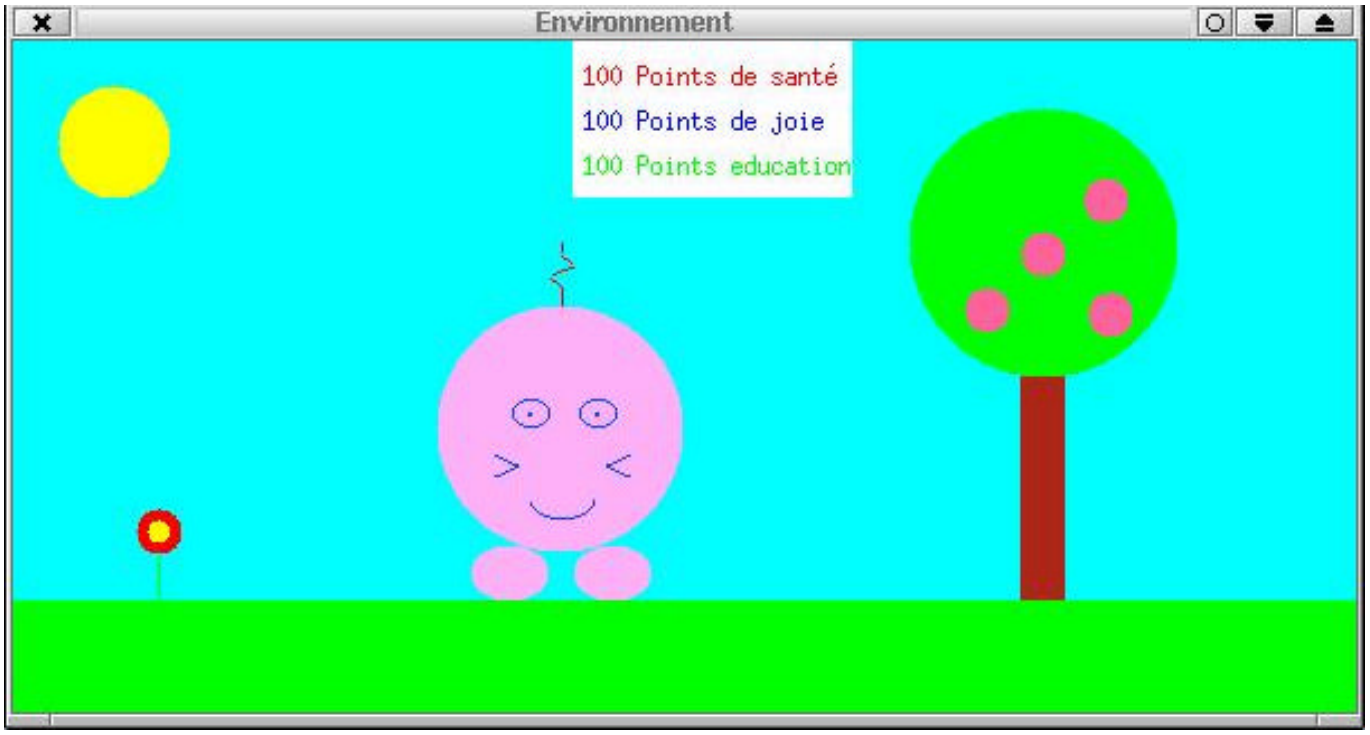


tamagoshi adulte :

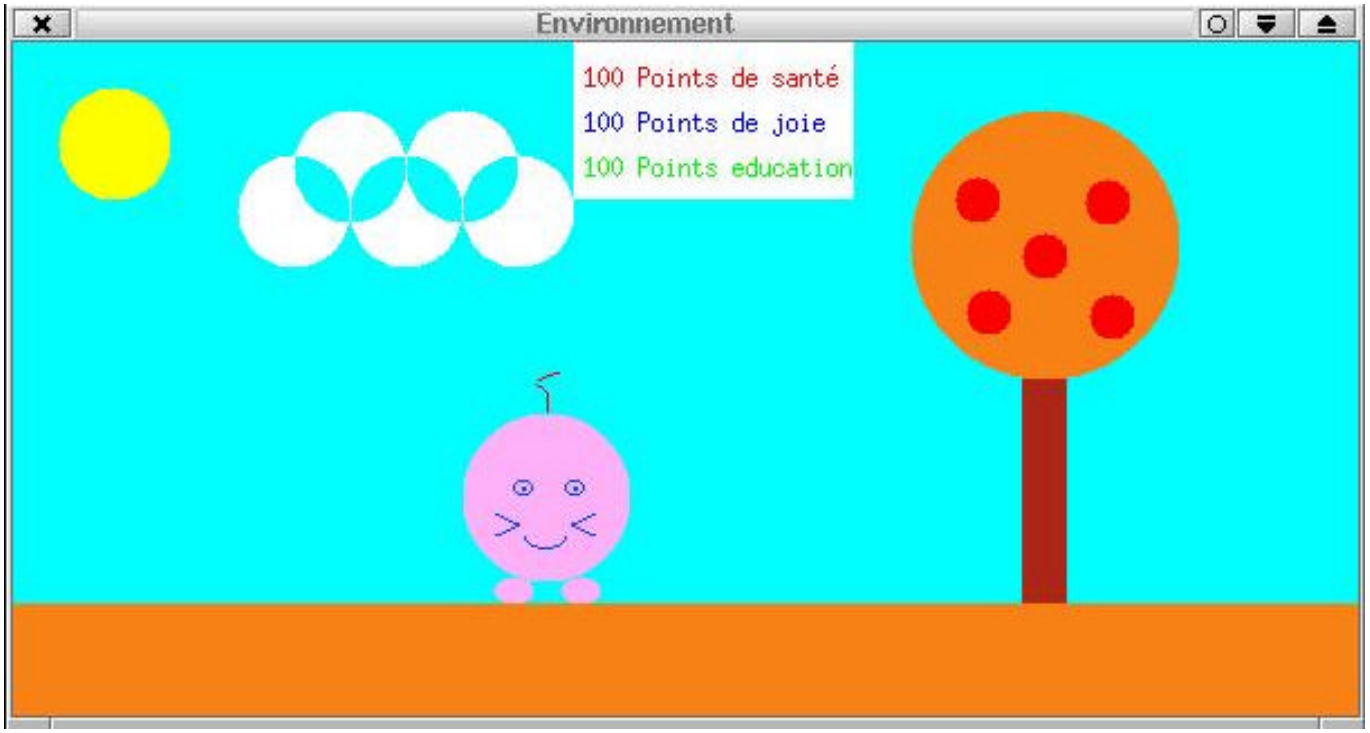


Rapide tour des saisons :

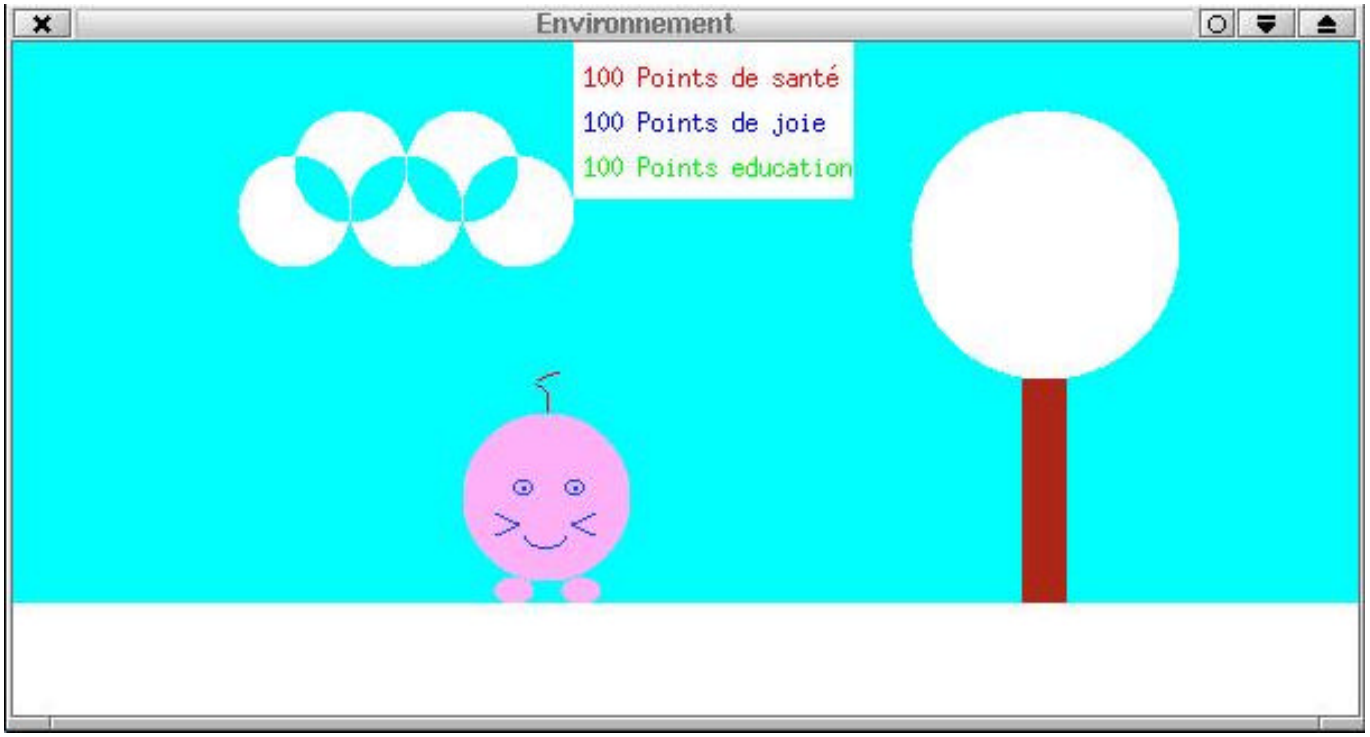
Eté :



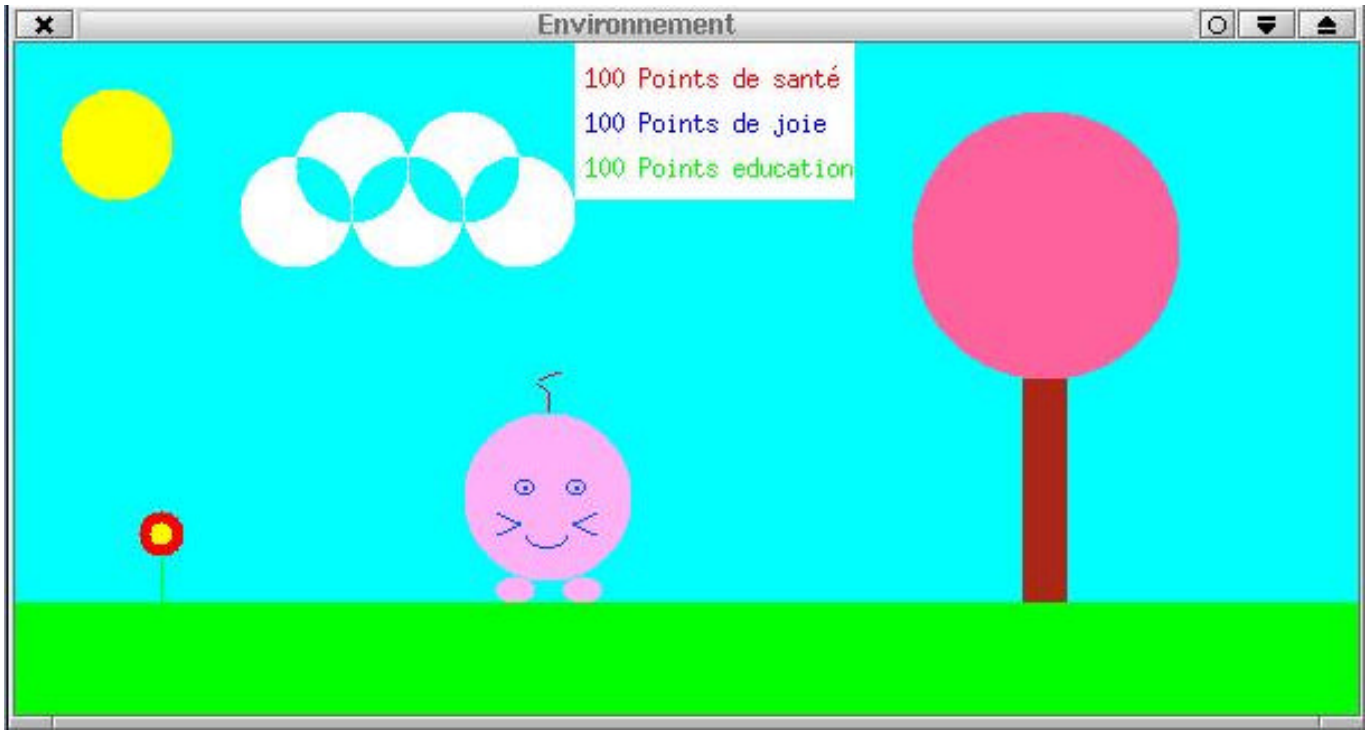
Automne :



hiver :

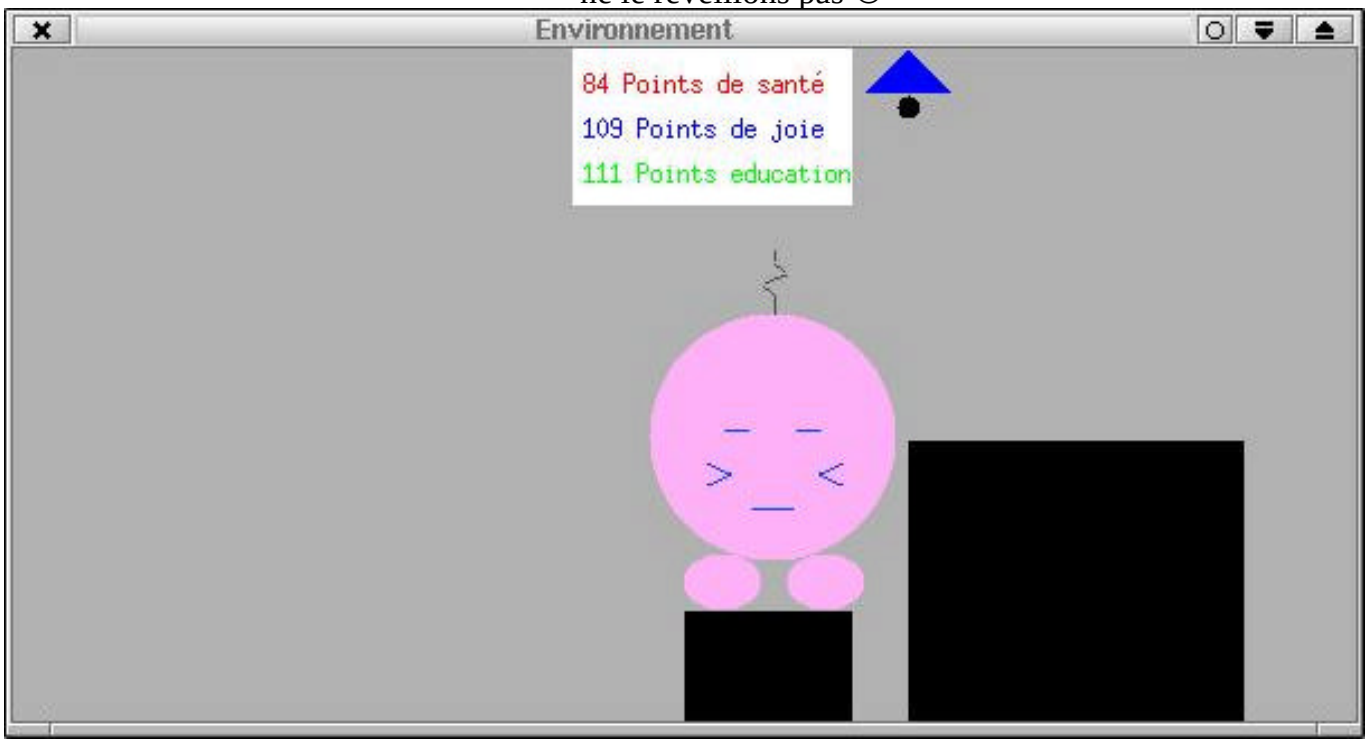


printemps :



enfin le voilà endormi :

ne le réveillons pas ☺



Conclusion :

L'objectif de créer un animal virtuel (dans un langage orienté objet) en interaction avec son environnement (auquel l'utilisateur, son maître, fait finalement partie) est bien atteint.
Nous regrettons cependant qu'au début du projet, nos connaissances en Conception Orientée Objet et surtout nos compétences dans ce domaine, étaient trop vagues. Nous nous sommes aperçus (mais trop tard pour la changer) que la conception n'était pas très judicieuse en certains points… Si le projet était à refaire, nous ne le referions pas de la même manière.
Ceci étant, le projet nous a pleinement profité et nous pensons beaucoup mieux comprendre maintenant ce que l'on attendait de nous et ce qu'un langage orienté objet peut apporter.

ANNEXES

POINTS ET QUOTAS

A tout moment, l'état du tamagoshi est défini par un certain nombre de points et quotas. Voici leur explication en quelques mots :

Coefficient âge :

Permet de calculer ce qui doit être fait à son âge et détermine son âge (1 = Bébé ; 2 = adolescent ; 4 = Adulte). Ce coefficient sera utilisé dans de nombreux cas comme pour le calcul de la résistance à une maladie, la fin de partie, etc…

Les points, initialisés en début de partie, varient selon la manière dont l'utilisateur joue. Si l'un de ces paramètres tombe à zéro à un moment quelconque de la partie, le jeu est perdu.

Points de santé :

Ils démarrent à 100 mais sont limités à 200. Plusieurs facteurs peuvent les faire changer : la maladie, la malnutrition, etc… S'ils tombent à zéro, le tamagoshi meurt de maladie.

Points de joie :

Ils démarrent à 100 et sont limités à 200. On en gagne quand on joue avec lui, quand on lui donne des bonbons… Puis il est joyeux, plus il est actif (parle plus). S'ils tombent à zéro, tamagoshi se ferme du monde extérieur.

Points d'éducation :

Ils démarrent à 100 et sont limité à 200. S'ils deviennent inférieurs à 100, le tamagoshi commence à être vulgaire. Si les points sont toujours à 50, le tamagoshi fugue de la maison.

Enfin, viennent les quotas, qui sont initialisés en début de journée, et définissent ce qu'il reste à faire pour rendre tamagoshi heureux. S'ils ne sont pas satisfaits, le tamagoshi tombe malade (voir l'annexe Maladie et Médicaments)

Quota nourriture :

Tamagoshi doit manger un certain nombre de fois par jour (4 à 8 quand bébé, 2 à 4 quand adolescent, 1 à 2 quand adulte).

Quota sommeil :

Durée pendant laquelle il dort dans une journée (12h quand bébé, 6h quand adolescent, 3h quand adulte). Avec ce système, il ne dort pas « trop » quand il est adulte. S'il ne dort pas assez, perd des points de santé et de joie. Il ne peut pas trop dormir puisqu'il se réveille quand il n'a plus sommeil.

Quota sortie :

Nombre de fois où il peut sortir pour jouer/prendre l'air (0/1 quand bébé, 1/2 quand adolescent, 2/4 quand adulte) : premier chiffre : hiver/automne, deuxième : été/printemps.

ANNEXE

MALADIES ET MEDICAMENTS

Quand tamagoshi tombe malade, il faut le guérir. Mais pas n'importe comment !

- ➔ **Grippe :** 5 unités_maladies. Si le quota sortie est dépassé, il attrapera immanquablement la grippe. Il faudra alors lui administrer un *antibiotique*.

- ➔ **Indigestion :** 3 unités_maladies. Si tamagoshi mange ce qu'il ne doit pas (par exemple si vous lui donner de la viande alors qu'il est bébé) ou s'il mange trop, il a mal au ventre. Il faut lui donner un *Rennie* et la douleur passera.

- ➔ **Migraine / faiblesse :** 2 unités_maladies si le quota nourriture est trop bas ou 4 unités_maladie si le quota sommeil trop bas. Une *aspirine* et tamagoshi sera vite sur pattes.

- ➔ **Jaunisse :** 2 unités_maladies. Si tamagoshi ne sort pas assez, il peut attraper cette maladie fantaisiste. Il faudra alors lui faire une *piqûre*.

NOURRITURE :

Chaque aliment est plus ou moins nutritif. Tamagoshi doit manger 4 à 8 unités de nourriture par jour. Il ne peut manger certains types de nourriture que s'il est assez évolué.

Biberon de lait : 1 unité.
Purée / pâté : 2 unités (interdit si bébé).
Viande et boulettes : 4 unités (uniquement si adulte).
Bonbons / gâteau : 0 unité (donnent des points de joie mais attention aux excès).
Pomme : 1 unité (donne aussi 2 points de joie). Attention il n'y a que 5 pommes sur l'arbre par année. Attention les pommes ne sont mangeables qu'en automne. En été elles sont mangeables mais ne procurent pas de point de joie (trop vertes).