Projet MobiNet

Plateforme pédagogique pour l'informatique, les mathématiques et la physique

Rapport de projet de monitorat 2002-2004
http://mobinet.imag.fr

Sylvain Lefebvre *Moniteur en informatique, INPG*

Tuteur du projet : Fabrice Neyret *Chargé de recherche au CNRS*

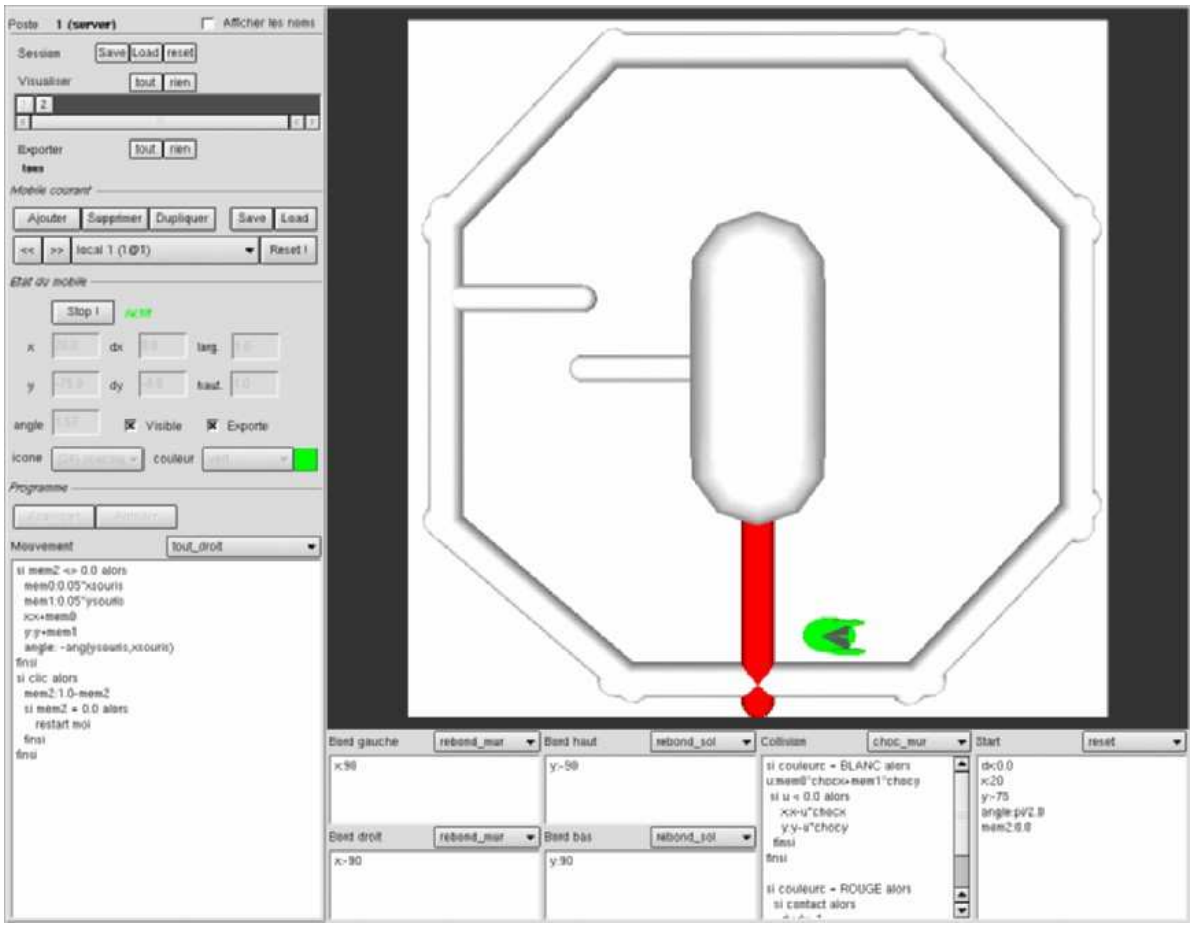


FIG. 1 – Le logiciel MobiNet

1 Introduction

MobiNet est un logiciel développé dans le cadre du projet de monitorat en relation avec les "semaines découverte ingénieur" INPG. Il permet d'appliquer de manière concrète et ludique les notions de mathématique et physique acquises au lycée. Il simule graphiquement un ensemble d'*objets mobiles* créés interactivement par l'utilisateur (élève, professeur ou autre), dont ce dernier spécifie les *variables d'état* et le *programme de comportement* au moyen d'un langage dédié.

MobiNet, à travers les exercices de travaux pratiques qui l'accompagnent, permet aux étudiants de découvrir comment un processus complexe peut être décomposé en éléments simples puis formalisés pour devenir maîtrisables. Grâce à cette approche, les élèves peuvent mettre en pratique leurs connaissances et élaborer des applications complexes, comme la simulation de phénomènes physiques, mais aussi des jeux vidéos. Le travail collaboratif est également mis en avant.

Ce projet a été expérimenté dans le cadre des "semaines découverte ingénieur" organisées par l'INPG avec plusieurs classes de lycéens.

1.1 Bref historique de MobiNet

Depuis quelques années les filières scientifiques attirent de moins en moins d'étudiants. Dans le rapport du Sénat intitulé *La culture scientifique et technique pour tous : une priorité nationale*, il est souligné que la désaffection pour les sciences touche essentiellement les études supérieures. Dans ce contexte, l'institut National Polytechnique de Grenoble a mis en place dès décembre 2002 la *semaine découverte ingénieur*. Cet événement bi–annuel vise à faire découvrir à des classes de lycéens (première et seconde) les métiers de l'ingénieur et le monde de la recherche scientifique. Pendant une semaine, des classes de lycées visitent les laboratoires de la région Grenobloise et participent à divers ateliers ayant pour thèmes les sciences et les métiers de l'ingénieur. C'est précisément dans le cadre de la semaine de l'ingénieur 2002 que le projet MobiNet a vu le jour au sein de l'équipe iMAGIS du laboratoire GRAVIR. Le projet a été réalisé par Joelle Thollot (Maître de Conférence à l'ENSIMAG), Fabrice Neyret (Chargé de Recherche au CNRS), Samuel Hornus (Doctorant UJF, Moniteur à l'UJF) et Sylvain Lefebvre (Doctorant UJF, Moniteur à l'INPG). Les premières séances de travaux pratiques avec les lycéens se sont déroulées en décembre 2002 dans l'Atelier Réalité Virtuelle (ARV) situé dans les locaux de l'ENSIMAG à Montbonnot.

2 Motivations pédagogiques

L'objectif affiché autour duquel MobiNet a été conçu est de faire découvrir la conception des jeux vidéos aux lycéens. La création de jeux vidéos constitue en effet un excellent terrain d'apprentissage. En premier lieu, leur conception nécessite à la fois des notions de physique et de mathématique correspondant (pour les plus simples) aux programmes de première/terminale. Un jeu vidéo est également un très bon exemple de processus complexe obtenu par l'assemblage de composants plus simples. Le but est ici de "déconstruire" un objet complexe du quotidien (le

jeu) afin de le mettre à la portée des étudiants, de le “démystifier”. De plus la conception d'un jeu mimant un jeu du monde réel requiert une étape *modélisation*, que l'on retrouve à la base de la démarche scientifique. Enfin, les jeux vidéos sont un thème porteur auprès des lycéens. Ceci crée une bonne motivation chez les élèves qui acceptent dès lors plus facilement d'effectuer les étapes de prise en main du logiciel MobiNet. Cette motivation est également renforcée par le fait qu'au cours de l'atelier les lycéens créent un jeu complet auquel ils peuvent jouer ensemble en réseau.

Néanmoins sous cet objectif se cachent des buts plus profonds :

- **Permettre aux élèves de s'approprier les notions vues en cours**, en donnant à celle-ci un sens concret (les formules créent le mouvement des objets, ...).
- **Initier les élèves à la démarche scientifique**. Les objets visibles se caractérisent par différents attributs mesurables, correspondant aux *variables d'état*, qui constituent ainsi un *modèle mathématique* du réel. Faire évoluer un objet revient à modifier au cours du temps ses variables d'états, ce qui s'exprime par des formules (a minima de type `x : x+1` , ou `x : 10*cos(t)`), permettant de simuler des *modèles physiques* de comportement.

Les motivations sous-jacentes sont multiples :

- L'importance de **manipuler soit-même** pour conforter l'apprentissage (appropriation) n'est plus à démontrer.
- Manipuler soit-même et avoir un **but concret** est **motivant** pour l'élève (alors que l'on reproche à contrario à l'enseignement traditionnel d'être trop abstrait et désincarné au goût des élèves d'aujourd'hui).
- Jouer sur les paramètres d'une formule contrôlant le mouvement d'un objet permet de **se constituer une intuition liant le sens d'un paramètre numérique** (et de ses variations) à un comportement physique ou mathématique (amplitude, fréquence, échelle...).
- Les élèves **apprennent par essai-erreur** : l'erreur est permise, n'est en aucun cas une sanction, et permet même de comprendre mieux (nous incitons les élèves à expliquer a posteriori les comportements inattendus).
- La démarche suggérée correspond précisément à la façon dont nombre d'**ingénieurs, programmeurs et chercheurs manipulent ces notions dans leur métier**, alors que les élèves n'ont généralement aucune idée que c'est à ce type d'usage que leur savoir leur servira – s'ils ne se détournent pas auparavant de la filière scientifique.

Afin d'atteindre ces objectifs nous avons fait le choix de concevoir un outil n'incluant pas déjà de notions de physique : tout comportement ou phénomène à simuler doit être programmé (gravité, dynamique ou même rebond sur les bords de l'écran).

3 Le logiciel MobiNet

MobiNet est un logiciel permettant de créer des applications complexes en combinant des objets dont le comportement et l'apparence peuvent être déterminés soit directement (entrée utilisateur) soit par un ensemble de formules. Pour plus de clarté nous appelons ces objets des *mobiles*. Chaque mobile est décrit par un ensemble de *variables d'état* (forme, couleur, position, ...). La figure 2 présente l'interface du logiciel ainsi que les divers éléments qui la composent.

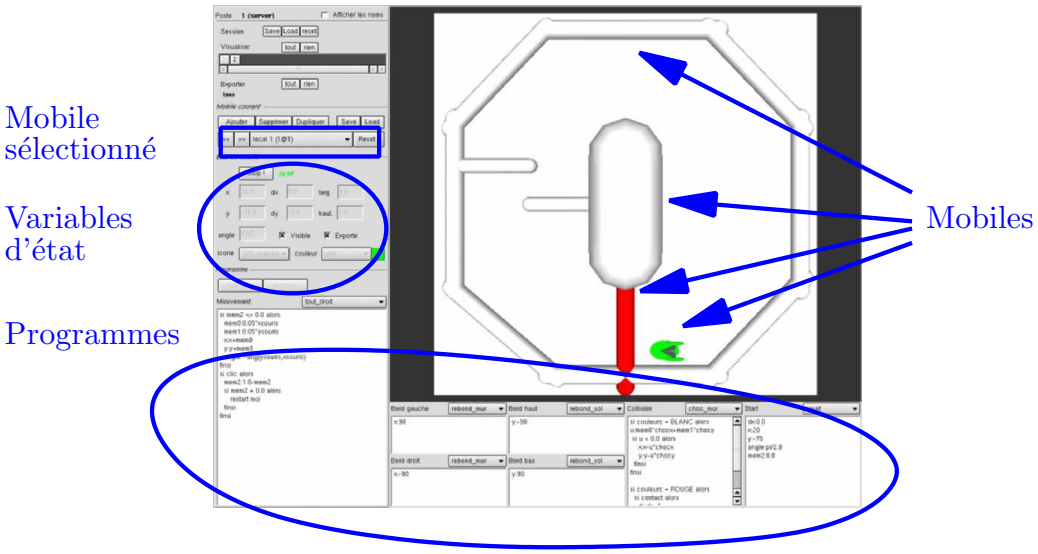


FIG. 2 – *L'interface de MobiNet*

MobiNet propose un véritable langage de programmation permettant de décrire l'évolution des variables d'états des mobiles mais aussi leurs interactions (le comportement d'un mobile peut dépendre de l'état d'un ou plusieurs autres mobiles).

Un soin particulier a été apporté lors de la conception afin de faciliter la prise en main du logiciel. L'interface est épurée et il n'est pas nécessaire de maîtriser l'ensemble des fonctionnalités pour l'utiliser. La syntaxe du langage a été élaborée en collaboration avec des professeurs de physique et mathématique afin de correspondre au mieux aux habitudes des lycéens. Nous avons également veillé à réduire les entrées clavier et à mettre en place un système de fonctions prédéfinies pour éviter la répétitions de tâches déjà assimilées par les élèves. Le mode d'emploi de MobiNet est joint en annexe de ce document (annexe A).

Un des atouts de MobiNet est de permettre la visualisation et l'interaction avec les mobiles d'un autre ordinateur. Ainsi, lors de travaux pratiques dans une salle équipée d'un réseau informatique, il est possible de voir et d'interagir avec les mobiles de tous les postes. Il est donc possible de créer des travaux pratiques basés sur le travail collaboratif entre les divers postes. Les différentes modalités de travail en réseau sont :

- travail indépendant sur chaque poste ;
- visualisation de n'importe quel poste depuis le poste maître (le votre) ;
- visualisation superposé de tous les postes (e.g. pour projection murale) ;

– travail en binômes ou en trinômes (ou autre) ;
– travail collectif sur plusieurs postes.

Les activités que nous avons organisé autour de cette fonctionnalité montrent qu'elle permet d'impliquer beaucoup plus les élèves qui échangent alors des conseils et travaillent ensemble pour atteindre leur objectif commun.

3.1 Que peut–on faire avec MobiNet ?

MobiNet permet donc de créer une large gamme d'applications. Voici quelques exemples que nous fournissons avec MobiNet (voir figure 3) :

– système solaire animé (possibilité de changer de référentiel, de voir tourner les planètes),
– jeux vidéos simples (tennis, astéroids, casse-brique, course de voitures, ...),
– figures géométriques animées (parabole, cercle inscrit, ...)
– simulation de la propagation des influs nerveux, des contractions musculaires,
– solides articulés
– ...

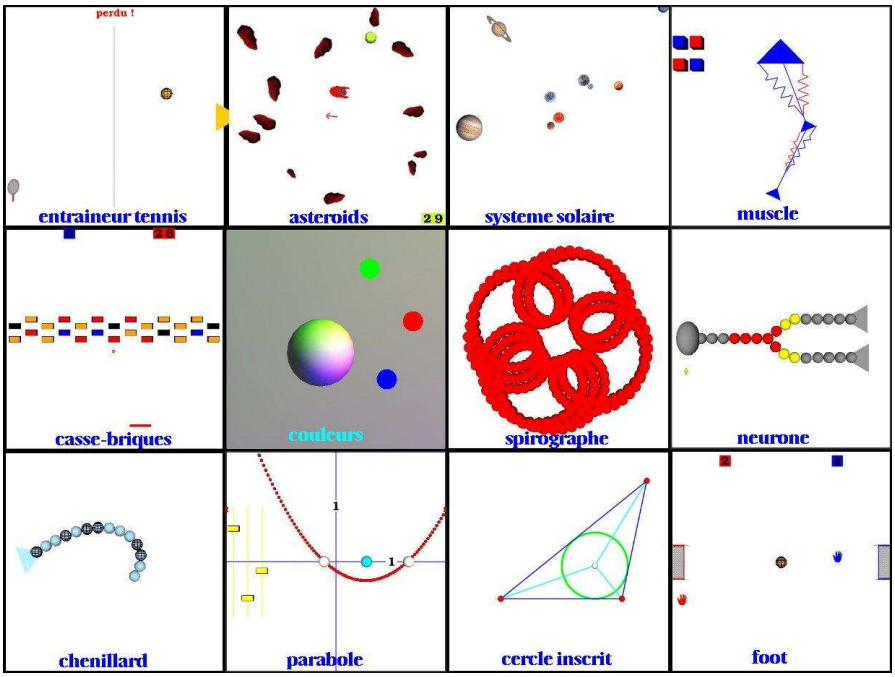


FIG. 3 – Différentes applications créées en programmant les mobiles de MobiNet

Chacun de ces exemples est obtenu en combinant des mobiles et en *programmant* leur comportement à l'aide de MobiNet. C'est en cela que MobiNet constitue une *plateforme pédagogique ouverte* : son utilisation peut être adaptée à différents domaines et à différents niveaux de complexité. Le langage de programmation est très complet et dépasse largement les besoins nécessaires pour les travaux pratiques destinés aux classes de lycées.

3.2 Utilisation pédagogique

Nous avons utilisé Mobinet pour organiser un atelier dans le cadre des “semaines découverte ingénieur” INPG. Les buts étaient assez généralistes et multiples puisqu'il s'agissait d'illustrer les métiers de l'informatique, de démontrer l'utilité des mathématiques et de la physique, et plus largement de faire découvrir la recherche et la démarche scientifique aux lycéens. Cette expérience est décrite plus en détail dans la section 4.

Néanmoins l'utilisation de Mobinet peut être déclinée de multiples autres manières pour être introduite dans les classes :

- TD illustrant des notions de cours, éventuellement en partie pré–construit par l'enseignant ;
- visualisation d'une illustration du concept du cours (préparé par l'enseignant) ;
- exercices à la maison ;
- clubs
- TPE
- ...

4 L'atelier MobiNet

4.1 Principe

L'atelier Mobinet est prévu pour accueillir une quinzaine d'élèves (une trentaine en binômes) pendant trois heures. Chaque élève dispose d'un poste informatique relié au réseau. La première partie de la séance est dédiée à la prise en main du logiciel. L'accent est mis sur l'approche essai–erreur. Les élèves sont encouragés à tester différentes solutions et à comprendre les erreurs qu'ils effectuent.

A mi–séance, après la pause, lorsque les principes essentiels du logiciel sont compris et que des exercices simples ont été effectués (mobile décrivant un cercle, mobiles reliés par un ressort, ...), des groupes de deux postes sont définis afin de réaliser un travail collaboratif. Le but de chaque groupe est de créer un petit jeu de tennis (connu sous le nom de *pong*). Chaque poste héberge un coté du terrain de jeu (cage de but, raquette). Le résultat de cet exercice est illustré en figure 4. La première étape est la création du mobile représentant la balle rebondissante. Ensuite, les mobiles représentant les cages et les raquettes sont créés sur chaque poste. Une fois ces étapes accomplies, les deux postes du groupe peuvent jouer ensemble. C'est un temps très dynamique où les élèves s'impliquent, communiquent et se ré–expliquent intensément. L'énoncé utilisé lors de l'atelier MobiNet de la “semaine découverte ingénieur” est joint à ce document en annexe (annexe B).

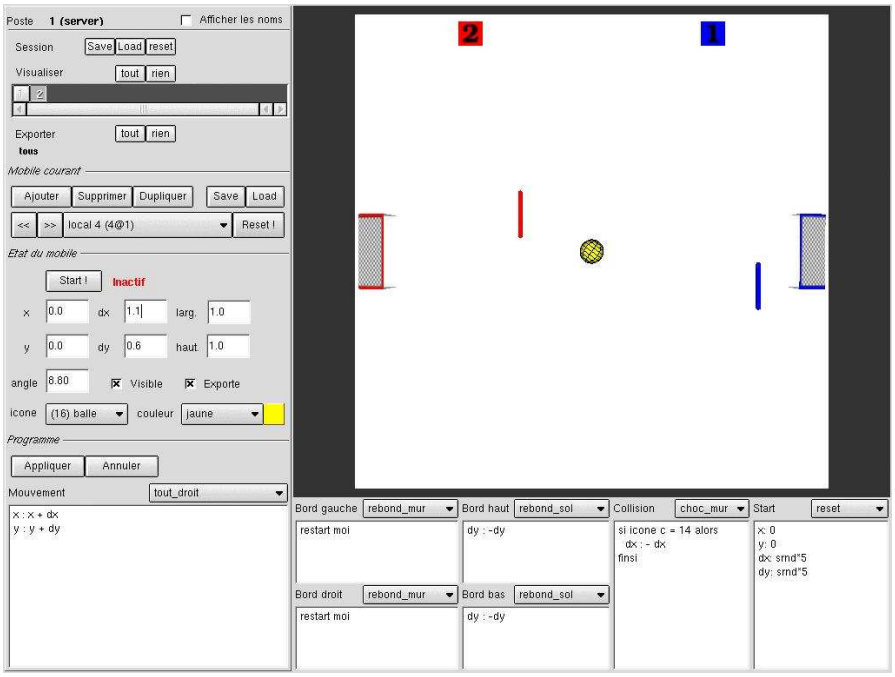


FIG. 4 – Le jeu de tennis complet. Chaque coté est réalisé sur un poste informatique différent. La balle est gérée par un seul des deux postes.

4.2 Encadrement

L'encadrement est organisé autour d'un animateur qui est chargé de donner le rythme des exercices et de faire les explications communes. Afin d'appuyer ses explications, il dispose d'un poste maître relié à un grand écran sur lequel il est possible d'afficher les mobiles des différents postes de la salle (éventuellement tous). Au début il est chargé de présenter le lieu, les intervenants et le but de l'atelier. Néanmoins, dans le souci de laisser aux élèves le temps d'expérimenter différentes solutions ses interventions sont progressivement espacées.

Un groupe d'encadrants (généralement trois encadrants pour dix postes) est chargé d'assister les élèves sur chacun des exercices. Il s'agit tout d'abord d'éviter une perte de temps due à une méconnaissance du logiciel (manipulation de l'interface) ou au manque d'habitude de l'outil informatique en général. Mais la partie la plus importante du rôle des encadrants est d'aider les élèves à **comprendre** leurs erreurs, à faire le **lien entre les formules** qu'ils ont entrées et qui guident les mobiles **et leur mouvement observable, concret**.

4.3 Retour d'expérience

Nous avons fait passer environ 120 élèves de première et de seconde en 2002-2003 (cet atelier se poursuit en 2004), en 8 groupes effectuant chacun un TP de 3 heures. Les élèves sont d'origines culturelles diverses, et n'ont reçu aucune préparation préalable. Quelques élèves tentent de raisonner sur papier, puis vérifient sur programme, progressant par étapes. D'autres essaient tout et n'importe quoi, pour se rendre compte rapidement qu'un peu de méthode s'avère plus efficace.

Parmi les divers types de comportements rencontrés au sein des élèves, on en trouve certains qui demandent une petite adaptation :

– des élèves qui essaient d'appliquer tel quel le plan d'exercices retenus par coeur (par exemple l'équation de droite) et mettent un temps à s'adapter à des objectifs concrets. Beaucoup de ceux-ci finissent par découvrir que ces concepts – pour lesquels ils avaient des facilités – servent *aussi* à faire des choses concrètes ! (qui les motivent a posteriori).
– des élèves qui n'osent pas s'impliquer car ils pensent ne pas être capable d'y arriver. Ils essaient souvent de se trouver une place de second passif dans un binôme et il faut pousser un peu pour qu'ils se rendent finalement compte qu'ils sont capables comme les autres, et peuvent mobiliser un certain savoir.

Les lycéens sont dans l'ensemble repartis enchantés, certains ayant continué à utiliser MobiNet durant la pause à mi–séance. Il faut cependant noter que quelques rares élèves n'accrochent pas et trouvent la tâche encore trop scolaire.

5 Évaluation du logiciel

A la suite des deux premières séances de l'atelier MobiNet, qui ont eu lieu en décembre 2002 et avril 2003, nous avons demandé aux lycéens de remplir un questionnaire d'évaluation. Trois classes ont participé : les classes de seconde et première du Lycée Pablo Néruda et la classe de seconde du Lycée du Grésivaudan à Meylan.

Chaque question donnait lieu à une note de 1 à 5 (le plus positif) et pouvait être accompagnée d'un commentaire (qui se sont révélés abondants et pas toujours en correspondance avec l'appréciation numérique). Les figures 5 à 10 montrent les réponses de chaque classe (classe de seconde du Lycée du Grésivaudan en haut à gauche, classe de seconde de Pablo Neruda en bas à gauche, classe de première de Pablo Neruda en haut à droite) ainsi que la somme des réponses (en bas à droite).

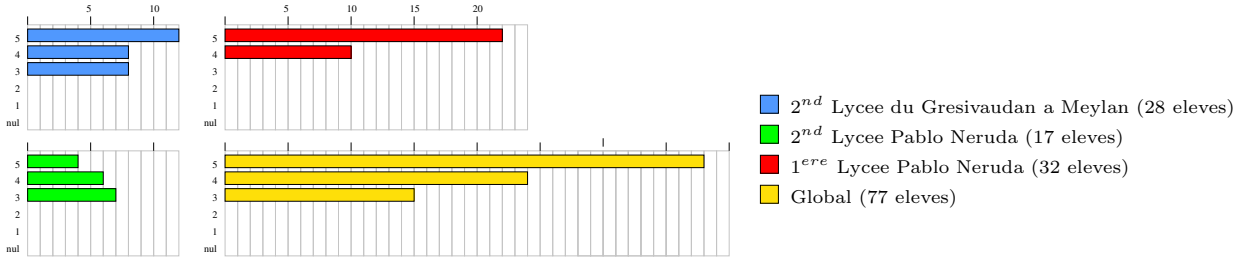


FIG. 5 – Le TP avec MobiNet vous a t–il intéressé ?

Le premier constat est que la séance a intéressé tous les élèves (figure 5). Les commentaires évoquent le coté amusant du TP et l'intérêt de comprendre comment les jeux vidéos sont conçus. La mise en pratique (création du jeu) est également appréciée. Généralement la séance a été perçue comme utile (figure 6). Les raisons évoquées sont principalement : l'initiation à la programmation de jeux vidéos (16 commentaires), l'initiation à l'informatique en général (13

commentaires). Seule une minorité d'élèves ont trouvé la séance ennuyeuse (3 commentaires), ce qu'ils justifient par un manque d'intérêt envers l'informatique.

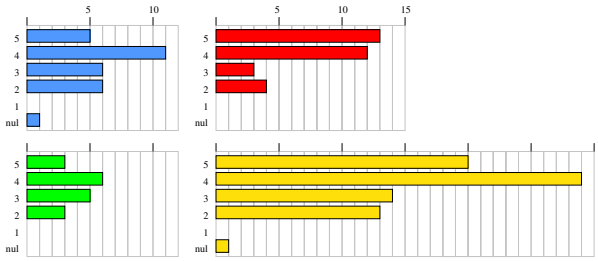


FIG. 6 – Le TP avec MobiNet vous a t–il paru utile ?

Au niveau de la prise en main du logiciel les élèves ont généralement trouvé la manipulation plus facile que ce à quoi ils s'attendaient (figures 7 et 8). Ils ont particulièrement apprécié la documentation et l'interface (19 commentaires favorables, contre 3 défavorables). Certains ont trouvé le premier contact difficile mais notent qu'après un temps d'adaptation la manipulation devient aisée (11 commentaires). Un petit groupe d'élèves (9 commentaires) éprouve des difficultés à manipuler les notions mathématiques requises (variables, équations de droite, trigonométrie). Cette difficulté disparaît généralement assez vite, mais peut chez certains persister durant la séance. Globalement ce résultat est encourageant car nous appréhendions plus de difficultés dues aux notions nouvelles introduites par le logiciel. Mais il montre aussi, ce que nous constatons durant la séance, que beaucoup de notions fondamentales sont en fait très fragiles dans l'esprit des élèves. Le manque d'appropriation de ces notions étant certainement lié au faible lien avec le concret.

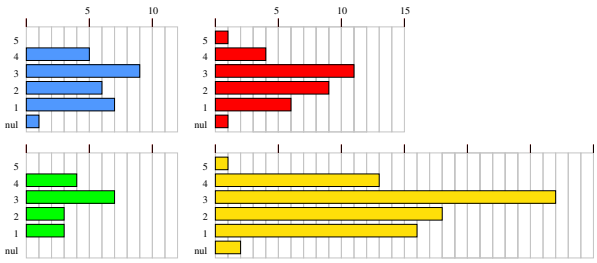


FIG. 7 – Avant de commencer vraiment le TP, pensiez–vous alors que ce serait facile ?

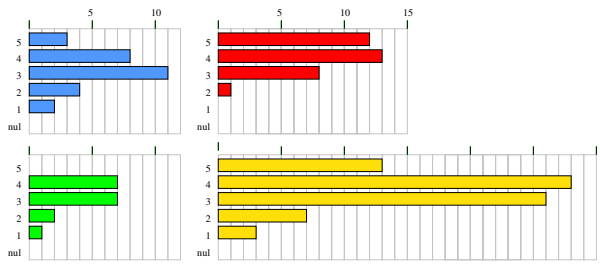


FIG. 8 – Avez vous trouvé MobiNet facile ?

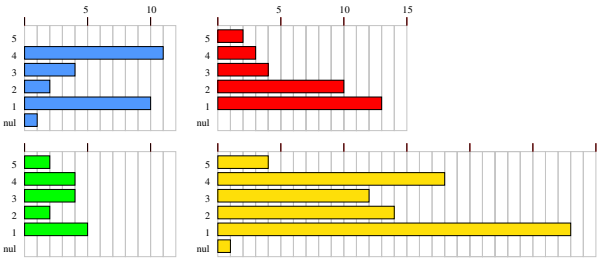


FIG. 9 – Ce TP change t–il votre point de vue sur les maths et la physique ?

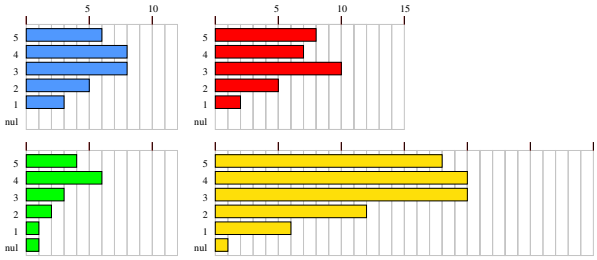


FIG. 10 – Utiliser MobiNet au lycée vous paraît–il utile ?

En ce qui concerne le changement de point de vue sur les mathématiques et la physique, on peut distinguer trois profils. Le premier concerne les élèves qui découvrent que les enseignements qu'ils ont reçus au lycée ont des applications concrètes directes (16 commentaires). Pour ceux–ci l'objectif est pleinement atteint. Le second profil représente l'élève déjà intéressé par les matières scientifiques (9 commentaires) et qui n'a donc pas changé de point de vue (ce qui est somme toute rassurant !). Le dernier profil concerne l'élève qui ne fait pas le lien entre MobiNet et les matières scientifiques ou qui est rebuté par l'aspect mathématique (11 commentaires). Comme le montre le graphique (figure 9) beaucoup d'élèves estiment ne pas avoir changé de point de vue. Néanmoins environ un tiers d'entre eux déclare avoir été positivement influencé. Cette tendance est encore plus marquée dans les classes de seconde – point important étant donné le choix d'orientation que doivent effectuer les élèves à la fin de cette année scolaire.

Nous avions également demandé aux élèves s'ils pensaient que MobiNet pourrait s'avérer utile au lycée. Les réponses se montrent plutôt favorables. Les commentaires mettent en avant l'aspect ludique du logiciel (14 commentaires) et l'utilisation de MobiNet pour appuyer les cours de mathématiques et (11 commentaires). Les élèves proposent également de pouvoir accéder au logiciel en libre service ou dans le cadre de clubs informatique (13 commentaires) mais certains soulignent les problèmes de disponibilité des salles informatiques dans leur lycée (11 commentaires). Nous discutons plus avant les différentes approches envisagées pour le futur des actions pédagogiques avec MobiNet dans la section 6.

L'atelier MobiNet a également été évalué par l'INPG tout comme l'ensemble des ateliers organisés pour la "semaine découverte ingénieur". L'évaluation montre que cet atelier est l'un des plus apprécié par les élèves. A noter que les bilans des "semaines découverte ingénieur" sont très positifs puisque l'ensemble des élèves déclare apprécier cette semaine et un tiers d'entre eux estiment que cela va les aider dans leur orientation (ce qui rejoint notre analyse).

En conclusion, quel que soit leur niveau il semble que la plupart des élèves ne soupçonnaient pas que l'ingénierie, les jeux vidéos, voire la description du monde réel, pouvaient avoir un rapport avec les notions vues en mathématiques et physique ! Pour les 'bons' élèves cela renforce un intérêt qu'ils avaient déjà pour les disciplines ; et les rassure par le lien établi entre leur goût pour des disciplines abstraites et des possibilités professionnelles futures. Pour quelques élèves en difficulté le niveau mathématique minimal requis était encore trop haut, et la motivation pro-

posée trop éloignée de leur propre centre d'intérêt. Pour ceux-là, il faudra encore réfléchir à des approches plus adaptées, en groupes plus petits. Pour beaucoup l'exercice était passionnant et ludique mais beaucoup n'ont pas vu qu'il y avait un lien conceptuel avec leur enseignement des mathématiques et de la physique. Ceci force à s'interroger quant à la conception que se font les élèves de l'enseignement des sciences !

6 Le futur de MobiNet

Tout d'abord MobiNet continue bien sûr de faire partie des "semaines découverte ingénieur" de l'INPG qui ont lieu en décembre et avril de chaque année et mobilisent chaque fois une quinzaine d'encadrants suite à un appel à volontaires. Nous encourageons donc les moniteurs à venir encadrer et à développer des séances de travaux pratiques MobiNet afin de continuer à **faire vivre et évoluer cette expérience**.

Nous souhaitons également **diffuser MobiNet** à un large public : enseignants, élèves ou particuliers, niveau lycée mais aussi peut être dans l'enseignement supérieur (universités, écoles). Le logiciel est gratuit et peut être téléchargé à l'adresse `http ://www-imagis.imag.fr/mobinet`. Il serait intéressant, notamment par l'intermédiaire des CIES et des projets de monitorat, de diffuser MobiNet dans d'autres académies. Nous avons aussi contacté et présenté le logiciel à plusieurs organismes (CCSTI de Grenoble, GTD de mathématiques à l'origine des nouveaux programmes, IREM de Grenoble, et journées GRECO) afin d'étudier les diverses possibilité d'intégration de MobiNet dans les lycées. Nous avons gardé contact avec les professeurs accompagnant les classes qui ont participé à l'atelier MobiNet et évoqué diverses possibilités d'action. Nous visons maintenant à mettre en place un club d'utilisateurs enseignants avec l'aide de l'IREM. Il s'agit en particulier de concevoir des travaux pratiques plus directement adaptés à l'usage scolaire, en collaboration avec les enseignants (et autant que possible conçus par eux).

Il existe donc de nombreuses possibilités d'évolutions pour ce projet (voir annexe C) et nous espérons que les moniteurs pourront apporter un soutient précieux au projet dans le futur !

7 Remerciements

Les premiers remerciements vont bien entendu aux élèves et professeurs qui nous permettent d'améliorer le logiciel et les exercices l'accompagnant, mais également aux encadrants des séances MobiNet. Merci aussi à Patrick Kocelniak pour son aide avec l'équipement informatique de l'ARV. Enfin, nous remercions l'INPG de nous avoir permis de développer ce projet à l'occasion des "semaines découverte ingénieur" et de nous apporter son soutien.

Annexe A

Documentation du logiciel Mobinet

MobiNet - langage des mobiles

http://mobinet.imag.fr

Variables d'état (ou attributs) d'un mobile :

x	l'abscisse de sa position
y	l'ordonnée de sa position
visible	indique s'il est visible
angle	son orientation (en radians)
icone	son icône
largeur	sa largeur
hauteur	sa hauteur
zoom	(pour régler les deux à la fois)
couleur	sa couleur
rouge	la composante rouge de la couleur
vert	la composante verte de la couleur
bleu	la composante bleu de la couleur
gris	la composante grise de la couleur
dx	(libre ; par ex : vx)
dy	(libre ; par ex : vy)
mem0..4	(libre)

Programme

Un programme de mobile est une suite d'*instructions* permettant de modifier les attributs. Ces instructions peuvent être entrées dans diverses zones (voir le *guide de l'interface*), qui seront exécutées selon les circonstances (tout le temps, en cas de collision aux bords ou avec un autre mobile, ou juste au démarrage). Le programme est pris en compte une fois que l'on a cliqué sur "Appliquer". S'il comporte des erreurs, la zone s'affiche en rouge, et un message apparaît en bas d'écran. (Notez que le mobile n'exécutera le programme que si on l'a mis en marche, en cliquant sur "Start"). Voici par exemple 4 instructions qui modifient différents attributs du mobile courant :

```
x : 50
couleur : ROUGE
dy : cos(t)
y : y + dy
```

Remarquons que la dernière instruction, `y : y + dy`, indique que l'abscisse du mobile courant doit être augmentée de la valeur `dy` . Dans la 3ème instruction, `t` représente le temps.

Voici un récapitulatif des instructions que l'on peut utiliser :

Fonctions :

+	-	*	/	carre	racine	log
sin	cos	tan	asn	acs	ang	exp
norme	dist	rnd	srnd	min	max	
abs	ent	frac	sgn	mod	et	ou
=	<	>	<=	>=	<>	

Variables et constantes :

t	dt	PI	clic	cliquee	declic	contact
moi	suiv	prec	souris	lui	camera	chocx
NOIR	BLANC	ROUGE	VERT	BLEU	lampe (,2,3)	chocy
GRIS	CYAN	ORANGE	JAUNE	VIOLET		VIDE

plusproche...	,2,3	_vers(,2,3)	_gauche	_droite	_haut	_bas
touche...		_espace	_gauche	_droite	_haut	_bas

Commandes :

:	stop	start	restart	comme	met_en	si	alors	sinon	finsi	trace :

Exemple :

```
y : 80*cos(t)
mem1 : 80*sin(t + PI/3)
```

```
si y > 0 alors
   couleur : VERT
   x : mem1
sinon
   couleur : ROUGE
   x : 0
finsi
```

Remarquer l'instruction `si` : entre `si` et `alors` se trouve un *test*. Si le test est vérifié, alors les instructions qui suivent jusqu'au `sinon` sont exécutées, sinon ce sont celles entre `sinon` et `finsi` qui le seront. On peut omettre le `sinon` s'il n'y a rien à y mettre.
On peut également tout écrire sur une seule ligne (noter les ' ; ') :

```
si x>0 alors couleur : ROUGE ; sinon couleur : BLEU ; finsi
```

Les principales fonctions :

`abs(`f`)`	calcule la valeur absolue de f
`ent(`f`)`	calcule la partie entière de f
`frac(`f`)`	calcule la partie fractionnaire de f
`sgn(`f`)`	calcule le signe de f
a `mod` b	calcule a modulo b
`racine(`f`)`	calcule la racine carrée de $\|f\|$
`carre(`f`)`	calcule f^2
`sin(`f`)`	calcule le sinus de f (en radians)
`cos(`f`)`	calcule le cosinus de f (en radians)
`tan(`f`)`	calcule la tangente de f (en radians)
`ang(`a,b`)`	calcule l'angle (orientation) du vecteur (a, b)
`dist(`m,n`)`	calcule la distance entre les mobiles m et n
`norme(`a,b`)`	calcule $\sqrt{a^2+b^2}$, norme du vecteur (a, b)
`rnd`	tire un nombre aléatoire entre 0 et 1
`srnd`	tire un nombre aléatoire entre -1 et 1
`min(`a,b`)`	calcule le plus petit nombre entre a et b
`max(`a,b`)`	calcule le plus grand nombre entre a et b

Les autres mobiles

Un programme de mobile peut modifier les attributs de son mobile, mais ne peut pas modifier les attributs d'un autre mobile. En revanche, il peut les lire. Supposons que le mobile courant soit le numéro 1. Nous voulons que le mobile 1 ait la même abscisse que le mobile 2. Nous pouvons écrire ce programme pour le mobile 1 :

```
x : x2
```

- Les numéros des mobiles situés avant et après le mobile courant dans la liste sont donnés par `prec` et `suiv`.
- Le numéro du mobile courant est donné par `moi`.
- Le numéro du mobile le plus proche (à l'écran) du mobile courant est donné par `plusproche` (on peut aussi utiliser l'abréviation `pp`).
- En cas de collision, le numéro du mobile collisionné est donné par `lui`.

Les différents types de mobiles :

m	prec	moi	souris	VIDE	plusproche
m@n	suiv	lui	camera	lampe	(variantes de pp)

La souris

C'est aussi un mobile. On ne peut cependant pas modifier ses attributs. Mais on peut facilement programmer un mobile pour qu'il suive la souris :

```
y : xsouris                ou encore        s : souris
y : ysouris                                 x : xs
                                            y : ys
```

On peut également connaître sa direction et sa vitesse avec `dxsouris` et `dysouris`.
D'autre part, on peut savoir si le bouton de la souris a été cliqué par `si clic alors`, s'il a été relâché par `si declic alors`, s'il est maintenu appuyé par `si cliquee alors`.

La caméra

C'est aussi un mobile ! On peut changer sa position (ex : `xcamera : xsouris`), son zoom (`zoom camera : 2`), son angle (pour tourner la vue)...
En outre, on peut désactiver l'effacement d'écran par `visible camera : 1` pour que les objets laissent une trace visible de leur trajectoire, ce qui permet de dessiner des courbes (raccourcit : `trace : 1`).
Remarque : la caméra est remise à zéro quand on clique sur "reset".

Les lampes

De même, on peut changer la couleur de l'éclairage (ex : `couleur lampe : ROUGE` ou `gris lampe : (1+sin(t))/2`) et sa direction (ex : `x lampe : x1 ; y lampe : y1 ; hauteur lampe : 10`). Initialement la hauteur de la lampe est à l'infini. On dispose en fait de deux lampes supplémentaires `lampe2` et `lampe3`, initialement éteintes (i.e. couleur noire).

Lire les attributs d'un mobile sur une machine distante

Nous voulons par exemple que notre mobile prenne la couleur du mobile 3 de la machine 15.
Ceci s'écrit : `couleur : couleur3@15`

m`@`n désigne le mobile numéro m de la machine (ou "poste") numéro n (un peu comme pour une adresse mail). NB : pour que ce mobile m`@`n soit accessible, il faut que la machine n l'ait *exportée* (en cliquant sur "Exporter"), et que vous *importiez* ses mobiles en cliquant sur son numéro n dans le bandeau "Visualiser". Il est alors également pris en compte par les commandes comme `plusproche`.

Variables locales

On peut utiliser des mots quelconques pour stocker une valeur (variable intermédiaire) :

```
objet : souris          // objet devient un raccourci pour souris
d : dist(moi,objet)     // d contient la distance à la souris
x : x + 3*(x-xobjet)/d // avance de 3 dans la direction de la souris
y : y + 3*(y-yobjet)/d
```

Remarques sur les attributs

Icônes :
Noter que les icônes 0 à 9 représentent les chiffres de 0 à 9, ce qui facilite la construction de compteurs.

Largeur, hauteur :
Attention, il s'agit d'un *facteur de zoom* par rapport à la taille initiale de l'icône. La plupart des icônes ont une taille initiale d'environ 10.

Angle :
Ici aussi, il s'agit de l'angle de rotation par rapport à l'orientation initiale, qui peut être horizontale ou verticale selon les icônes. NB : tous les angles dans MobiNet sont en radians.

Les couleurs :
`couleur` permet juste d'indiquer un nom de couleur. Pour régler ou connaître précisément la teinte en RGB, on dispose des attributs dérivés `rouge` , `vert` , `bleu` (ainsi que `gris`).

Les collisions

On les traite dans la zone “Collision”. On dispose alors de divers variable supplémentaires : `lui` indique le mobile avec qui on est entré en collision. On peut ainsi tester cette variable s'il faut agir différemment selon le mobile. Exemple : `si lui=2`, ou `si lui=2@3`, ou `si couleur lui = BLEU` . À noter qu'il est souvent plus commode de tester l'icône ou la couleur du mobile, qui caractérisent généralement sa catégorie, plutôt que son numéro.

Pour une utilisation avancée, on dispose en outre de la variable `contact` qui indique si c'est la première fois que l'on traite cette collision, et du *vecteur de collision* `chocx,chocy` qui indique sous quel angle les mobiles se sont touchés. Voir par exemple comment le **preset** “choc” en fait usage.

Remarque : il y a toujours un risque que l'on soit à nouveau en état de collision au pas de temps suivant. Tester `contact` permet d'éviter de traiter 2 fois la collision (on risquerait alors de re-rebondir dans la mauvaise direction). Une autre technique, également utilisable pour les collisions au bords, consiste à *décollisionner* avant toute chose, en faisait par exemple `x : x-dx; y : y-dy`. Sachez cependant que les collisions sont une tâche délicate en informatique, et qu'il est difficile d'obtenir un comportement parfait (notamment si on bouge vite).

Interaction entre mobiles

Agir sur un autre mobile :

On ne peut modifier les attributs d'un autre mobile (on peut juste le déplacer avec la commande `met_en(m,x,y)`). Par contre on peut agir sur son état avec commandes `stop m` pour l'arrêter, `start m` pour le démarrer, ou `restart m` pour le remettre en route même s'il était déjà en marche.

Quand ce mobile m redémarre, il commence par exécuter son programme “Start” (si vous en avez entré un dans cette zone). C'est utile par exemple pour remettre en jeu une balle qui est sortie, mais cela permet également d'*envoyer des signaux* à des mobiles : un compteur pourra par exemple être incrémenté à distance par `restart`, en mettant `icone : icone+1` dans sa zone “Start”.

Imiter le comportement d'un autre mobile :

Quand plusieurs mobiles partagent le même comportement, on peut dire, pour chaque champs souhaité, de se ‘reporter’ au programme du champs correspondant du module de référence, à l'aide de la commande `comme m` (`m` étant un mobile tournant sur le même poste).

Poursuite, fuite :

`plusproche` permet de ‘voir’ quel est le mobile le plus proche, ce qui permet par exemple de s'en approcher ou de s'en éloigner. De nombreuses variantes de cette fonction permettent d'affiner ce comportement : `plusproche2` et `3` permettent de ‘voir’ qui arrive en second ou en troisième, `plusproche_gauche` permet de ne regarder que dans le cadran Ouest du champs de vision, etc. `plusproche_vers(vx,vy,ang)` ne regarde que dans la direction (vx, vy) avec un champ de vision d'ouverture ang. (Abréviations : `pp, pp2, pp3, ppg, ppd, pph, ppb, ppv, ppv2, ppv3`). ATTENTION : dans ces conditions, il peut n'y avoir aucun mobile en vue. Il faut donc prendre garde à tester si la commande a bien trouvé un mobile :

```
m : plusproche_vers(dx,dy,Pi/2)
si m <> VIDE alors
    (fuir ou attaquer)
finsi
```

Démarrage de l'application

Si l'on n'utilise pas le réseau (usage individuel, ou usage non collaboratif et sans poste maître en salle de TP), lancer simplement l'application.

Dans les autres cas, lancer simplement l'application sur la machine maître, puis lancer **ensuite** `mobile -client nom_machine_maitre` sur les autres. (Pour une machine distante, le nom est l'adresse internet complète). NB : dans le cas de TP avec des élèves, on aura sans doute intérêt à ajouter (en premier) l'option `-nosave` pour interdire la sauvegarde.

MobiNet - l'interface

Plateforme de programmation de mobiles en réseau.

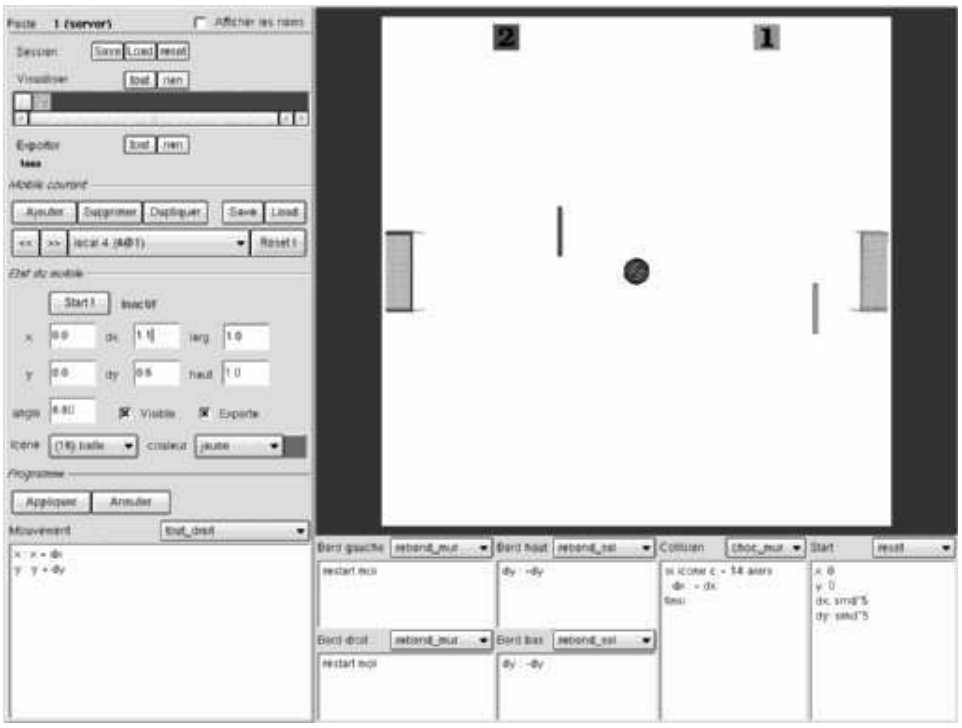
▸ Retour à la page de départ de Mobinet.

*Dans MobiNet, on programme le comportement de différents **mobiles**: ce sont les différents objets visibles (qu'ils bougent ou non!). L'interface permet de régler tous les comportements du mobile courant: aspect, mouvement, règles de collision avec les bords ou les autres mobiles... (on commute ensuite d'un mobile à l'autre).*

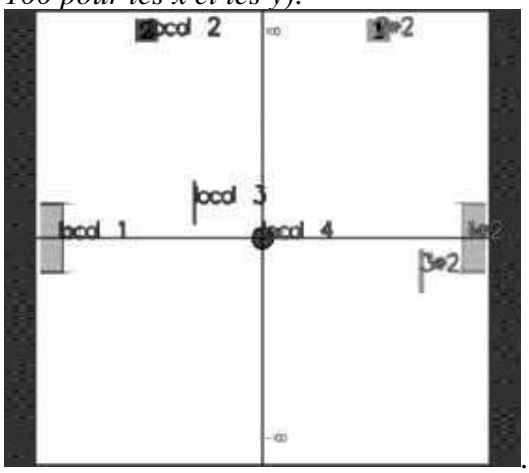
◆ Aspect général de l'écran (cliquer pour agrandir). Les différentes zones sont détaillées un peu plus loin.

Ici on a programmé 7 mobiles: la balle, les raquettes, les buts, les compteurs.

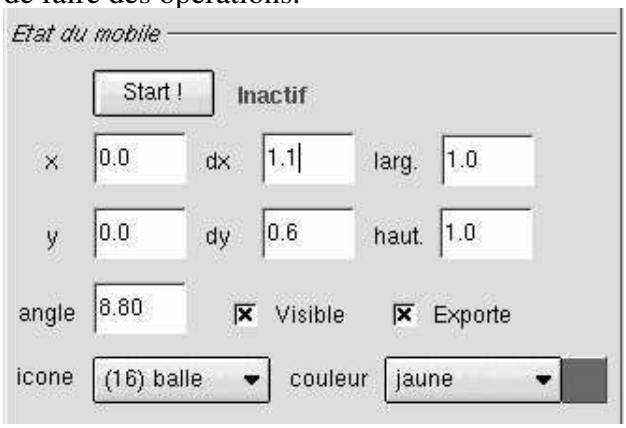
- La balle suit une trajectoire rectiligne, et rebondit sur les bords et les raquettes.
*- Les raquettes suivent la souris verticalement. L'une des raquettes (ainsi que le but et le compteur correspondant) est en fait gérée sur une **autre machine**: deux utilisateurs partagent ici leurs mobiles en réseaux. L'un gère les rouges et la balle, l'autre les bleus.*
- Les buts enregistrent les collisions avec la balle, et déclenchent les compteurs adverses.
- Les compteurs s'incrémentent quand les buts le leur signalent.



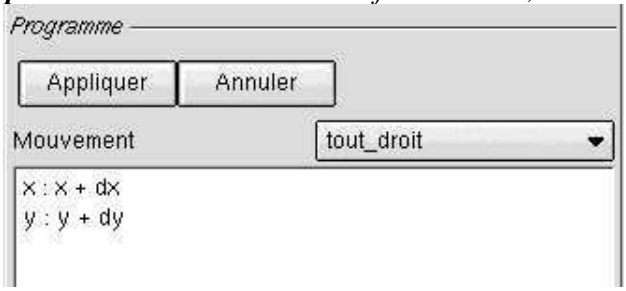
◆ En cliquant sur ‘afficher les noms’, on voit les numéros des mobiles (`local n` *s’il est sur la machine,* `n@m` *s’il est géré par la machine m*), ainsi que le système de coordonnées (*de -100 à 100 pour les x et les y*).



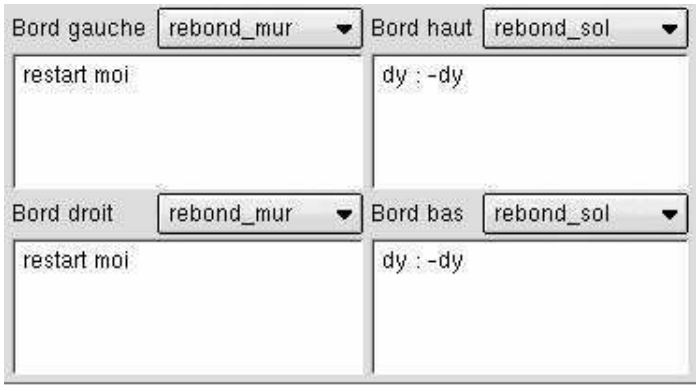
◆ Paramètres représentant le mobile courant (*ici, la balle*): position, dimension, icône, couleur, orientation... (*de plus on va utiliser dx,dy pour indiquer la direction du mouvement.*)
NB: en réalité tout est numérique (*e.g. l’icône de la balle est le numéro 16*), ce qui permettra de faire des opérations.



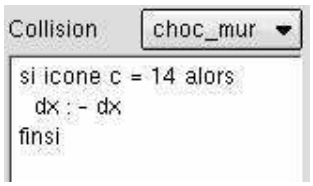
◆ Programmation du mouvement: consiste à décrire les changements à faire entre 2 images successives (env. tous les 25eme de seconde), par "`tel_paramètre: nouvelle_valeur`". Voir le manuel du langage pour le choix des **commandes** utilisables dans les zones de programmation.
On peut faire des petits mouvement successifs (‘dynamique’), décrire une fonction du temps (‘cinématique’), ou de la souris, voire dépendant de la position des autres mobiles (e.g. poursuite), ou toute combinaison programmée selon l’inspiration.
(NB: on dispose d’un choix de comportements prédéfinis où piocher si on le souhaite. Ces ***presets*** *sont stockés dans un fichier texte, un animateur peut donc facilement les modifier.)*



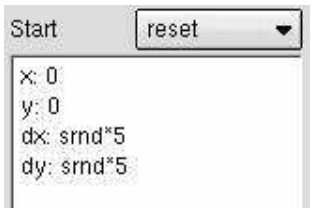
◆ Quoi faire quand on touche un bord.



◆ Quoi faire quand on collisionne un autre mobile.
Si nécessaire, on peut choisir un comportement différent selon l'identité du collisionneur.
(traduction: *si l'icône du collisionneur est une raquette, alors...*)

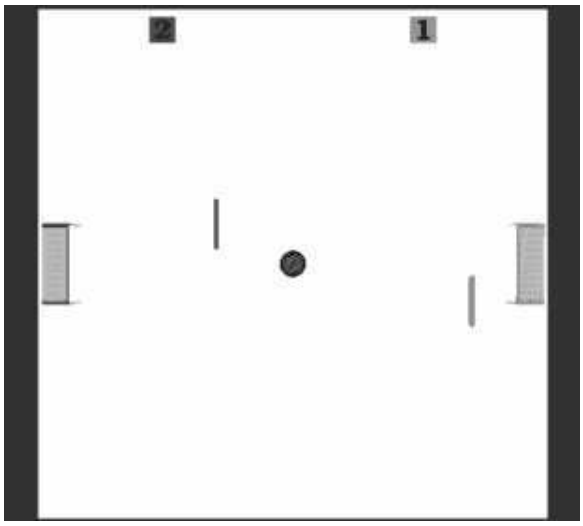


◆ Valeur des paramètres au (re)démarrage.
NB: on peut dire de se redémarrer (e.g. en cas de collision) ou de redémarrer un autre mobile (e.g. pour déclencher un compteur, un tir, remettre la balle en jeu...) avec l'instruction `restart num`. C'est une façon d'envoyer des **messages** entres mobiles.
(`srnd` *donne un nombre au hasard entre -1 et 1.*)



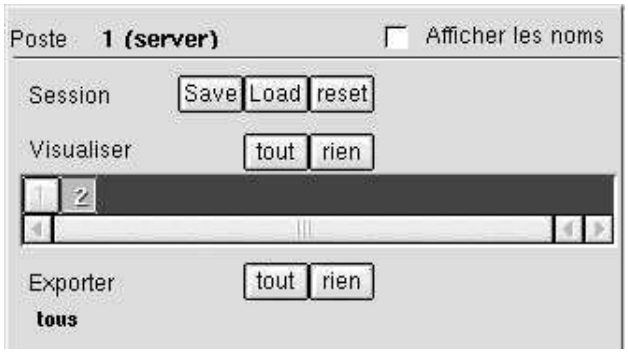
◆ Créer un nouveau mobile, choisir sur lequel on travaille, etc.

◆ Affichage du résultat.



◆ Lire, écrire, interagir avec d'autres machines en réseau...
On choisit d'une part les mobiles que l'on **exporte** (i.e. que l'on rend visible sur le réseau), et d'autre part les machines dont on importe les mobiles.

On dispose donc de plusieurs modes de travail:
- travail indépendant sur chaque poste.
- visionnage de n'importe quel poste depuis le poste maître.
- visionnage superposé de tous les postes (e.g. pour projection murale).
- travail en binômes ou en trinômes (ou autre).
- travail collectif sur N postes (pas forcément sur le réseau local).



◆ Et voilà ! plutôt facile, non ?

Annexe B

Enoncé des travaux pratiques Mobinet pour les “semaines de l’ingénieur” INPG

MOBINET - Fiche de TP

http://mobinet.imag.fr

Introduction

N'hésitez pas à essayer, à recommencer autrement, à tester les exemples :
ça ne gronde pas, ça n'explose pas !
Par contre, si ça fait autre chose que ce que vous attendiez, c'est bien d'essayer de comprendre pourquoi.

B.1 Les variables états

Créez un mobile ; choisissez-lui un icône. Modifiez la valeur de x, y, angle, etc...

B.2 Mouvements simples

Les instructions de la zone *mouvement* sont exécutées à chaque instant,
entre deux affichages d'écran (tous les 25^{eme} de seconde).
Exemples :
`x : x+1` , ou `x : x+10` , ou `x : x-1` , ou `angle : angle + 0.1` ,
ou `x : x+dx` (mettre une valeur dans la variable d'état dx !).

Exercice :
Faites un mobile qui va du point (100,0) au point (-100,100).
Remarques :
- plutôt que re-régler chaque fois à la main la position initiale, faites le une fois pour toute dans la zone *start*.

B.3 Évènements : les bords

Les instructions des zones *bords...* sont exécutées quand on touche les bords, si l'on veut que quelque chose de particulier se passe (rebondir, coller, cycler, s'arrêter...).
Exemple :
dans *bord droit*, mettre `dx : -dx` (rebond), ou `dx : 0` (colle),
ou `x : 0` (cycle) , ou `start` (redémarre le mobile).

Exercice :
Créez un mouvement quelconque (pas juste horizontal). Faites qu'il se passe quelque chose d'intéressant aux bords (pas forcément pareil sur les quatre bords).

B.4 Autres mouvements

Exemple :
`dy : dy-1` (gravité)
`dx : dx+10*srnd; dy : dy+10*srnd` (papillon)
`x : 10*cos(t); y : 10*sin(t)` (cercle autour de (0,0))

B.5 La souris

la position de la souris est `xsouris, ysouris`.
Exemple :
`x : xsouris-10; y : ysouris-10`

Exercices :
- **Faites que le mobile soit décalé de quelques cm à droite de la souris.**
- **Faites bouger le mobile 'en miroir' (i.e. symétrique) de la souris.**
- **Faites tourner le mobile en rond autour de la souris.**

Exemple :
faites *load mobile* et prenez 'aiguille.mobile' :
On peut interpréter la direction de la souris comme une direction à suivre. On peut se servir de ça comme volant, pour diriger le mouvement.

Exercices : (si on a le temps)
- **Faites un mobile piloté à la souris, c'est à dire qui avance dans la direction de l'aiguille.**
- **Faites que le mobile (avion, soucoupe) s'oriente dans la direction où il va.** (indice : regardez comment est fait le mobile de l'aiguille !).

B.6 Plusieurs mobiles

On a actuellement à l'écran deux mobiles à la fois : l'aiguille et votre véhicule.
En fait on peut en mettre autant qu'on veut, et même les faire interagir :
Faites `ajouter`, et entrez le mouvement `x : x1-1`

Exercices :
Faites *reset* pour tout effacer.
- **Faites un mobile simple (qui suit la souris, ou qui avance en rebondissant sur les bords). Faites un second mobile qui tourne autour du premier.**
- **Faites des planètes qui tournent autour du soleil. Ajouter la lune qui tourne autour de la terre.**
- (si on a le temps). **Faites un mobile simple. Faites qu'un second mobile soit attiré par le premier (indice : pensez aux forces en physique, comme s'il y avait un ressort invisible !).**

Remarques :
- Regardez le *preset* 'ressort2' : il simule des ressorts qui ont une longueur à vide.
- On peut utiliser `prec` (signifiant 'mobile précédant') au lieu de 1 : `x : xprec+1`

Exercice :
Reprenez le mobile attiré par le mobile simple (ou refaites le avec 'ressort2'). Faites un serpent en dupliquant dix fois le dernier mobile.

B.7 Évènements : collisions

Comme pour les bords, les instructions de la zone *collision* sont exécutées quand le mobile en touche un autre. Le numéro du mobile touché est dans `c` (ses coordonnées sont donc `xc, yc`).
C'est ce qu'on utilise pour faire qu'un mobile raquette tape dans un mobile balle. (Astuce : faire `stop c` pour arrêter la balle quand elle sort du terrain. Cliquez sur *start* pour redémarrer la balle. Ou alors entrez `restart c` à la place de `stop` pour faire redémarrer directement la balle !).

Exercices :

- **Faites un entraîneur de tennis : la balle part (par exemple du point (100,0)) dans une direction aléatoire. Le mobile raquette, dirigé par la souris, la renvoie. Quand elle sort du terrain on en relance une autre.**
- **Cage de buts : c'est aussi une collision ! Mettez un mobile (immobile !) de buts à droite de l'écran. La balle doit s'arrêter quand elle tombe dedans.**
- **ajoutez un compteur pour le score.**
  Indices : si on fait `icone : 3` le dessin de l'icône est le chiffre 3.
  Si dans la zone *start* on met `icone : icone+1`, ce sera exécuté chaque fois qu'on démarre ou redémarre ce mobile, par exemple avec l'instruction `restart`.

B.8 Réseau

Tout ce qu'on a vu marche aussi en faisant intervenir les mobiles qui sont sur des machines différentes. Mettez vous par groupe de deux machines, faites *exporter tout* pour laisser voir vos mobiles, faites *importer* le numéro de poste de l'autre pour voir ses mobiles.
On voit le mobile 3 du poste 5 sous le nom 3@5 (un peu comme une adresse mail).

Exercices :

- **Pong : un poste fait une raquette et une balle. L'autre fait juste une raquette. La balle rebondi sur les bords haut et bas, et sort à gauche et à droite.**
- **Score : un compteur par camp. Ajouter 1 au compteur gauche quand on sort à droite et réciproquement. Chaque poste s'occupe de son compteur.**
- **Billard à 4 ou 6 trous. Un trou, c'est un mobile rond, noir, immobile, qui fait disparaître les balles qui le collisionnent... Ajoutez le score.**

Exercice final :

Foot : Faites `reset` et chargez le terrain de foot 'foot.session'. Créez un mobile de joueur différent des autres (exportez le pour qu'on le voit). Une moitié de la salle se met en rouge, l'autre en bleu. Faites *importer tout* pour voir les autres. Go !

Annexe C

Recensement des actions potentielles autour de MobiNet

Ce document recense les tâches et actions à effectuer autour de Mobinet. Il liste également les différents soutiens en personnel envisageables pour pérériser le projet.

C.1 Actions autour de Mobinet

Aide aux semaines ingénieur

- préparation des journées (réunions, planning, tests préalables, installations éventuelles),
- animation pédagogique de l'atelier : rôle principal (animateur) ou secondaire (encadrant).
- traitement administratif des vacataires aidant au TP (une douzaine par semaine ingénieur).
- gestion et exploitation du questionnaire aux élèves et aux enseignants

Démonstrations

- à l'IUFM, au CRDP, en divers lycées, etc ... (déjà fait à GTD math, CCSTI, IREM, mais rappel nécessaire).
- salons/expos scientifiques ou pédagogiques appropries
- fête de la science
- atelier type "semaine découverte ingénieur" pour les enseignants ?

Montage de dossiers

- synthèse quantitative d'expériences pour constituer un dossier recevable
- soumission a appels d'offres pour visibilité et financement
- soumissions auprès des institutionnels en vue de fléchage par les 'prescripteurs' :
  - label "intéret et pédagogique" sur site du ministère
  - fléchage CRDP
  - fléchage pour cartable électronique (en Isère : 6 collèges, 8 equipes enseignantes).
  - publication IREM

Diffusion

- maintenance et développement du site web http ://www-imagis.imag.fr/mobinet/
- annonces et contacts pour fléchage par les sites web essentiels (pivots)
- contact des organisations et institutionnels essentiels (en vue de démo/soutient notamment)
- gestion de la mailing-list d'annonces

Animation pédagogique

- suivi auprès de l'IREM et de l'IUFM, pour établir une participation active d'un sous-groupe dans ces deux instances (essentiel pour que les enseignants aient un veritable soutien pédagogique, en lien avec leur corpus)
- suivi auprès des lycees et enseignants partenaires :
  - support et soutien technique
  - éventuellement, aide a l'animation sur place (a la mise en place)
  - relais entre les enseignants, entre les enseignants et les développeurs
  - gestion de la mailing-list pédagogique
  - aide a la constitution d'un corpus de TP pédagogiques, en lien notamment avec le programme scolaire
  - intégration de cette base dans la distribution ou sur le site

Support

- amélioration du manuel d'utilisation
- version anglaise du manuel (et du site !)
- gestion de la mailing-list technique
- recensement des suggestions d'améliorations demandées par les enseignants
- développements légers (debug, robustesse, pilotage par l'enseignant, suggestions triviales à ajouter...).
- éventuellement, développements plus lourds (nouvelles fonctionnalités).

C.2 Actions en lycée

Les modalités d'actions envisagées en lycée (voire en collège ?) :

- séances de TD articulées avec le cours. Format à définir :
  - élèves actifs. En particulier, cas du cartable électronique
  - enseignants actifs (+ manipulation par élèves) : démonstration de concepts (e.g. courbes, géométrie...)
- TP de physique avec matériel virtuel (table a coussin d'air, etc).
- séance de TP sans connexion directe au cours (demi journée ?)
- séances facultatives pour les élèves
- club type "club informatique"
- TPE
- opérations particulières (par exemple "concours de programme" dans le lycée).
- autres ?

C.3 Personnel

Soutiens possibles en personnel pour les tâches évoquées :

- équipe actuelle de développeurs : 1 CR, 1 MdC, 2 doctorants.
- soutien(s) dédié (INPG, aides publiques, ...)
- un moniteur en remplacement l'an prochain
- **une équipe pérenne de moniteurs (comme pour 'Midiscience', 'Zététique' ou 'le Gluon')**
- stages d'élèves (ingénieur ? master en didactique ?)
- autres ?

Certaines de ces ressources nécessitent elles-mêmes un encadrement supplémentaire (stages), ce qui ne résout que partiellement le problème de suivi.