Enseignement ludique de la programmation objets à l'aide des applications de traitement de l'image

Bertrand LE GAL, Lilian BOSSUET et Michael GRAND
bertrand.legal@ipb.fr, lilian.bossuet@ipb.fr,
Institut Polytechnique de Bordeaux (Ecole ENSEIRB-MATMECA)
1, avenue du Dr Albert Schweitzer
BP 99 – 33402 TALENCE Cedex - France

RESUME : *L'informatique industrielle a connu d'importantes avancées ces dernières décennies. La place prise par cette thématique dans la formation des ingénieurs en électronique croît en adéquation avec l'importance croissante des systèmes numériques, embarqués et/ou communicants. L'informatique est maintenant un nouveau pré-requis pour bon nombre de thématiques d'enseignement l'exploitant comme un outil : Traitement du Signal et de l'Image (TSI), automatique, électronique numérique, etc. Aujourd'hui la grande difficulté est de concevoir des projets transdisciplinaires simples en informatique, permettant d'assurer : l'assimilation de l'enseignement théorique et la mise en relief de l'utilisation de l'informatique dans la carrière d'un « électronicien » tout en respectant un faible volume horaire. Dans le cadre d'une réforme des contenus de semestre 4 (niveau master 1), nous avons reconçu un enseignement d'informatique industrielle (programmation objets et langage C++) et nous avons développé un projet de traitement vidéo temps réel sur plateforme Windows, Linux et MacOS. La plateforme pédagogique peut facilement être dérivée pour d'autres domaines pédagogiques comme nous le montrerons dans la suite de l'article. Cet article permet à un enseignant de comprendre les justifications des choix réalisés lors de l'écriture du sujet du projet et du développement de l'application.*

Mots clés : Programmation objets, Traitement du Signal, Architectures des processeurs, Informatique industrielle.

1 INTRODUCTION

L'informatique ainsi que la manière de l'enseigner a grandement évolué durant les dernières décennies avec pour cause l'omniprésence du logiciel dans les produits électroniques. Dans le même temps, l'explosion de la complexité des applications à concevoir a été de pair avec l'évolution des langages informatiques dont le niveau d'abstraction a continuellement augmenté.

Ces évolutions impliquent des modifications nécessaires des enseignements liés à l'informatique industrielle auprès de la population des étudiants dans les domaines connexes tels que l'électronique pour lesquels l'informatique devient omniprésente. En effet, il est actuellement quasiment impossible de concevoir un système électronique ne comportant pas un microprocesseur (ou microcontrôleur), un programme et un OS... Afin de concevoir un système « complet », les ingénieurs en électronique doivent donc être capables d'appréhender tous ces concepts informatiques subjacents.

Toutefois, l'enseignement de l'informatique dans les filières d'électronique doit être adapté aux besoins réels des étudiants dans leur future vie professionnelle. Cette adaptation aux besoins « industriels » implique de fait une augmentation de la transdisciplinarité des enseignements afin de faciliter l'assimilation des liens intrinsèques entre les différents cours suivis.

Prendre en considération l'ensemble de ces points est souvent complexe lors de la refonte d'un enseignement, car :

- L'informatique est souvent enseignée comme une thématique à part entière où les liens avec les do-

maines l'employant ne sont pas toujours clairement établis.

- De tels enseignements nécessitent de la part des enseignants des compétences multiples dans les domaines d'application (automatismes, TSI, systèmes embarqués, etc.) ce qui est assez rare.
- L'investissement temporel est conséquent pour monter un projet transdisciplinaire [1]. De plus, l'enseignant risque de noyer les compétences à transmettre dans sa matière à cause d'un projet trop complexe ou trop ambitieux.

Face à ce constat, lors de la réforme des contenus pédagogiques du 4^{ème} semestre au sein de la filière d'électronique à l'IPB (ENSEIRB-MATMECA), nous avons cherché à développer une maquette logicielle permettant d'illustrer les problématiques de la programmation objets avec comme cadre applicatif le traitement d'image temps réel.

Le principe du projet est exposé dans la Figure 1. À partir d'une source d'information (disque dur, webcam) une application écrite en C++ va acquérir des images en temps réel. Ces images sont affichées sur l'écran de l'ordinateur au fur et à mesure. Le rôle des étudiants va être de concevoir différents filtres vidéo permettant de traiter les images en temps réel et d'ajouter ces fonctionnalités dans l'outil (dont le code source leur est fourni). La plateforme pour faire tourner ce logiciel est un PC pour des raisons pratiques.

L'ensemble des traitements vidéo à mettre en œuvre a été soigneusement choisi afin :

- de mettre en lumière l'intérêt de la programmation objets tout en réduisant la complexité du code à développer par les étudiants.

Figure .1: Principe de la maquette pédagogique du projet de conception objets – C++

- d'illustrer les enseignements de traitement d'images réalisés en parallèle.

Dans cet article nous présentons le sujet du projet ainsi que la base logicielle développée spécifiquement pour cet enseignement. Le plan suivi pour la suite de cet article est le suivant : la section 2 décrit les motivations ayant entraîné le travail présenté. Dans la section 3, nous présentons les objectifs du projet de conception objets ainsi que le matériel pédagogique développé. La section 4 dresse le bilan pédagogique de cet enseignement. Finalement, la section 5 adresse la possibilité de réutiliser l'outil développé dans d'autres enseignements.

2 MOTIVATION DE CE TRAVAIL

Les projets liés à la programmation objets sont généralement orientés vers le génie logiciel. Ils mettent en œuvre des connaissances liées à d'autres enseignements d'informatique théoriques tels que l'algorithmique, la théorie des graphes, etc. Cela rend les matériels pédagogiques développés dans ces approches inadaptés pour les étudiants des cursus connexes (tel l'électronique) pour lesquels l'informatique est avant tout un outil.

C'est pour cette raison que nous avons décidé dans le cadre d'une refonte des enseignements de tronc commun et des enseignements optionnels du 4^{ème} semestre de concevoir un projet d'informatique transdisciplinaire afin d'illustrer au mieux le besoin de telles compétences dans un cursus d'électronicien.

L'enseignement de conception objets dont traite cet article possède un volume horaire restreint : 12 heures de cours magistral, 6 heures de TD devant des machines et 18 heures de projet dont 6 heures sont planifiées, mais non encadrées.

Le matériel pédagogique et le projet que nous avons développés devaient répondre à deux contraintes supplémentaires :

1. Être réutilisable au sein d'autres enseignements d'informatique industrielle,
2. Être partageable avec des enseignants de l'ENSEIRB et d'ailleurs (libre de droits).

3 L'APPLICATION QTPROCESSING

3.1 Programmation objets

Les méthodologies de conception basées sur l'utilisation d'objets ont été développées afin de permettre une réduction de la complexité des applications logicielles. Dans le même temps, elles doivent permettre d'en augmenter leur flexibilité et leur évolutivité tout en assurant la réutilisation partielle ou totale des codes écrits dans d'autres applications.

Tous les enseignants confrontés à ce type d'enseignement savent les challenges que représente cette mission. La principale difficulté est d'abord conceptuelle : la décomposition d'une application en objets en exploitant leurs similitudes & relations va à l'encontre des habitudes de programmation des étudiants.

Une autre difficulté provient du matériel pédagogique employé pour l'enseignement : pour démontrer les avantages de la programmation objets, il est indispensable que l'application possède une complexité minimum difficilement compatible avec les volumes horaires imposés.

Pour remédier à ce dernier point, nous avons décidé de reposer le projet sur 2 concepts :

1. *Fournir un matériel pédagogique aux étudiants* afin de leur éviter tous problèmes situés en dehors des objectifs pédagogiques à atteindre (extraction d'images depuis un fichier vidéo, affichage optimisé des images extraites, création de l'interface graphiques, etc.).
2. *Définir un sujet de projet motivant* pour les étudiants afin de les inciter à y investir leur temps libre. Cela doit permettre de pallier le fait que seules les deux premières et les deux dernières séances de projet sont encadrées.

Le dernier point a été réalisé en concertation avec un collègue enseignant les bases du traitement d'images, afin d'assurer l'illustration de son cours et assurer une cohérence pédagogique. Plus de détails sont donnés dans la section suivante.

3.2 Descriptif du projet étudiant

L'objectif principal du projet est de permettre aux étudiants de mettre en œuvre les connaissances acqui-

Figure 2 : Interface graphique de l'outil QtProcessing. (a) Vidéo d'origine (b) avec traitement vidéo temps réel

ses en programmation objets. Lors du démarrage du projet, les étudiants ont à leur disposition :

- Des codes sources, fournis par l'enseignant, permettant d'extraire des images depuis une vidéo et de les afficher en temps réel à l'écran.
- Un sujet répertoriant l'ensemble des fonctionnalités supplémentaires à intégrer.

Deux captures d'écran de l'application sont présentées dans la figure 2. Cette interface volontairement minimaliste correspond à l'application telle qu'elle est fournie aux étudiants. Elle permet à l'origine du projet de lire des fichiers vidéo et d'afficher les images à l'écran (Figure 2a). En fonction des traitements vidéo choisis l'image affichée évolue (Figure 2b).

L'interface graphique est munie de boutons et de listes sur sa partie gauche. Les boutons servent à sélectionner un fichier vidéo, à démarrer et arrêter la lecture ainsi qu'à passer en mode image par image ou à limiter le rafraichissement à 24 images par secondes. Par défaut l'application traite autant d'images que possible. Les listes sont quant à elles utiles afin de sélectionner les traitements vidéo à appliquer en temps réel sur la vidéo. Au début du projet, elles sont vides car l'écriture des traitements vidéo correspond au travail à réaliser par les étudiants. Enfin, le titre de la fenêtre indique aux étudiants la vitesse de décodage de l'application en image par seconde. Cette information leur permet de savoir si le traitement vidéo sélectionné réduit les per-

formances sous la contrainte temps réel.

Cette application servant de base pour le projet a été écrite en C++ :

- L'interface graphique repose sur la bibliothèque d'éléments graphiques *QT* développée par *Trolltech (Nokia)* [2] et dont l'utilisation croît dans les systèmes embarqués tels que les téléphones portables.
- Le décodage des fichiers vidéo et extraction des images en vue de leur traitement est basé sur la bibliothèque *ffmpeg* [2] utilisée communément dans les lecteurs multimédias, cadres photo, etc.
- La partie liée à l'affichage est quant à elle réalisée et optimisée en *OpenGL* afin d'accroître les performances de l'outil

Le code source fourni aux étudiants représente environ 1000 lignes de code. Les étudiants n'ont nullement besoin de comprendre l'ensemble du matériel pédagogique mis à disposition. Néanmoins, le fait de leur fournir une application déjà écrite leur permet de revêtir deux rôles distincts lors du projet :

1. *concepteurs* des classes de traitement vidéo qu'ils doivent planter
2. *utilisateurs* des objets et des API fournies.

Cela leur permet d'assimiler pleinement les avantages et les contraintes liées à la programmation objets en leur faisant revêtir à tour de rôle les casquettes de concepteurs et d'utilisateurs d'objets.

Fig. 3: Diagramme de classes issu de l'analyse d'une partie des traitements à intégrer lors du projet

Les traitements mathématiques à intégrer ont été soigneusement sélectionnés afin de présenter une complexité d'implantation faible (car le traitement d'image n'est pas l'objet principal du projet) tout en proposant des relations intéressantes entre les filtres d'un point de vue de la conception objets. Voici une liste non exhaustive des traitements vidéo à intégrer :

- *Filtrages passe-haut et bas* à l'aide de convolutions (matrices 1 et 2 dimensions),
- *Masquage* des canaux de couleur,
- *Transformation* des images *couleur* en nuance de gris (2 méthodes),
- *Sur et sous échantillonnage* à l'aide de diverses fonctions d'interpolation (poinçon, linéaire et cubique),
- *Motion blur* : génération d'un flou de bougé à l'aide d'une sommation temporelle pondérée des images,
- Assemblage de différents traitements énoncés.

Le projet se déroule en trois étapes principales :

La première étape à réaliser par les étudiants est de concevoir un filtre et de le mettre en œuvre afin de se familiariser avec l'environnement logiciel fourni. Cela assure la bonne compréhension des mécanismes de l'application et la maîtrise des API fournies.

La seconde étape consiste à analyser les besoins, à concevoir et à intégrer l'ensemble des filtres en utilisant les principes objets et la méthodologie UML vus en cours. L'ensemble des traitements vidéo choisis permet de mettre en exergue l'intérêt des objets : héritage, utilisation, surcharge des méthodes, etc. Un exemple succinct de diagramme de classes issu de l'analyse de quelques-uns des traitements est présenté dans la figure 3. La décomposition fournie n'est pas l'unique solution, des formes plus développées sont possibles : dans le cas de la convolution par des matrices, il est possible de différencier les convolutions (matrices 3x1, 1x3, croix, etc.) ou de factoriser par une classe commune au risque de perdre en temps de traitement (réalisation de toutes les opérations, même celles avec des coefficients nuls).

La dernière étape consiste à imaginer et à mettre en œuvre une modification de l'interface graphique afin de permettre à l'utilisateur d'effectuer de manière séquentielle autant de traitements vidéo qu'il le souhaite. Cela impose aux étudiants de se confronter à la documentation de la bibliothèque graphique QT et de

concevoir une manière générique de pallier au problème (utilisation de la compatibilité des classes vis-à-vis de leur mère et du mot clef *virtual*).

En parallèle à ces différentes étapes, l'enseignant peut en fonction du niveau des élèves leur fait prendre conscience des aspects négatifs de l'utilisation des objets sur les performances de l'application temps réel (mémoire, vitesse). Ces points sont facilement illustrables grâce au faible nombre de lignes de code nécessaire pour ajouter une fonctionnalité dans l'outil. Cela permet simplement de pouvoir comparer différentes variantes d'implantations du même traitement vidéo et d'observer leur impact sur les performances.

Pour s'en convaincre, il suffit d'observer le code source d'une méthode de la classe permettant de réaliser un filtrage sur le canal rouge proposé en figure 4. Cette fonctionnalité présentée dans sa version intégralement objet a été réécrite dans la figure 5. Cette nouvelle méthode d'une autre classe tire parti de l'utilisation de pointeurs augmentant fortement les performances (80% de temps de calcul en moins).

Cette technique est proposée aux étudiants afin de leur faire prendre conscience qu'il y a une différence entre les bases théoriques (tout objet) et l'utilisation qu'ils en feront dans la conception de systèmes embarqués contraints. Dans ce cas, l'expérience et les contraintes leur feront réaliser des compromis dans l'écriture des méthodes de classe en y insérant du code « C » optimisé.

Pour les plus « rapides » qui terminent en autonomie l'ensemble de ces points avant la moitié des séances, d'autres transformations plus complexes peuvent être proposées : optimisation de la luminosité, affichage d'un histogramme incrusté dans la vidéo en temps réel, etc.

Le projet tel qu'il a été conçu offre des avantages pédagogiques indéniables. Tout d'abord, il permet aux étudiants d'appréhender une nouvelle manière de concevoir les outils logiciels dont ils ont besoin. Il illustre aussi l'enseignement de traitement vidéo purement théorique. Ensuite cela leur permet de réviser les bases acquises en langage C sur les pointeurs en percevant de manière concrète leurs intérêts.

3.3 Performances obtenues

L'application est capable de fonctionner sur un PC d'entrée de gamme (Pentium 4, 2Ghz, 512Mo) sans

```
bool CopyTranfo::Execute(FastImage *in, FastImage *out){
    for(int y=0; y<in->height(); y++){
        for(int x=0; x<in->width(); x++){
            out->Red(y, x, in->Red(y,x));
            out->Green(y, x, in->Green(y,x));
            out->Blue(y, x, in->Blue(y,x));
        }
    }
    return true;
}
```

Figure 4 : Code source purement objet permettant la recopie de l'image d'entrée en sortie (écran)

```
bool CopyTranfo::Execute(FastImage *in, FastImage *out){
    int size = in->height() * in->width();
    int* p = (int*) in->getPointer();
    int* q = (int*) out->getPointer();
    while( size -- ){
        *(p++) = *(q++);
    }
    return true;
}
```

Figure 5 : Code source optimisé à l'aide de pointeurs et permettant la recopie de l'image d'entrée en sortie

carte graphique particulière. Toutefois pour des raisons de performance une carte vidéo capable d'exécuter des traitements *OpenGL* peut être un plus (réduction de l'utilisation du processeur dans la phase d'affichage des images).

Les étudiants sont d'abord orientés vers une vidéo de taille « réduite ». Sans appliquer de filtrage sur la vidéo, l'application atteint des débits de 130 à 200fps environ. Cela permet aux étudiants de cascader plusieurs filtres successifs tout en respectant la contrainte temps réel. Par la suite, ces derniers sont amenés à tester les performances de leur code sur une vidéo haute définition HD (1280x720 pixels). Cette dernière réduit le débit de l'application à 60 images par seconde. Ce test, grandeur nature leur permet de s'apercevoir qu'un simple traitement sur un canal (filtrage le plus simple) requiert trop de temps pour respecter le temps réel.

Toutefois, cela permet d'aborder avec les étudiants l'utilisation avancée des pointeurs et des traitements en virgule fixe permettant facilement de respecter à nouveau la contrainte.

3.4 Portabilité de l'application

Le projet de conception informatique a été conçu de telle manière que les étudiants réalisent une partie du développement lors des séances de projet avec l'enseignant alors que le reste est à réaliser en autonomie. Les moyens informatiques dont disposent les étudiants en dehors de la salle équipée de MacOS à l'école sont hétérogènes, mélangeant divers systèmes d'exploitation et différentes versions de ces derniers.

```
bool LuminanceFilter::Execute(FastImage *in, FastImage *out){
    for(int y=0; y<in->height(); y++){
        for(int x=0; x<in->width(); x++){
            r = in->Red (y,x); g = in->Green(y,x);
            b = in->Blue (y,x);
            int v = floor( 0.299 * (float) r + 0.587 * (float) g
                          + 0.114 * (float) b );
            out->RGB(y, x, v, v, v);
        }
    }
    return true;
}
```

Figure 6 : Code source du calcul de la luminance (conversion en gris) à l'aide de nombres flottants.

```
#define C1 ((int)(0.299 * (1 << 16)))
#define C2 ((int)(0.587 * (1 << 16)))
#define C3 ((int)(0.114 * (1 << 16)))

bool LuminanceFilter::Execute(FastImage *in, FastImage *out){
    int size = in->height() * in->width();
    int* p = (int*)in->image;
    int* q = (int*)out->image;
    while( size -- ){
        int g = ( C1 * ( (*p & 0xFF0000) >>16 )
                + C2 * ( (*p & 0x00FF00) >>8 )
                + C3 * ( (*p & 0x0000FF) ) ) >> 16;
        (*q++) = (g << 16) | (g << 8) | g;
        p += 1;
    }
    return true;
}
```

Figure 7 : Code source réalisant le calcul de la luminance après transformation des calculs en virgule fixe.

Afin de résoudre ce problème, il a été décidé d'utiliser uniquement des composants open-source disponibles sur les environnements Windows, Linux et MacOS.

Les codes sources de l'application dans sa version étudiante (sans les classes de traitement vidéo) sont disponibles gratuitement sur internet¹.

La notion de performance souvent absente des enseignements d'informatique ou traitée uniquement du point de vue de la complexité algorithmique est ainsi abordée ludiquement. Ce dernier point est à notre avis important lorsque l'on forme des électroniciens.

4 BILAN PEDAGOGIQUE

Le projet et l'application présentés ici ont été utilisés avec succès l'an dernier au sein de la filière électronique de l'ENSEIRB-MATMECA. Un enseignement sur les architectures avancées des processeurs utilise la plateforme pédagogique pour illustrer le cours. Son utilisation devrait être étendue dans des cours de spécialité en option de 3ème année en TSI et en filière Systèmes électroniques embarqués.

La portabilité de l'application sur l'ensemble des OS communément utilisés a permis aux étudiants cette année de travailler en autonomie sans avoir à recourir à des schémas d'installation complexes sur leurs propres machines. La conséquence directe de cette facilité d'installation a été l'implication des étudiants dans le projet en dehors des heures de cours.

Finalement, un point qui n'avait pas été prévu, fut que les étudiants se sont mis en compétition afin de réaliser l'ensemble du projet rapidement tout en essayant d'améliorer la qualité du code vis à vis des autres groupes (interface graphique, vitesse des traitements vidéo, etc.). Cette « compétition » a été bénéfique pour l'avancement du projet pendant les séances en autonomie.

5 EXTENSIONS POSSIBLES

L'application *QtProcessing* développée initialement pour le projet d'objets a été conçue de manière concertée avec d'autres enseignants afin de permettre son exploitation dans des cours connexes. L'application permet par exemple :

- Évaluation de l'impact de l'écriture du code C et optimisations associées pour les applications TSI : (1) Gestion des accès par pointeurs (2) Codage des données en virgule fixe.
- Gestion des caractéristiques architecturales des processeurs modernes (cas des instructions SIMD [4] dans le traitement du signal),
- Répartition des tâches dans les systèmes multiprocesseurs [2, 5].

En effet, le fait de pouvoir intégrer facilement de nouveaux traitements (classes) permet aisément de

¹ http://uuu.enseirb.fr/~legal/wp_ens/

```
bool GreenFilter::Execute(FastImage *in, FastImage *out){
    for(int y=0; y<in->height(); y++){
        for(int x=0; x<in->width(); x++){
            out->RGB(y, x, in->Red(), 0, 0);
        }
    }
    return true;
}
```

Figure 8 : Filtrage du canal rouge (version « objets »)

```
bool RedFilter::Execute(FastImage *in, FastImage *out){
    int size = in->height() * in->width();
    unsigned char* p = (unsigned char*)in->image;
    unsigned char* q = (unsigned char*)out->image;
    while( size -- ){
        (*q++) = (*p);          (*q++) = 0;
        (*q++) = 0;             (*q++) = 0;
        p += 4;
    }
    return true;
}
```

Figure 9 : Filtrage du canal rouge (pointeur)

```
bool RedFilter::Execute(FastImage *in, FastImage *out){
    int size = (in->height() * in->width() * 4) / 16;
    unsigned char table[16] = {0,0,255,0,0,0,255, ...};
    __m128i* p = (__m128i*) in->image;
    __m128i* q = (__m128i*) out->image;
    __m128i mask = (__m128i*)table;
    while( size -- )
        (*q++) = (*p++) & mask;
    return true;
}
```

Figure 10 : Filtrage du canal rouge (SIMD)

```
bool GreenFilter::Execute(FastImage *in, FastImage *out){
    #pragma omp sections
    {
        #pragma omp section // THREAD ONE
        {
            for(int y=0; y<in->height()/2; y++)
                for(int x=0; x<in->width(); x++)
                    out->RGB(y, x, in->Red(y,x), 0, 0);
        }
        #pragma omp section // THREAD TWO
        {
            for(int y=in->height()/2+1; y<in->height(); y++)
                for(int x=0; x<in->width(); x++)
                    out->RGB(y, x, in->Red(y,x), 0, 0);
        }
    }
    return true;
}
```

Figure 11 : Filtrage du canal rouge (multiprocesseur)

faire évaluer aux étudiants des concepts souvent abstraits à leurs yeux comme les nombres en virgule fixe. Cela peut être illustré simplement à l'aide du travail déjà effectué autour de la conversion en nuance de gris demandée dans le cadre du projet. Le code de la méthode de la classe réalisant cette transformation (nombres flottants) est fourni dans la figure 6. La version réalisant le même traitement en virgule fixe est fournie en figure 7.

L'ensemble des autres points mentionnés est tout aussi aisément illustrable à l'aide du matériel pédagogique développé. Les codes sources des méthodes employant de telles techniques au sein de l'outil sont

fournis dans les figures 8, 9, 10, 11. Ces codes sources sont donnés à titre informatif afin de montrer la facilité du déploiement de telles techniques dans l'outil. Les gains apportés par ces derniers en terme de vitesse de traitement atteignent jusqu'à 80% du temps de calcul.

L'évaluation des gains en performance obtenus à l'aide de ces techniques d'optimisation avancées se fait par observation du nombre d'images traitées par seconde. L'outil assure donc un rôle de démonstrateur des aspects théoriques cités en nécessitant un minimum d'efforts de la part des étudiants et des enseignants (la plateforme logicielle est déjà développée). Une version complète de l'outil comprenant des exemples d'optimisations est disponible² sur demande.

D'autres utilisations sont aussi possibles avec cette plateforme. On peut penser à utiliser l'outil pour illustrer par exemple :

- La mise en œuvre de traitements à l'aide des fonctionnalités *OpenGL*,
- La gestion de l'externalisation des traitements vidéo sur *GPUs*,
- Le lien avec des systèmes de traitement externes (*FPGA*, *DSP*),

6 CONCLUSION

Le projet d'informatique « dédié » à des cursus d'électroniciens que nous venons de présenter ainsi que le matériel pédagogique associé a été conçu spécialement afin d'améliorer la compréhension de l'intérêt de ces matières pour nos ingénieurs. La pluridisciplinarité voulue dans ce projet permet aux étudiants d'acquérir et de mettre en pratique les connaissances acquises, mais aussi de percevoir les liens existants entre les différents enseignements reçus dans des thématiques diverses. Cela renforce leur prise de conscience des nécessités de posséder des bases pluridisciplinaires solides afin d'évoluer dans les domaines de l'électronique numérique.

La plateforme logicielle développée pour ce projet et disponible gratuitement sur internet pour tout enseignant souhaitant faire évoluer ou mettre en œuvre ce projet. Elle peut aussi être adaptée grâce à sa flexibilité dans d'autres enseignements connexes.

Bibliographie

1. J. Sérot, J. Laffont and M. James, "De l'utilité des jeux vidéo pour l'enseignement de l'informatique temps-réel", *Journal de l'Enseignement des Sciences et Technologies de l'Information et des Systèmes (J3EA)*, Vol. 8, 2009.
2. <http://qt.nokia.com/products/>
3. <http://ffmpeg.org/>
4. Intel. AP-809 Real and Complex FIR Filter Using Streaming SIMD Extensions, http://cache-www.intel.com/cd/00/00/01/76/17650_rc_fir.pdf
5. <http://openmp.org/wp/>

² bertrand.legal@ims-bordeaux.fr