Prep'ISIMA

Rapport de Génie Logiciel

IDE : Integrated Development Environment

Auteurs :
Benoît Garçon
Loïck Chiron
Marie Cassaing

Enseignant :
Jonathan
Passerat-Palmbach

3 mai 2013

Sommaire

Introduction

Il existe aujourd'hui une multitude de façons de produire du code et par extension un programme. Elles sont plus ou moins conseillées selon le support, le langage de programmation et surtout l'utilisateur. De façon générale, tout ce qui nous est proposé reflète une qualité certaine et un avantage non négligeable pour le programmeur si toutefois les bonnes pratiques de programmation sont respectées.

C'est donc pour facilité la vie des programmeurs et surtout pour pallier leur oisiveté qu'ont été créés les Environnements de Développement Intégré (EDI) ou encore integrated development environment (IDE) en anglais. En effet, au début de l'ère informatique, le développement ne nécessitait pas autant de fichiers et autres car l'on utilisait des cartes perforées. C'est donc avec l'apparition de la programmation générique actuelle que sont apparus les EDI. Le premier étant Dartmouth BASIC en 1964.

Un EDI est donc un programme regroupant de multiples sous programmes, facilitant la production et la mise en œuvre de code. Mais alors, comment un IDE peut-il assister un programmeur ?

Nous verrons donc qu'il existe tout une liste de fonctionnalités qui font d'un IDE une solution de développement intéressante puis nous concentrerons notre attention sur l'exemple de CodeBlocks.

1 IDE : un éventail de fonctionnalités en un programme

1.1 Les fonctionnalités basiques

Un environnement de développement comporte tous les outils nécessaires pour analyser, écrire et debugger votre programme. Il comporte une jolie interface graphique permettant une utilisation simple de tous les outils qu'il met à votre disposition. On retrouvera donc dans tous les IDE, des éléments incontournables tels qu'un éditeur de texte incorporé comportant une syntaxe typique. C'est-à-dire une mise en valeur de votre code par l'intermédiaire d'une indentation automatique, de mise en couleur des mots clés et d'une complétion automatique de ces mots clés. Il y aura également dans la plupart des cas un débuggeur et un compilateur de manière à pouvoir rédiger, corriger et tester votre programme avec un seul logiciel. De manière générale, le processus de débogage commence pendant la génération. La génération de votre programme aide à détecter les erreurs de compilation et vous affiche les numéros de lignes correspondants. Les erreurs les plus fréquentes étant dues à une syntaxe incorrecte, à des mots clés mal orthographiés ou à une incompatibilité de type. Enfin, un IDE est également très souvent muni d'un système de gestion de projet très pratique lorsque l'on va réaliser un programme de manière collaboratif au sein d'une entreprise par exemple. Cela regroupe donc l'interconnexion des données, dossiers et fichiers nécessaires à la réalisation du projet.

1.2 Les fonctionnalités avancées

Certains EDI plus perfectionnés vont être dotés d'autres fonctionnalités : on trouvera par exemple un outil de création d'interface graphique. Cela permet, comme son nom l'indique, d'aider le programmeur lors de la réalisation de l'interface graphique de son programme. Un EDI peut également comporter un moteur de recherche intégré tenant compte du langage de programmation que vous utilisez, cela vous permet d'effectuer (de manière rapide et efficace) une recherche sur une fonction ou une syntaxe qui pourrait poser problème. Certains environnements sont munis d'un générateur de documentation ou intègre un générateur existant comme Javadoc, Doxygen ou bien Scaladoc. Ceci est très pratique lorsque vous réaliser un programme avec d'autres personnes, car avec la documentation, chacun peut comprendre la signification de vos fonctions et autres boucles qui peuvent paraître étranges à première vue. Enfin, on trouvera un gestionnaire de version ou là aussi l'intégration d'un gestionnaire de version comme Subversion, CVS ou Git permettant de conserver toutes les étapes de la réalisation de votre programme en utilisant des versions antérieures à votre fichier actuel.

2 CodeBlocks : un exemple d'application

2.1 Un premier aperçu

Nous avons choisis ici de vous présenter un IDE particulier puisqu'il est connu par la majorité des membres du groupe : il s'agit Code : :Blocks qui fait parti des IDE les plus répandus pour les langages C et C++ (aussi utile pour le Fortran et le D). Voici donc un outil privilégié pour les développeurs en herbe et peut-être même les plus confirmés. Les raisons d'un tel succès sont assez simple :

- libre et " open source " : c'est un logiciel gratuit donc accessible à tous mais aussi perfectible par tous
- multiplateforme : que l'on développe sous Linux, Mac OS X ou même Windows il existe une version disponible
- complet : Code : :Blocks offre une multitude de fonctionnalités pratiques que l'on peut attendre d'un bon IDE, mais s'il manque un outil il y aura certainement un plug-in pour pallier ce défaut.

L'aventure Code : :Blocks commença en 2005, il a été développé en C++ grâce à la bibliothèque wxWidgets, ce qui explique pourquoi il est si simple d'améliorer cet IDE. Code : :Blocks est un logiciel assez intuitif et simple d'utilisation (mais il reste toutefois très complet). Bien qu'utilisable en ligne de commande, il arbore aussi une interface graphique très classique. On retrouve en haut de la fenêtre les différents menus et boutons donnant accès à toutes les fonctionnalités. Le reste est divisé en deux zones :

- la zone centrale étant composé de l'indispensable éditeur de texte pour taper le code associé à un explorateur de fichiers
- la console de sortie au bas de l'écran regroupe les différents résultats des opérations effectuées et les classe par catégories : recherche, debugger, compilateur ...

[Voir la Figure 1]

Nous allons maintenant voir les différentes étapes pour concevoir un programme sous Code : :Blocks.

2.2 La création d'un projet

Code : :Blocks est aussi un bon gestionnaire de projets, il va pouvoir créer les bases de votre projet en quelques clics seulement et vous proposera un code initial prêt à la compilation. Vous avez la possibilité de créer votre propre modèle de projet mais si l'envie vous manque vous pourrez tout à fait choisir un modèle parmi les nombreux proposés allant de la simple application console jusqu'au projet QT, en passant par le SDL ou encore Matlab Project. Ensuite, il vous demandera de donner un nom au projet et une localisation. Après une vérification de l'emplacement des librairies, vous aurez le choix entre pas moins de 20 compilateur, GCC étant l'option par défaut. Enfin il vous sera demandé si le debugger, par défaut GDB, doit être intégré au projet mais nous y reviendront plus tard.

[Voir la Figure 2]

Avant de commencer à programmer (et même après) vous avez la possibilité d'activer l'autoversioning et même de commencer une liste de " TODO " afin de partager le travail avec vos éventuels collaborateurs. Voilà vous êtes maintenant prêt à saisir votre code.

[Voir la Figure 3]

2.3 La programmation

Vous entrez alors dans une phase d'écriture longue et laborieuse, heureusement Code : :Blocks est là pour vous aider. Concentrons nous d'abord sur l'éditeur de texte. De prime abord on constate qu'il possède bien la " coloration syntaxique " qui permet de mettre en couleur les mots-clés de votre langage de programmation ainsi que les éléments de ponctuation, utile pour éviter les fautes de frappe et la relecture. Le second élément flagrant est la " volonté " de Code : :Blocks à vous faire coder proprement : en effet il possède un système d'indentation automatique qui formatera votre code lisiblement. Ces deux premiers éléments sont tous deux paramétrables dans les menus déroulants. Ensuite l'éditeur possède un système de complètement : après la saisie des trois premières lettres d'un mot, un menu déroulant apparaîtra dans lequel se trouveront toutes les possibilités de saisies, libre à vous de choisir la bonne et de taper " Entrée ". Ceci fonctionne aussi pour la ponctuation et la création de blocs : lors de la saisie d'une parenthèse, crochet, accolade ou guillemet ouvrant son homologue fermant est aussi inséré, évitant bon nombre d'oublis. Le dernier aspect notable de cet éditeur est la possibilité de replier des blocs de code telle une arborescence de fichiers.

[Voir la Figure 4]

Ensuite Code : :Blocks vous offre une vue globale de tous les éléments de votre projet. Il permet d'ajouter, de supprimer et de parcourir aisément tous les fichiers au travers de l'explorateur. Un simple clic droit sur un identifiant permet l'apparition d'un riche menu contextuel avec, entre autre, la possibilité de rechercher la déclaration, l'implémentation, les occurrences, les appelés ou encore les appelant d'une fonction ou d'une simple variable. Le menu déroulant " Search " contient des outils utiles de recherche mais aussi un outil " Replace in files... " très utile pour remplacer toutes les occurrences d'une chaîne de caractères par une autre. Côté programmation objet, Code : :Blocks n'est pas en reste puisqu'il possède un outil de création de classe très complet (" File , New , Class... "). Enfin on peut remarquer la présence du plug-in DoxyBlocks qui permet une génération de documentation simplifiée.

[Voir la Figure 5]

Avec tout la programmation est simplifiée il est maintenant temps de terminer le projet en le compilant ... Enfin s'il n'y a pas d'erreurs.

2.4 La compilation

Côté compilation, les options sont vastes que ce soit au niveau du choix du compilateur ou encore de celui des options, tout est possible. Vous aurez la possibilité d'ajouter une option manquante le cas échéant. Vous aurez aussi la possibilité de personnaliser votre propre Makefile. En tout cas la compilation de votre projet sera très instinctif, elle ce fait effectivement grâce, entre autre, aux boutons que l'on remarque dès la première ouverture de Code : :Blocks. Le " rouage " permet de compiler, la " flèche verte " d'exécuter, la somme des deux de faire les deux et les flèches bleues de reconstruire tout le projet. Le résultat de la compilation apparaît alors dans la console de sortie. Si tout est correct vous obtiendrez un message du genre dans " Build log " :

" Process terminated with status 0 (0 minutes, 12 seconds)
0 errors, 0 warnings (0 minutes, 12 seconds) "

[Voir la Figure 6]

Sinon il faudra recourir au debugger, qui ne sera autre que GDB. Pour ceci il suffit de cliquer sur la " flèche rouge " pour lancer le debuggage. À la rencontre d'une erreur, la ligne concernée vous est indiquée, vous pouvez demander la pile d'appel, l'état des variables, etc. Vous avez aussi la possibilité de placer des breakpoints par un simple clique dans la marge. Les icônes à côté de la " flèche rouge " permettent l'exécution ligne par ligne, bloc par bloc, jusqu'à la ligne courante, etc. Bref, toutes les fonctionnalités d'un debugger sont présentes.

[Voir la Figure 7]

Conclusion

Un environnement de développement intégré est donc un outil non pas essentiel, mais naturel à utiliser. Bien plus pratique qu'un simple Emacs, gedit ou Vi, il permet de gagner du temps. Cette rapidité est acquise grâce à de nombreux raccourcis, aux nombreuses erreurs et à leur résolution interminable évitées. On apprécie la qualité de confort offerte par les IDE devant la primitivité du développement universitaire sous un éditeur de texte simple. La saisie du code est beaucoup plus simple car le programmeur est foncièrement plus assisté et cela est une aide non négligeable pour un programmeur débutant. La clarté d'un code lu sous un IDE est aussi un plus pour son étude, ainsi quand sous Code : :Blocks on tombe sur une fonction inconnue un simple clic droit puis l'utilisation de " Find implementation of : ... " est un luxe qui permet de comprendre plus vite un code et donc d'optimiser sa vitesse de travail.

Concernant plus précisément Code : :Blocks, son utilisation nous a été très bénéfique durant ce semestre puisque centré sur le C. On peut noter toutefois que bien que la puissance de cet outil soit immense, sa spécificité limite son utilisation aux langages C/C++, D et Fortran. Nous n'avons pas pu ici détailler la foultitude de plug-in présent sur Code : :Blocks mais il faut savoir qu'il en existe suffisamment pour que Code : :Blocks puissent humblement concurrencer ses paires commerciaux. Il est certain que nous utiliserons tous un IDE dès lors que nous en aurons l'occasion.

Bibliographie

[1] Wikinautes, *Integrated development environment*. Avril, 2013, $http://en.wikipedia.org/wiki/Integrated_development_environment$.

[2] Wikinautes, *Code : :Blocks*. Avril, 2013, http ://fr.wikipedia.org/wiki/Code : :Blocks.

[3] Wikinautes, *Test de l'EDI C/C++ Code : :Blocks*. Septembre, 2005, $http://loulou.developpez.com/tutoriels/cpp/codeblocks/$.

[4] E. Luneville, *Tutoriel Code : :Blocks*. 2006, $http://www.ensta-paristech.fr/\ lunevill/sim_numerique/cours/intro_codeblocks.pdf$.

[5] Nanoc et Mateo21, *Code : :Blocks (Windows, Mac OS, Linux)*. Avril, 2013, $http://www.siteduzero.com/informatique/tutoriels/programmez-avec-le-langage-c/code-blocks-windows-mac-os-linux-1$.

[6] Anders F. Bjorklund and al., *Code : :Blocks : Manuel Utilisateur*. Mai, 2010, Code : :Blocks.

Table des figures

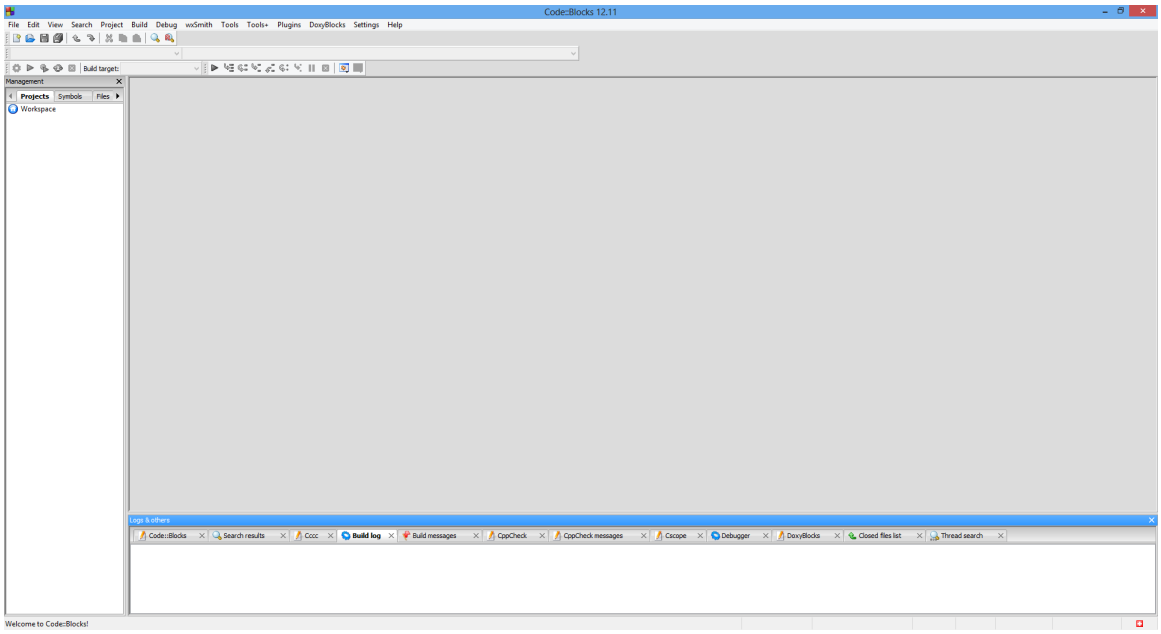

FIGURE 1 – Code : :Blocks

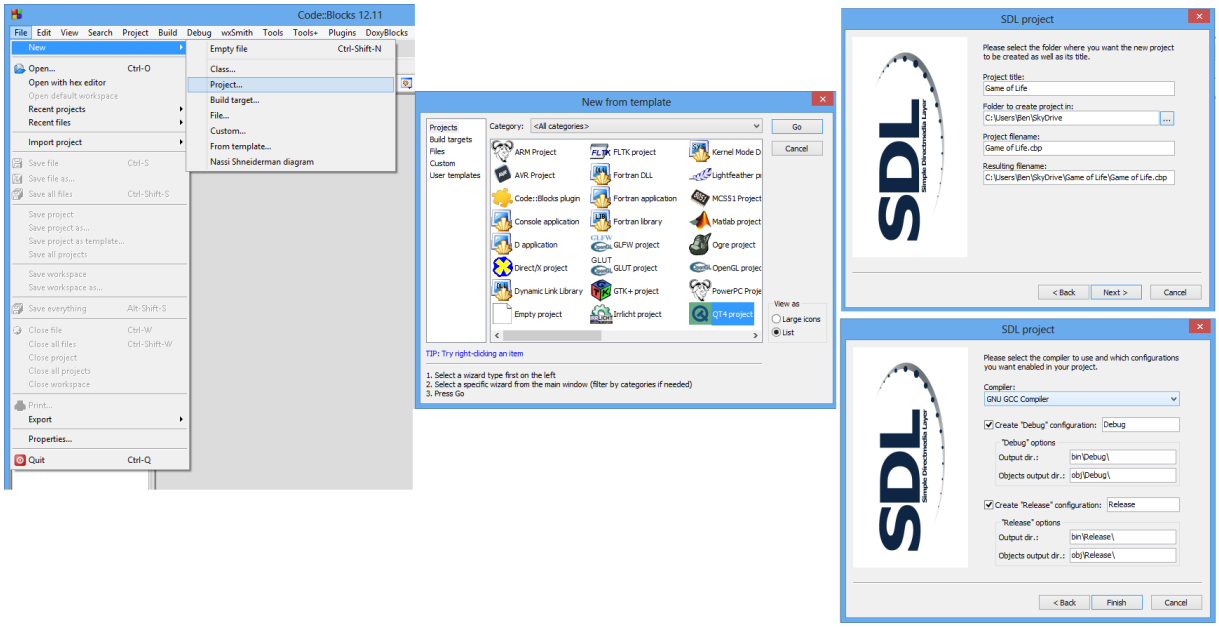

FIGURE 2 – Configuration du nouveau projet

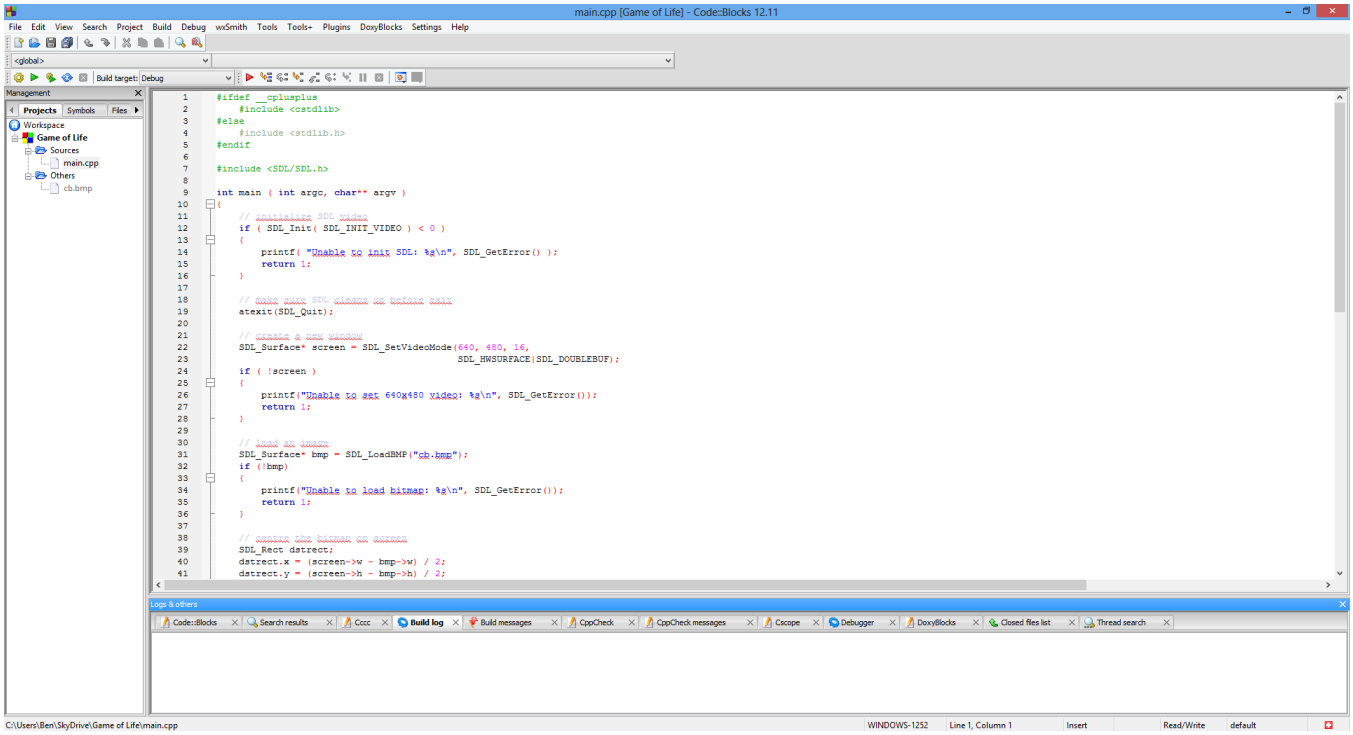

FIGURE 3 – Code par défaut clé en main

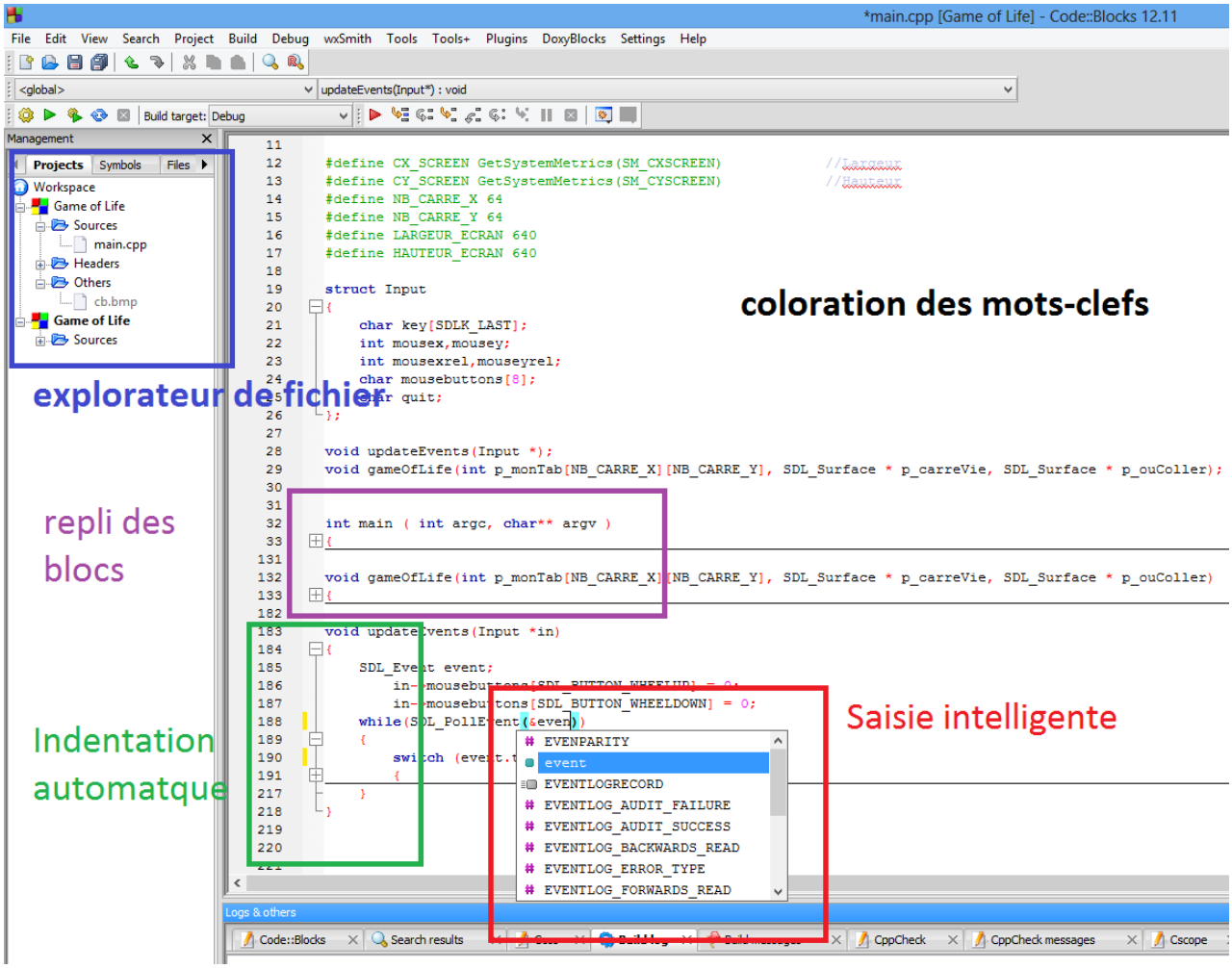

FIGURE 4 – Fonctionnalités intéressantes de l'éditeur de texte

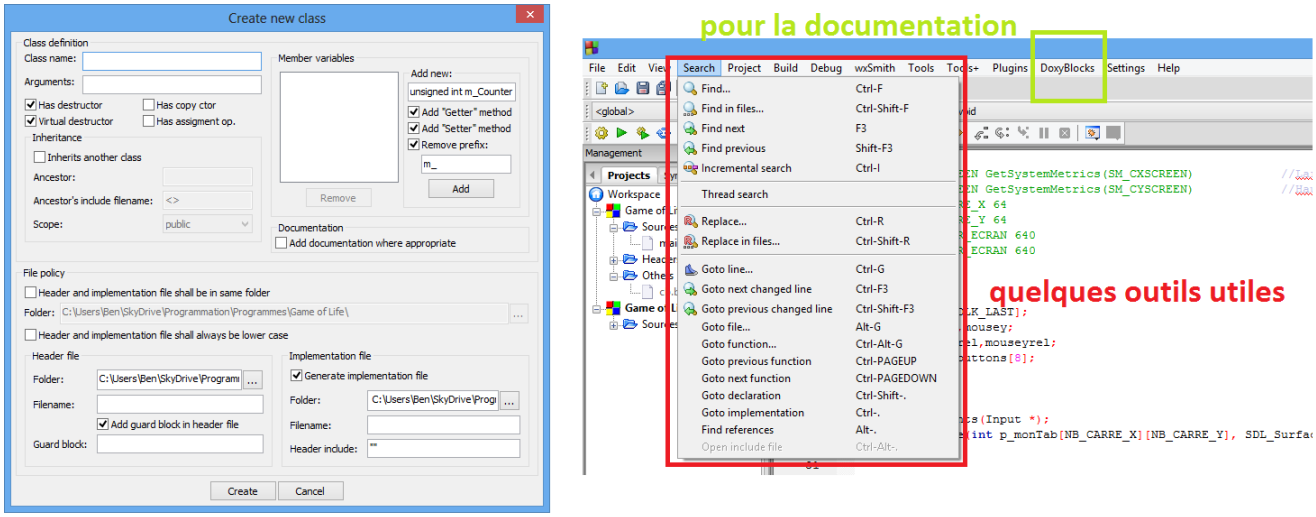

FIGURE 5 – Quelques outils pratiques

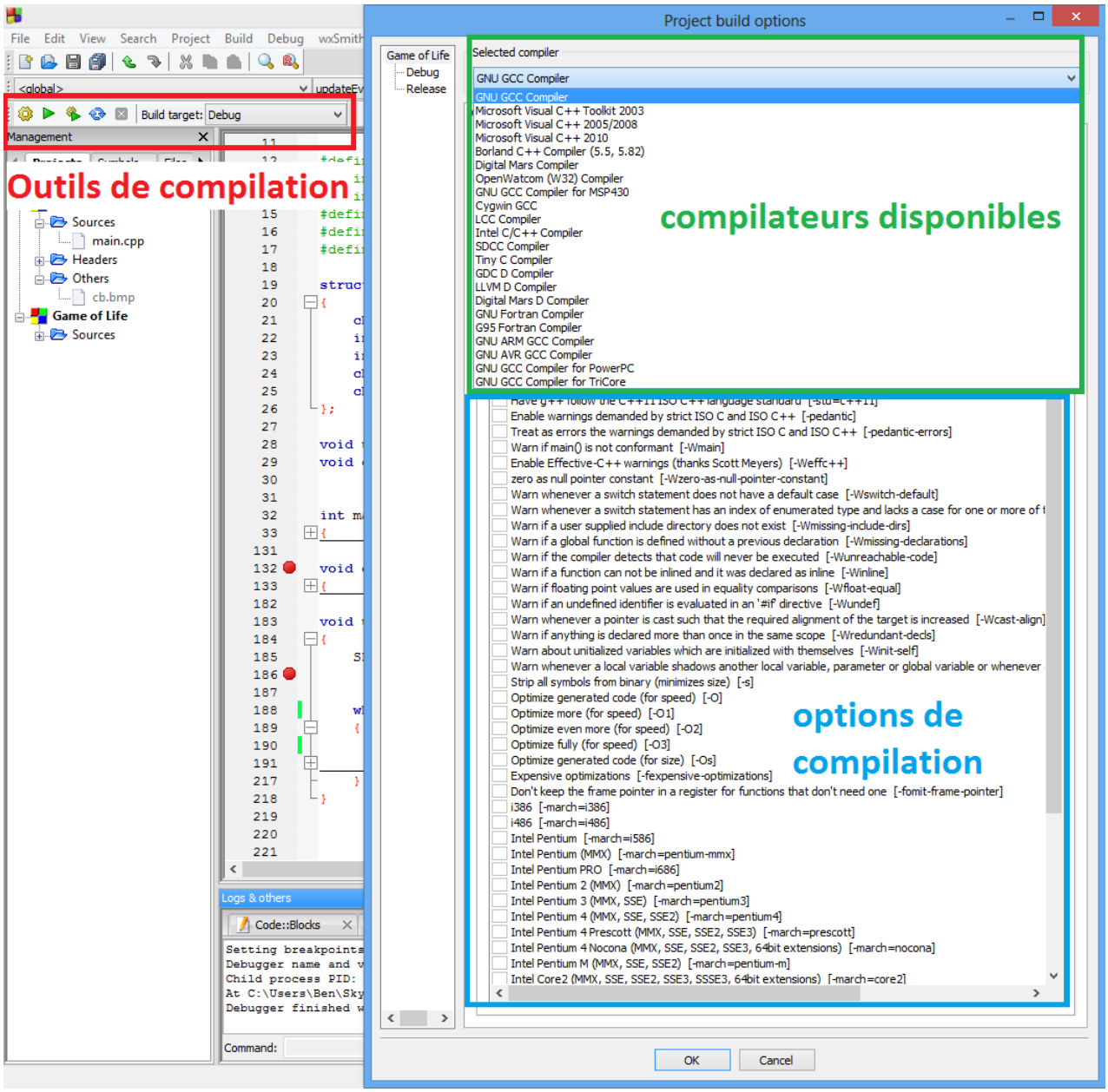

FIGURE 6 – La compilation

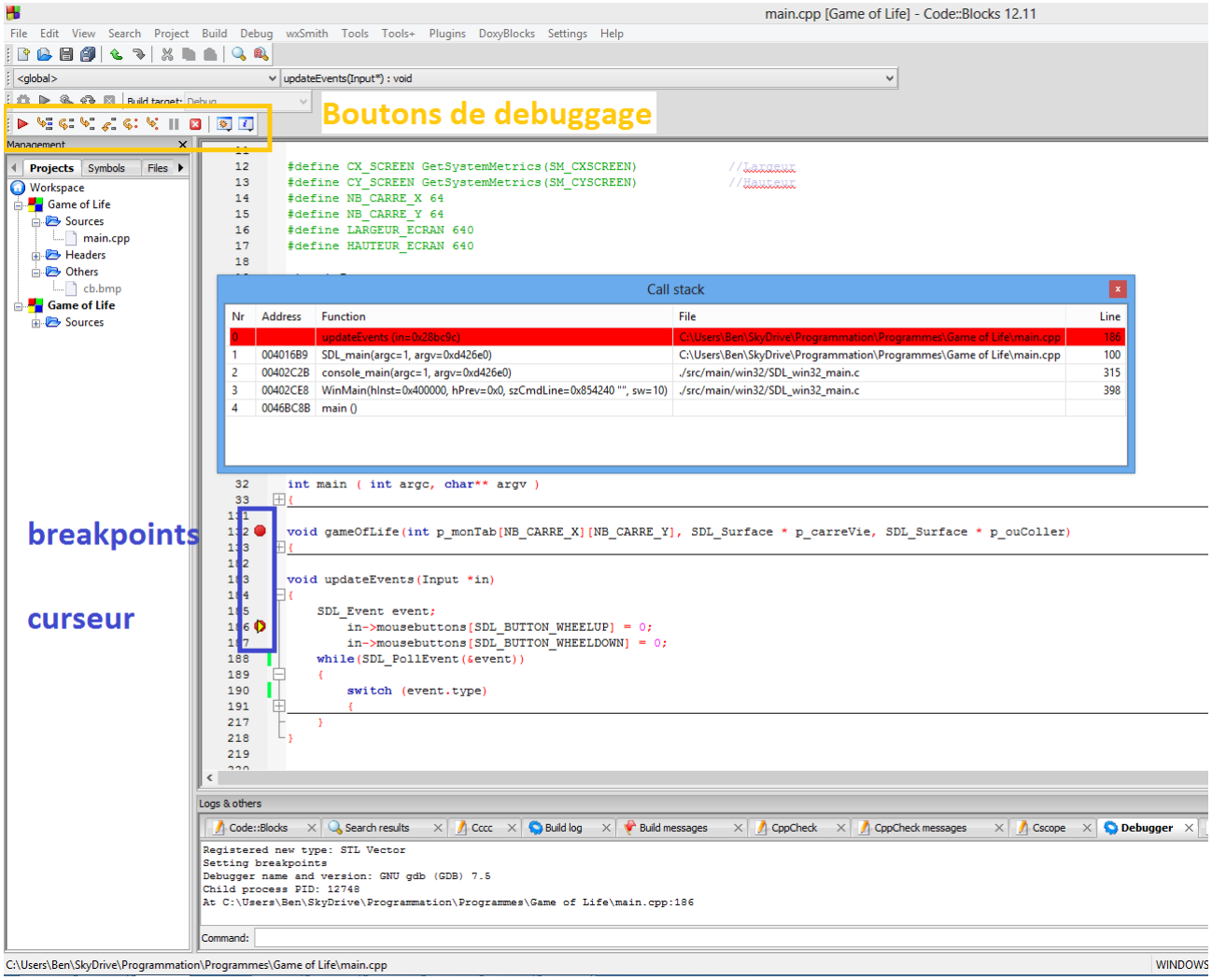

FIGURE 7 – Le débuggage