Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

heig-vd

Département des Technologies de l'Information et de la Communication

Travail de diplôme

TinyBuilder
Éditeur C/CPP

Manuel utilisateur

Professeur : Jean Pierre Molliet
Réaliser par : Boesch Pierre
Makrem Mahjoub

Yverdon-les-Bains, le 17/12/2008

Table des matières

1 Introduction :

Afin d'aider les utilisateurs à mieux comprendre notre application, nous avons réalisé ce document qui contient une explication détaillée de son utilisation. Dans le cadre des projets des diplômes, nous avons choisi de réaliser un outil de création d'interface utilisateur pour des développeurs débutants.

2 Vue générale :

Notre éditeur s'adresse à des utilisateurs débutants, il doit être simple d'utilisation et intuitif pour l'implémentation des applications. Ce défi se repose surtout sur interface graphique qui doit être bien conçu avec une automatisation des pluparts des fonctionnalités. Il est important de pouvoir créer des interfaces utilisateurs facilement, mais il faut aussi que l'utilisateur puisse réaliser des applications utiles.

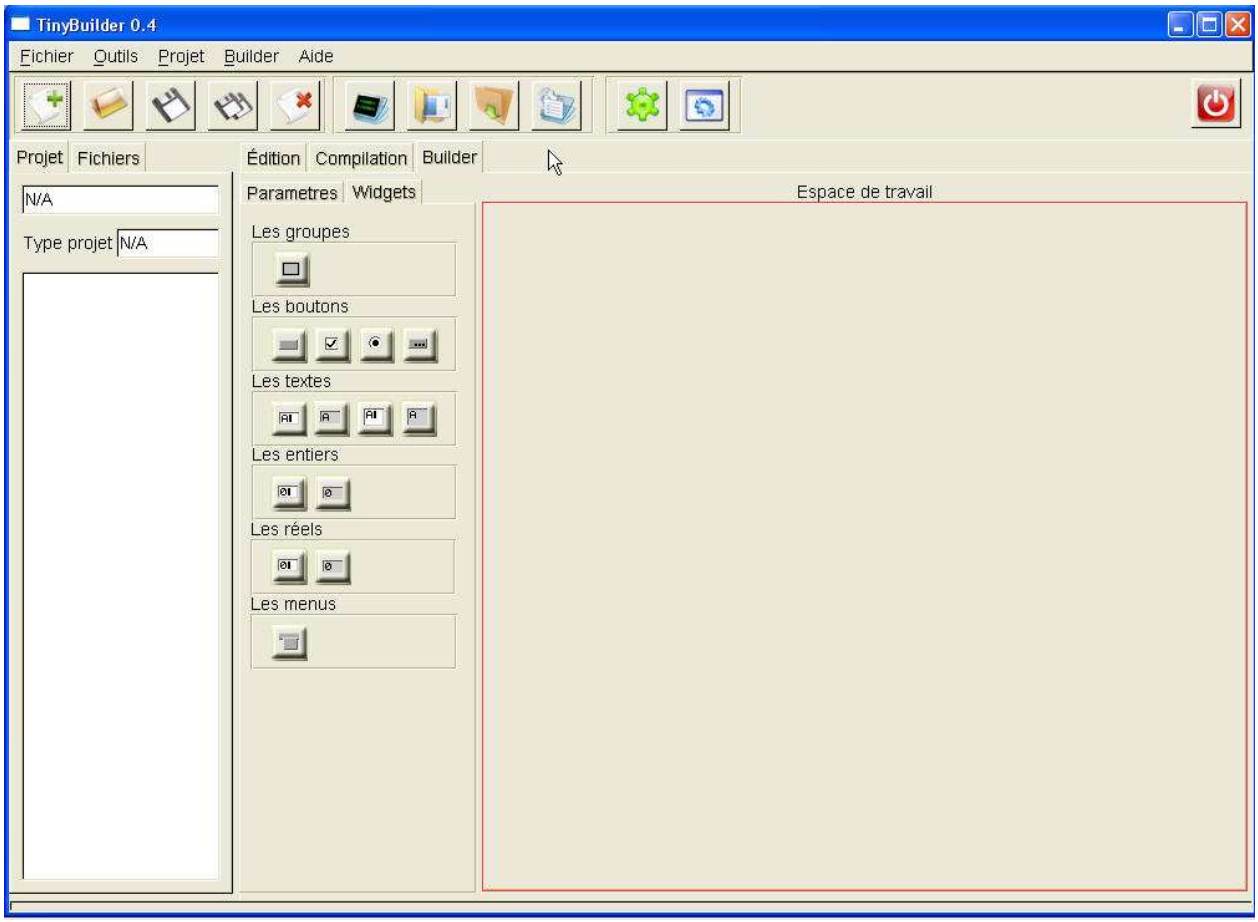

Figure 1 : Interface utilisateur de TinyBuilde.

3 Modules :

3.1 Explorateur des fichiers :

TinyBuilder permet une gestion avancée des fichiers et des projets. Vous avez la possibilité de gérer des fichiers dans un projet tbXML ou tout simplement d'ouvrir et d'éditer des fichiers externes.

Dans l'état initial, aucun projet n'est créé par défaut. Vous avez la possibilité d'utiliser le Builder avec toute liberté sauf la compilation, où il faut créer un projet.

3.1.1 Vue projet :

Cette vue nous renseigne sur les données du projet. Nous y trouvons le nom du projet, son type (tbXML, CPP) ainsi que la liste des fichiers qui se trouve dans le projet. Dans la figure 2, nous n'avons pas de projet ouvert ou crée alors il affiche la valeur par défaut « N/A ».

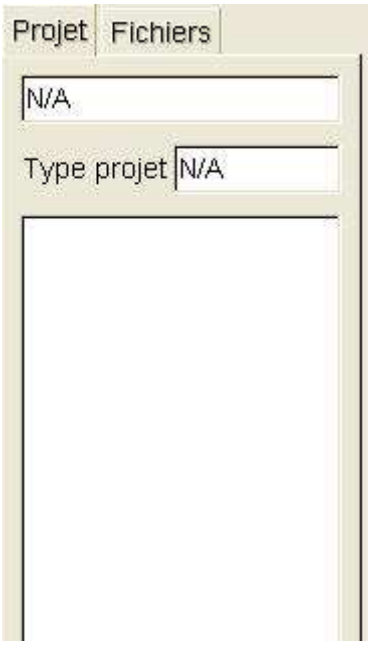

Figure 2 : Explorateur du projet.

3.1.2 Vue fichiers externes :

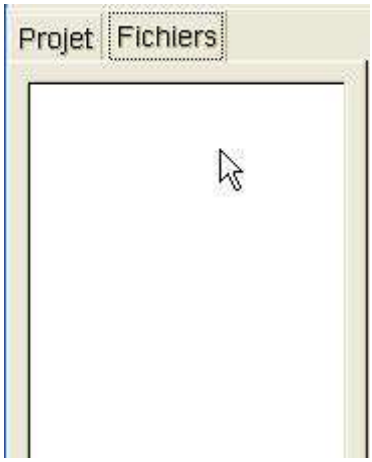

Figure 3 : Explorateur des fichiers.

3.2 Onglet édition :

Cet onglet permet d'éditer des fichiers textes. Vous pouvez ouvrir et éditer tout type de fichier texte. La plupart des touches raccourcis du clavier (ctrl+c ; ctrl+v) fonctionne normalement comme un éditeur de texte normal. Mais cet éditeur reste très minimaliste et n'offre pas toutes les fonctionnalités.

Figure 4 : Onglet éditeur des fichiers textes.

3.3 Onglet Builder :

Cette interface est l'espace de travail pour la création des interfaces utilisateurs. Nous avons ici un espace qui contient la création et le déplacement des widgets. Nous avons également la possibilité de sauvegarder les interfaces utilisateurs créé à l'aide du Builder.

Nous pourrons manipuler les widgets à l'aide de la souris ou à partir de l'onglet paramètre. Le Builder met à disposition les paramètres les plus courants.

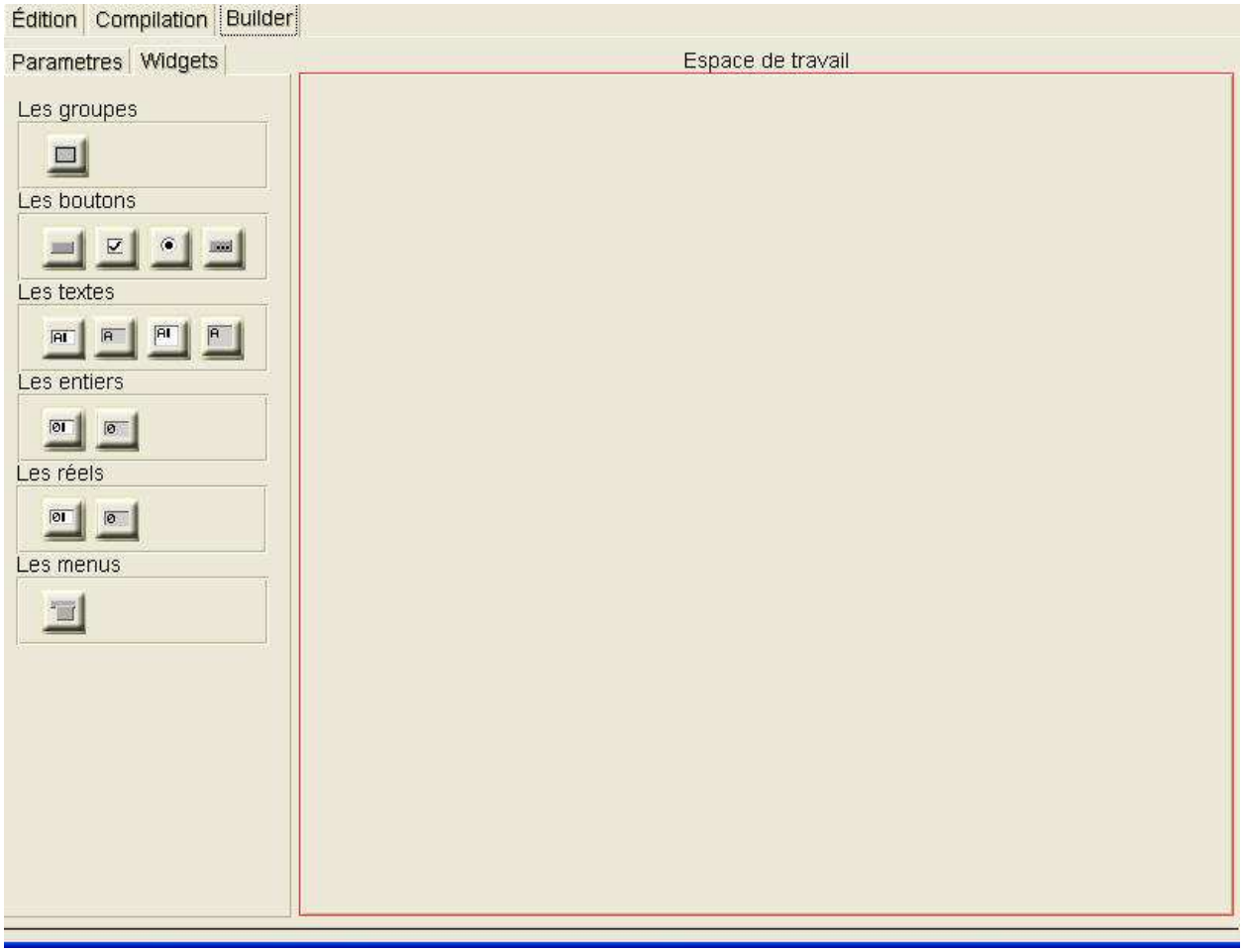

Figure 5 : Onglet Builder

3.3.1 Palette des widgets :

Dans cette palette, nous trouvons les widgets que nous pourrons ajouter dans l'espace de travail pour construire nos interfaces utilisateurs. Pour ajouter un widget dans la fenêtre principale, il suffit de cliquer sur le bouton du type de widget voulu.

Il faut se référer au rapport pour une meilleure compréhension des widgets. Ce qu'il faut savoir de ses widgets qu'elles se devisent en deux groupes dont :

1. Widget simple : Ce sont les boutons, les textes, les entiers et les réels. Ce type de widget ne peut pas contenir d'autre widget.

2. Widget Groupe : Ce sont les widgets qui peuvent contenir d'autres widgets. Ils nous permettent aussi la mise en page des widgets dans la fenêtre.

Figure 6 : Palette des widgets dans l'onglet Builder.

	Groupe : ce widget peut contenir d'autres widgets. La suppression, redimensionnement ou le déplacement de ce widget se percute sur tous ses enfants. Ce widget est utile pour regrouper un ensemble de widget dans un même ensemble.
	Bouton : c'est le widget bouton classique. Nous pourrons l'utiliser pour le lancement un évènement .
	Bouton à coche : Ce widget permet de proposer un choix à l'utilisateur qu'il peut ou pas cocher.
	Bouton à radio : Ce bouton permet de proposer à l'utilisateur un choix qu'il peut cocher ou pas.
	Bouton à répétions : Ce bouton permet d'exécuter des actions tant qu'il est pressé.
	Entrée texte : ce widget permet à l'utilisateur d'enter un texte sur une ligne.
	Sortie : ce widget permet d'afficher un texte. Il n'est pas éditable par l'utilisateur.
	Entrée texte multi ligne : ce widget permet à l'utilisateur d'enter un texte sur plusieurs lignes.
	Sortie : ce widget permet d'afficher un texte sur plusieurs lignes. Il n'est pas éditable par l'utilisateur.
	Entrée entier : ce widget permet à l'utilisateur d'entrer un nombre.
	Sortie entier : ce widget permet d'afficher un nombre. L'utilisateur ne pourra pas la modifier.
	Entrée réel : ce widget permet à l'utilisateur de saisir un nombre réel.
	Sortie Réel : ce widget permet d'afficher des valeurs réels.

4 Manipulation des widgets :

TinyBuiler met à disposition un système de manipulation des widgets très souple. Nous pourrons ajouter et déplacer les widgets à notre guise.

La manipulation des widgets est une fonctionnalité importante du Builder. Nous l'avons amélioré au fur et à mesure que l'implémentation. La plus grande difficulté, était de comment faire pour savoir les limites d'un widget, finalement nous avons trouvé les options nécessaires telles que : le changement du curseur de souris qui permet une manipulation aisée des widgets.

4.1 Déplacement d'un widget :

Pour déplacer un widget, il suffit de placer le curseur de la souris sur le widget en question, en cliquant sur le bouton gauche de la souris, nous pourrons déplacer le widget sélectionné.

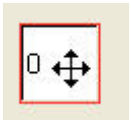

Figure 7 : Déplacement widget (curseur croix)

4.2 Redimensionner le widget :

Pour redimensionner un widget, il faut le sélectionner avec la souris en se plaçant sur l'onglet en bas à droite et se déplacer vers le bas pour agrandir, et en haut pour baisser la taille du widget.

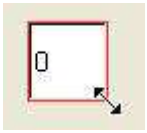

Figure 8 : redimensionner le widget

4.3 Changer la longueur widget

De la même manière que le redimensionnement, mais il faut placer le curseur à droite du widget. En déplaçant le curseur vers la droite pour augmenter le largueur et vers la gauche pour la diminuer.

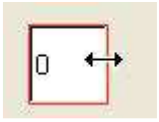

Figure 9 : Modifier la longueur du widget

4.4 Changer la hauteur widget

Pour changer la hauteur du widget, il faut placer le curseur en bas du widget et le déplacer vers le bas pour augmenter le largueur et vers le haut pour le contraire.

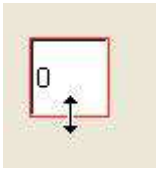

Figure 10 : modifier la hauteur du widget.

4.5 Paramètres widget :

Cet onglet nous donne accès à un grand nombre d'options pour personnaliser les widgets et les interfaces utilisateurs.

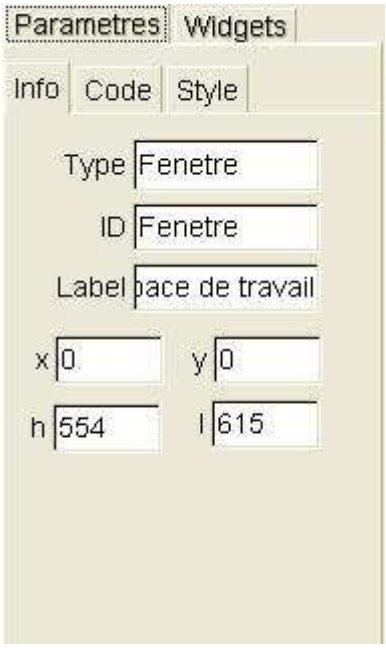

Figure 11 : Fenêtre des paramètres de widget

1. Onglet info : Dans cet onglet, nous avons la possibilité de personnaliser les informations essentielles sur notre widget. Nous y trouvons les coordonnées ainsi que les dimensions. Nous avons aussi les identificateurs du widget tel que l'identificateur et le label. Ce type de widget va nous renseigner sur le type de widget sélectionné à partir de la fenêtre principale.

2. Onglet code : Dans cet onglet, nous avons la possibilité d'entrer le code source qui sera exécuter lors d'un clique de souris par exemple. Nous trouvons aussi le code d'initialisation de chaque widget. Celui-ci va servir à l'initialisation du widget, le contenu par exemple.

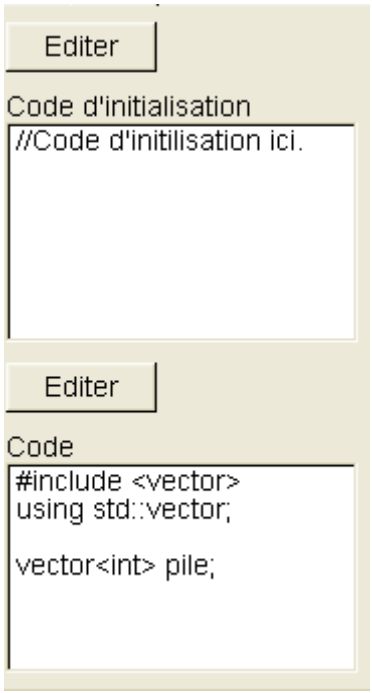

Figure 12 : Onglet gestion des codes sources.

Remarque :

La gestion du code source dans la fenêtre principale est un peu spéciale. Le code d'initialisation va être transformé en une fonction qui sera exécutée en début d'application. Cette partie va être utile pour l'initialisation des données avant le début d'utilisation de l'application.

La deuxième partie du code va servir à ajouter des déclarations globales. Par exemple, pour la déclaration d'une variable globale, nous devons l'ajouter dans cette partie.

Dans cet onglet, nous avons ajouté dernièrement deux boutons qui permettent d'ouvrir une nouvelle fenêtre d'édition du code source. Cette modification est très importante, car elle va nous permettre d'éditer le code source dans une plus grande interface.

Il y a une autre chose à savoir concernant la gestion évènement des boutons. Lorsqu'on clique sur le bouton, nous avons deux évènements, le bouton est en bas ensuite en haut. Pour que le bouton n'exécute pas votre code deux fois, il faut ajouter le code ci-dessous.

```
if (etatBouton())
        return;
```

Figure 13 : code pour la gestion du bouton.

3. Onglet style : Cet onglet nous donne la possibilité de personnaliser le style de police ainsi que la taille. Nous avons aussi la possibilité de personnaliser des couleurs.

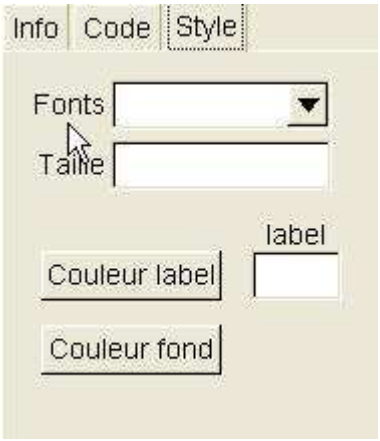

Figure 14 : L'onglet pour le paramétrage du style widget.

Nouveauté : Cette option est ajoutée dernièrement dans TinyBuilder. Elle nous permet de changer la position du label d'un widget. Il faut cliquer sur les flèches pour mettre le label dans une position voulue.

Figure 15 : position du label du widget.

4.6 Changement de l'identificateur

L'identificateur d'un widget, est une information très importante, car elle va nous permettre d'accéder au widget partout dans notre application.

Exemple :

Lorsqu'on ajoute un bouton à l'interface utilisateur, il a par défaut un identificateur « Bouton0 », le deuxième aura comme identificateur « Bouton1 » et ainsi de suite…

Remarque :

Il est préférable de changer les identificateurs par défaut, car lors de la réouverture d'un projet existant, les identificateurs s'ajoutent par défaut comme ci-dessus, ce qui peut provoquer des conflits.

Scénarios :

Les identificateurs se font de la même manière, alors il est préférable de changer les identificateurs pour éviter les conflits.

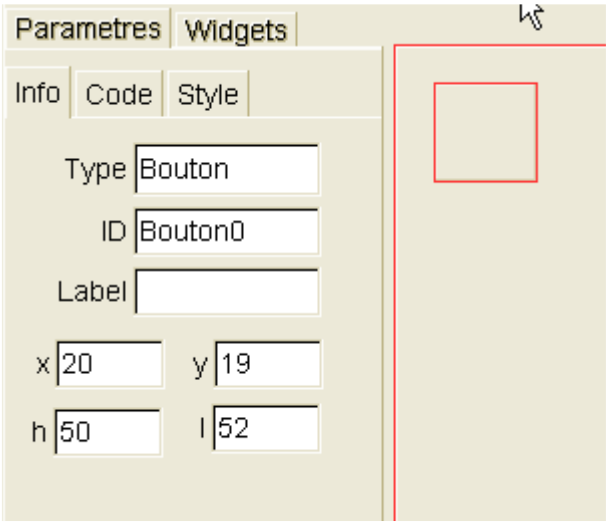

Figure 16 : ajouter un bouton dans l'espace du travail.

4.7 Supprimer un widget

Vous pouvez supprimer un widget en sélectionnant et appuyer sur les touches « ctrl+del » ou bien à partir de la barre de menu.

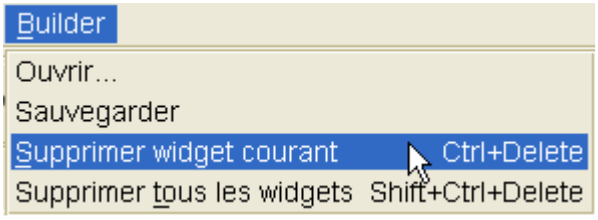

Figure 17 : Supprimer le widget courant.

4.8 Onglet compilation :

Dans cet onglet, nous trouvons les messages récupérés depuis le compilateur. Selon la configuration du projet (tbXML, CPP), nous avons une compilation différente.

Lorsqu'on lance la compilation, TinyBuilder sauvegarde l'interface utilisateur et génère le code source. Nous avons laissé un commentaire qui affiche le widget en cours de traitement lors de génération du code source.

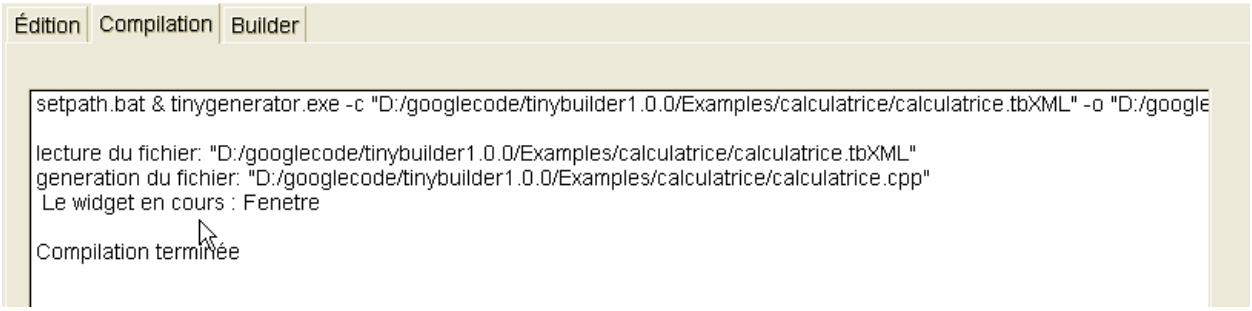

Figure 18 : Onglet compilation projet dans TinyBuilder

4.9 Barre de menu :

La barre de menu va nous permettre d'exécuter des fonctionnalités de TinyBuilder. Nous allons expliquer brièvement le but de chaque rubrique.

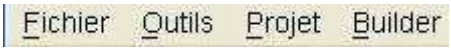

Figure 19 : barre de menu de TinyBuilder

1. Fichier :

Figure 20 : Le sous-menu Fichier.

a. Nouveau : création d'un nouveau fichier texte
b. Ouvrir : Ouvrir un fichier texte dans l'éditeur
c. Sauvegarder : Sauvegarder un fichier texte depuis l'éditeur du code source
d. Fermer : Fermer le fichier courant de l'éditeur des fichiers
e. Quitter : Quitter l'application

2. Outils :

Figure 21 : La rubrique outils de la barre de menu

a. Compiler : Compiler le projet courant (il faut qu'un projet soit créé)
b. Exécuter : Exécuter le programme qui est dans le projet
c. Générer code source : Génération du code source à partir d'un fichier tbXML. Il existe deux issues pour cette fonctionnalité, les voici :
    i. Un projet est ouvert : Le code générer est celui du fichier tbXML du projet.
    ii. Aucun projet : Une boite de dialogue s'ouvre pour demander à l'utilisateur de donner le fichier tbXML.

3. Projet :

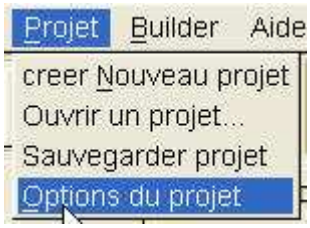

Figure 22 : la rubrique projet

a. Créer nouveau projet : création d'un nouveau projet
b. Ouvrir un projet : ouvrir un projet enregistré dans un fichier XML
c. Sauvegarder projet : sauvegarder un projet
d. Options du projet : édition des options du projet

4. Builder :

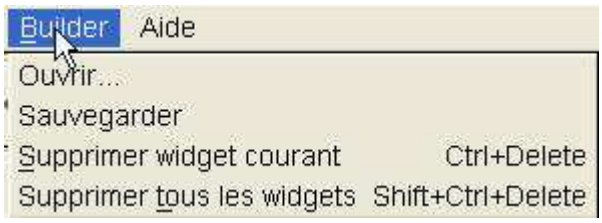

Figure 23 : l'onglet builder de la barre de menu

a. Ouvrir … : ouvrir une interface utilisateur depuis fichier XML
b. Sauvegarder : sauvegarder l'interface dans un fichier XML
c. Supprimer widget courant : supprimer le widget courant
d. Supprimer tous les widgets : supprimer tous les widgets.

4.10 Raccourcis fichiers

Figure 24 : Les raccourcis des fichiers

 Création d'un nouveau fichier texte

 Ouvrir un fichier texte dans l'éditeur

 Sauvegarder un fichier texte depuis l'éditeur du code source

 Sauvegarder tous fichiers.

 Fermer le fichier courant de l'éditeur de fichier

4.11 Raccourcis projets

Les raccourcis ci-dessous vont nous permettre de gérer nos projets. Nous allons vous décrire brièvement les fonctionnalités de mise à disposition.

Figure 25 : les raccourcis de la gestion du projet.

 Création d'un nouveau projet

 Ouverture d'un projet

 Sauvegarder le projet courant

 Éditer les options du projet

4.12 Raccourcis compilation

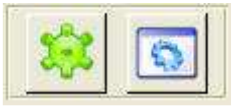

Figure 26 : Raccourci de compilation

 Compiler le projet courant

 Exécuter le programme du projet

5 Fonctionnalités :

5.1 Création d'un nouveau projet :

Cette fonctionnalité permet de créer un nouveau projet. Elle est possible depuis la barre de menu ou bien du raccourci des projets.

- Demande de création d'un nouveau projet : depuis la barre de menu ou le raccourci  , nous pourrons lancer la procédure de création d'un nouveau projet.

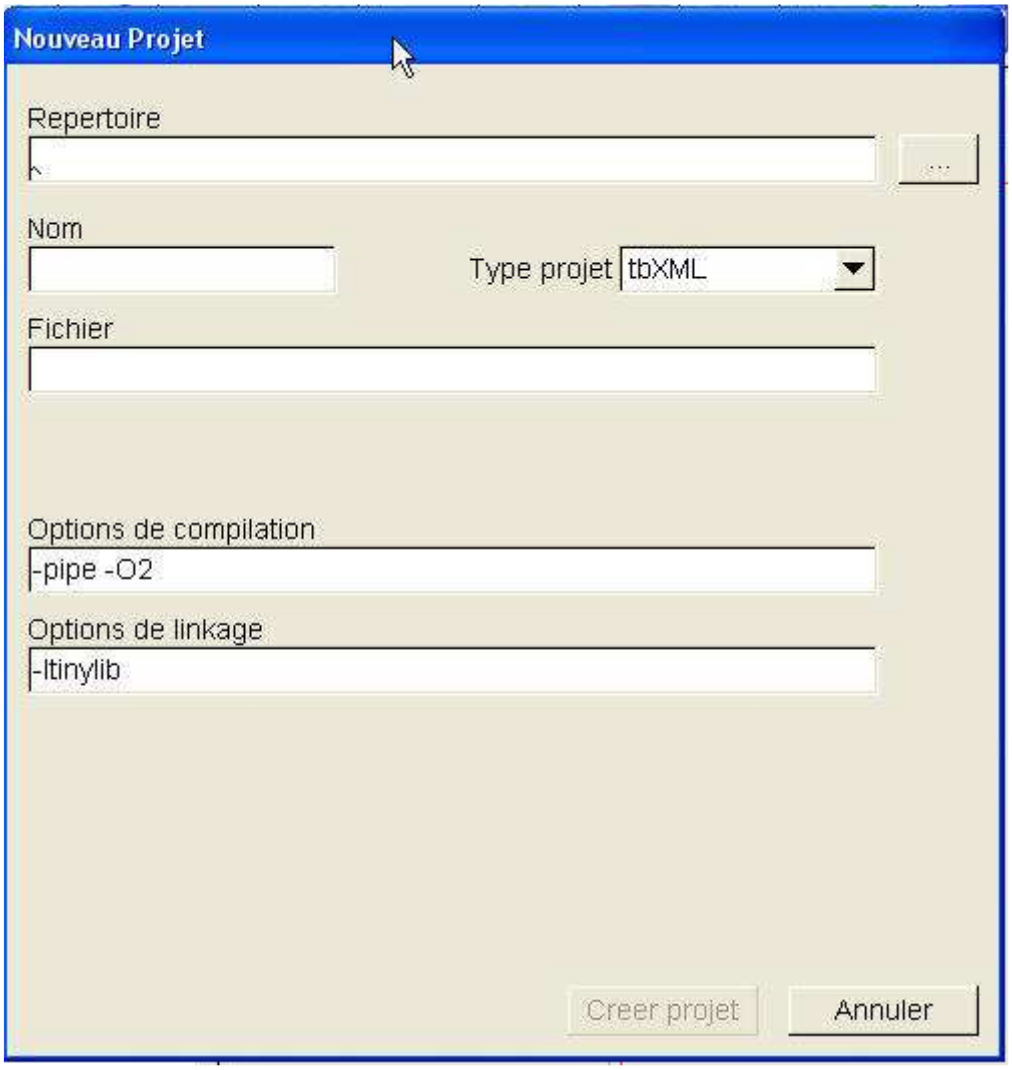

Figure 27 : La boite de dialogue de création d'un nouveau projet.

- Remplir le formulaire de création d'un nouveau projet :

a. Sélection du répertoire du projet :

Il faut commencer par choisir un répertoire pour le projet. Il faut cliquer sur le bouton « … » (voir figure 16, emplacement du curseur souris) pour choisir un répertoire pour votre projet. Vous avez la possibilité aussi de créer de nouveaux répertoires (voir b.).

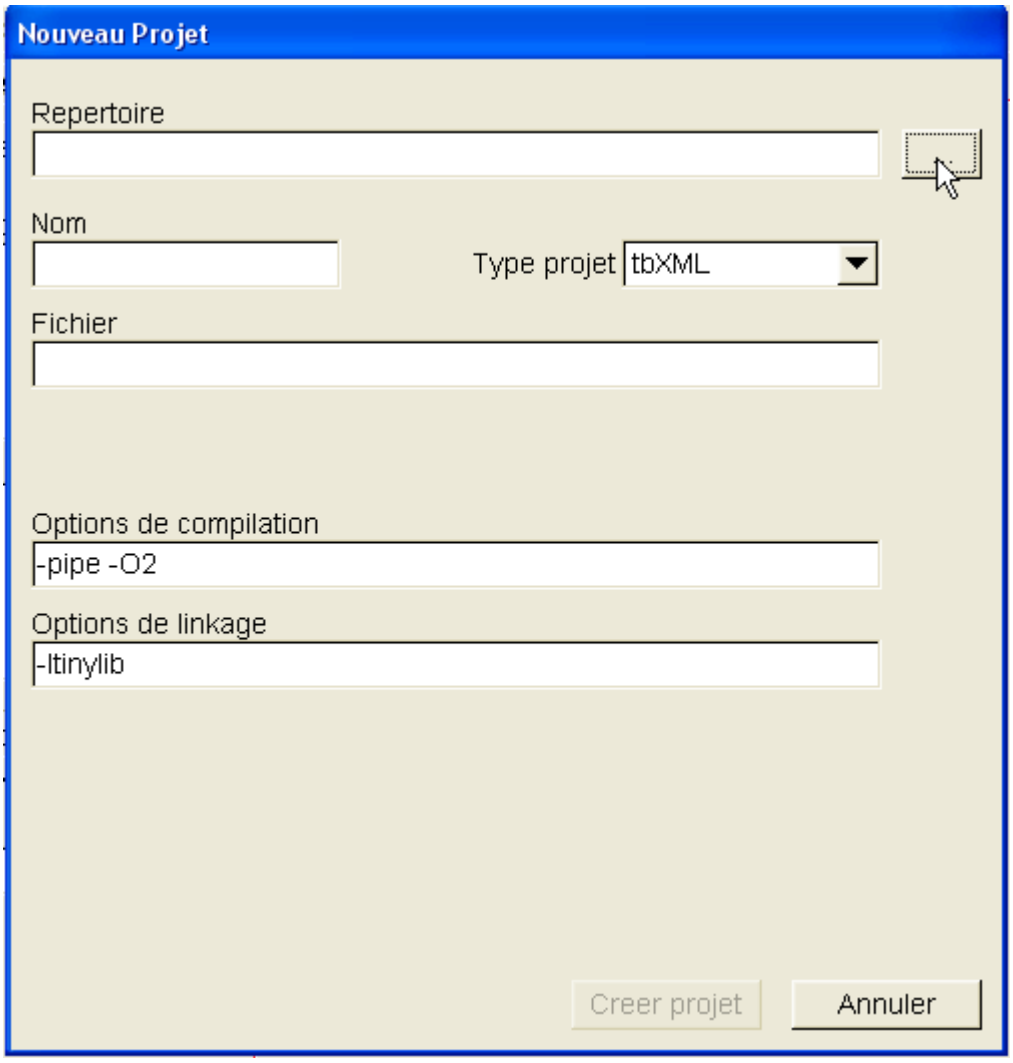

Figure 28 : sélection du répertoire du projet

b. Création d'un nouveau répertoire :

Il faut cliquer sur le bouton « + » (voir figure 17, emplacement curseur) et une nouvelle fenêtre sera ouverte pour vous demander le nom du répertoire à créer (voir figure 18).
Il faut sélectionner le répertoire apparent pour créer le répertoire dans celui-ci.

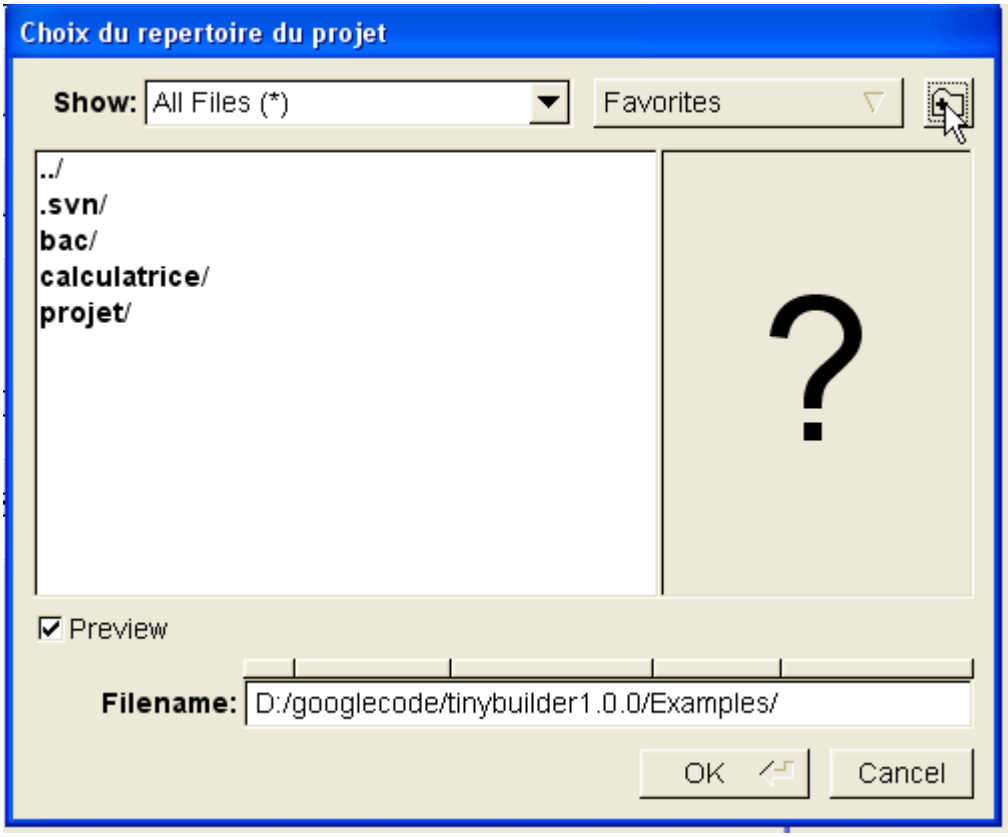

Figure 29 : création d'un nouveau répertoire

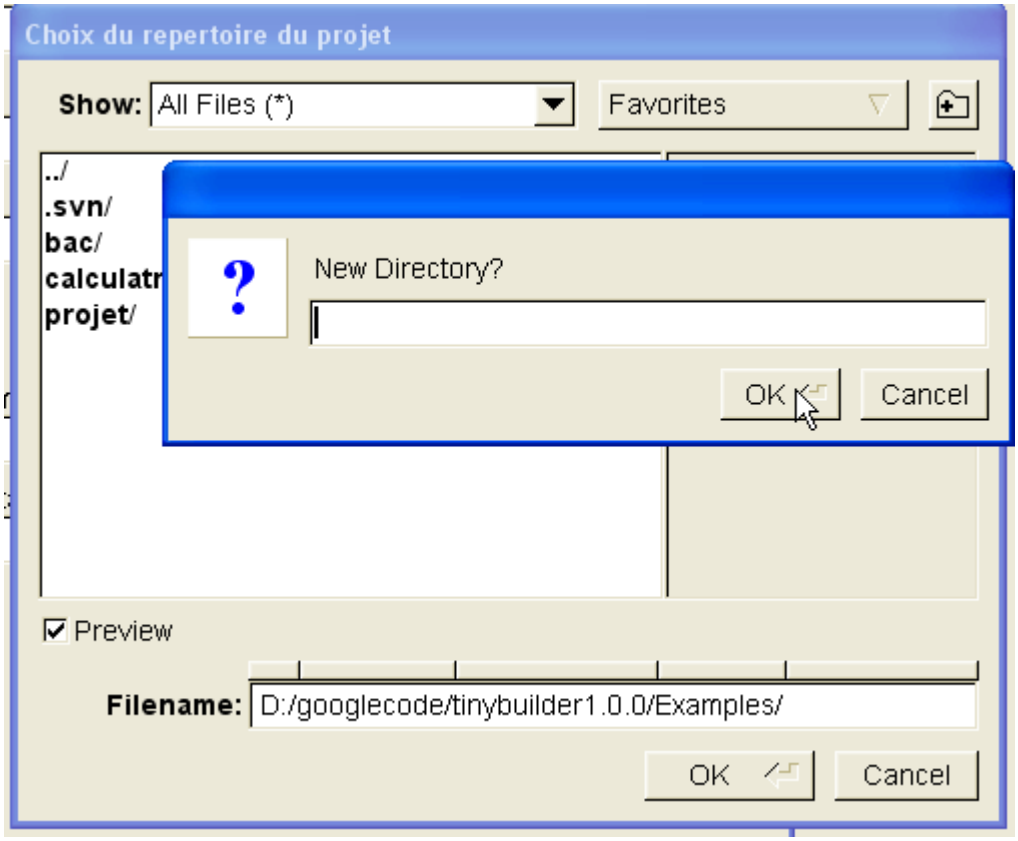

Figure 30 : Saisie du nom de répertoire à créer.

Nous avons besoin d'entrer le nom de répertoire à créer. Cliquer sur « ok » pour valider la création du répertoire.

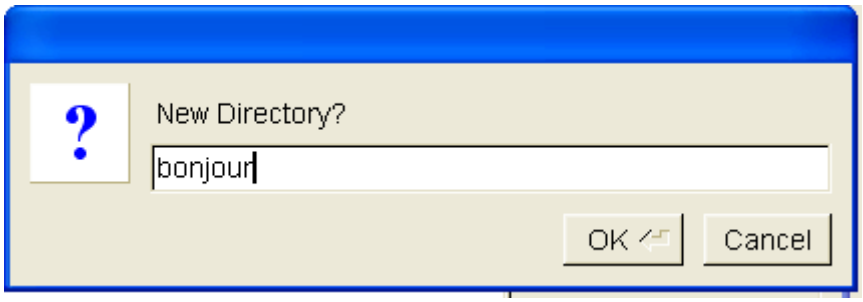

Figure 31 : Saisie du nom répertoire. Suite.

c. Saisie du nom du projet : sous le nom du projet, il faut taper un nom de projet (voir figure 20).

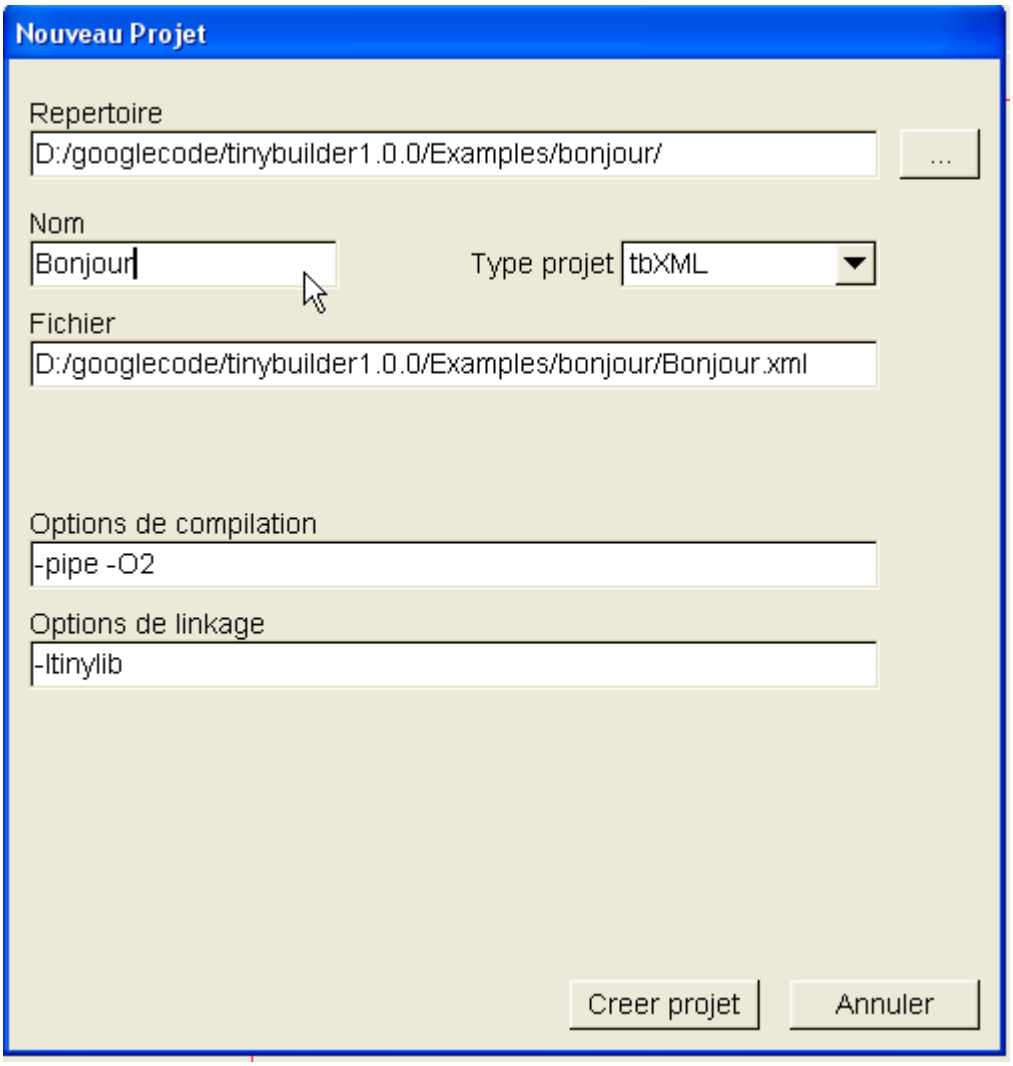

Figure 32 : Création d'un nouveau projet. Suite.

d. Type du projet : Nous avons la possibilité de créer des projets de type tbXML, c'est-à-dire avec une interface utilisateur construite avec TinyBuilder ou bien un projet CPP.

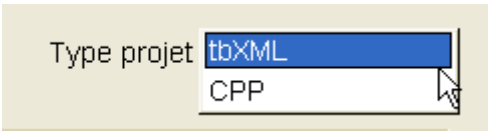

Figure 33 : type de projet TinyBuilder

e. Valider ou annuler :
   i. Validation de la création : appuyer sur le bouton « Créer projet ».
   ii. Annuler la procédure de création de projet.

Le reste des options se met automatiquement. Nous avons maintenant créé un projet et nous pourrons modéliser notre interface utilisateur. Il y a un fichier tbXML qui existe par défaut mais il ne contient aucun widget.

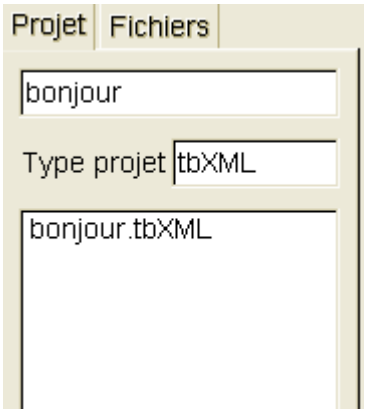

Figure 34 : Un nouveau projet

5.2 Ouverture d'un projet

Cette fonctionnalité consiste à l'ouverture d'un projet déjà enregistré sur le disque. Les fichiers des sauvegardes des fichiers TinyBuilder sont des fichiers XML. Ils contiennent les informations nécessaires à la réouverture du projet.

Vous pouvez ouvrir le projet depuis la barre de menu (voir figure 23) ou par le bouton  de la liste des raccourcis projet.

Figure 35 : Ouverture d'un projet tbXML

Après la demande d'ouverture d'un projet, une boite de dialogue vous demande de choisir le projet à ouvrir (voir figure 24).

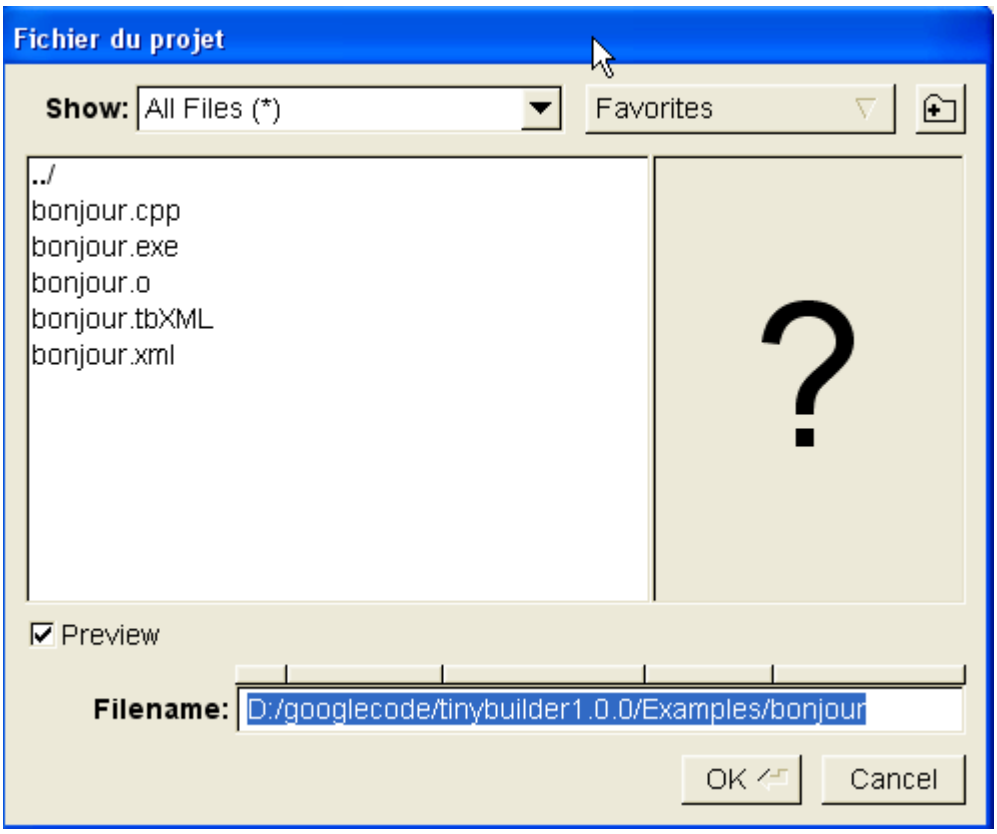

Figure 36 : ouverture d'un projet tbXML

Il faut sélectionner le fichier XML qui contient les données d'un projet. Dans notre cas c'est le fichier « bonjour.xml » qui contient les informations du projet.

5.3 Sauvegarder projet

Lors de la sauvegarde d'un projet, TinyBuilder sauvegarde tous les fichiers du projet ainsi que le fichier utilisateur. Elle est lancée à partir de la barre de menu (voir figure 25) et en cliquant sur le raccourci. 

Figure 37 : sauvegarder projet

5.4 Options projet

En cliquant sur les options du projet depuis la barre de menu (voir figure 26) ou le raccourci  , nous pourrons changer les paramètres du projet.

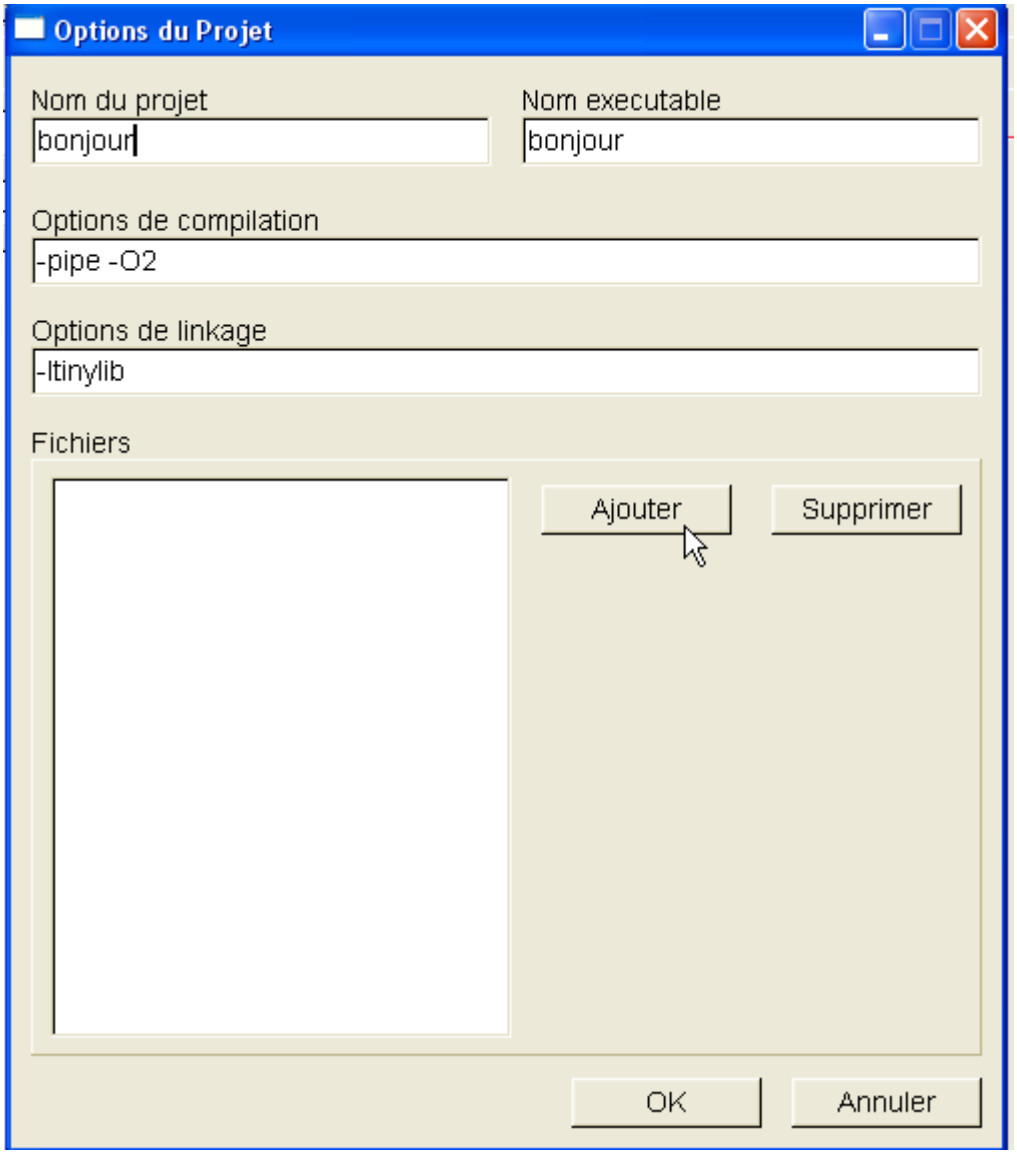

Figure 38 : Options du projet

- Nom du projet : Changer le nom du projet
- Nom exécutable : Changer le nom de l'exécutable
- Options de compilation : Paramètres de compilation
- Options de linkage : Paramètres de liaisons lors de la compilation
- Liste des fichiers : La liste des fichiers qui font partie du projet
    a. Ajouter : ce bouton nous permet d'ajouter un fichier au projet. Lorsqu'on clique dessus, une boite de dialogue nous permet de sélectionner le fichier à ajouter.

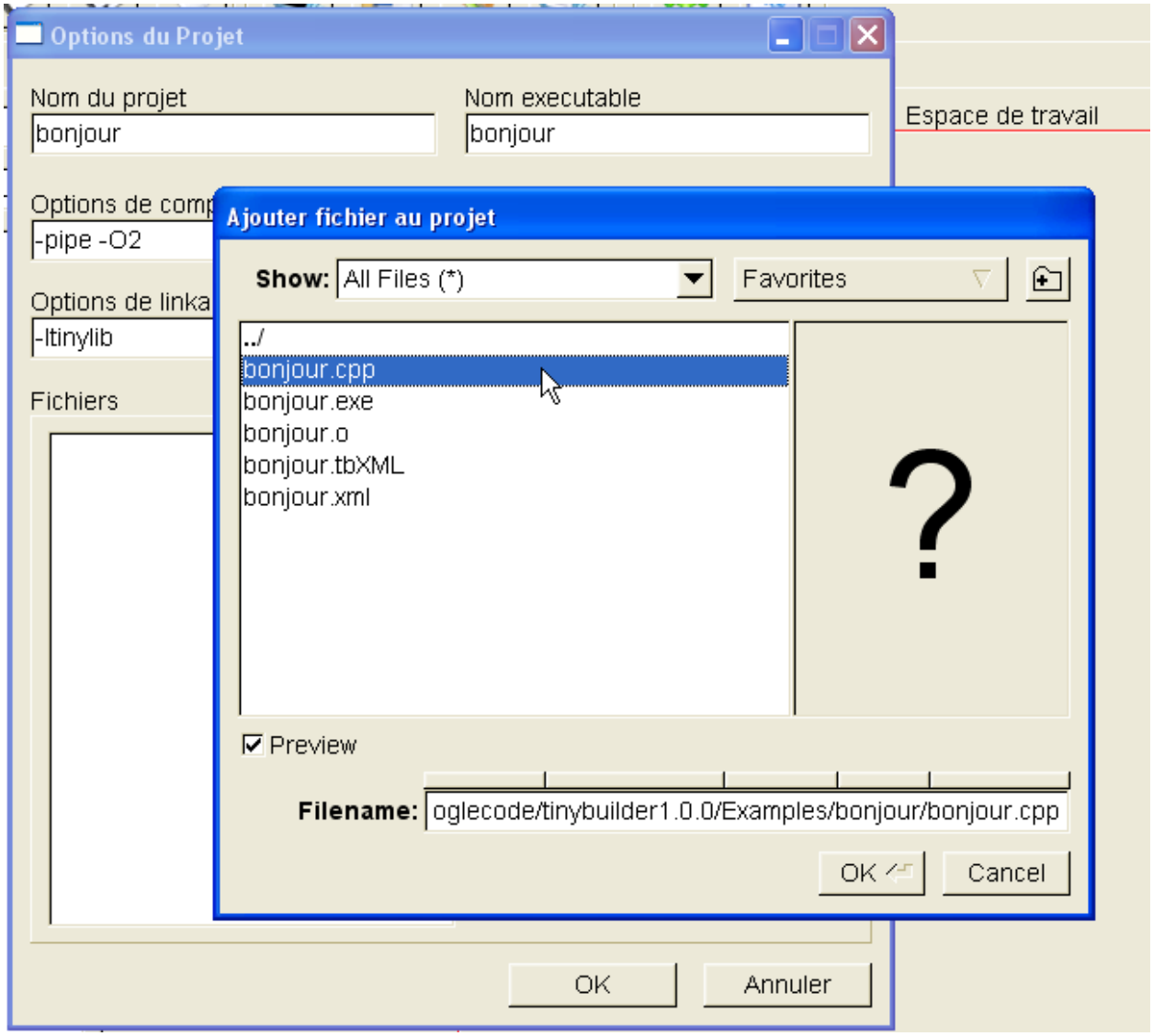

Figure 39 : Ajouter un fichier au projet courant

b. Supprimer : Ce bouton nous permet de supprimer un fichier du projet courant. Il faut le sélectionner depuis la liste des fichiers, ensuite cliquer sur le bouton « supprimer » et enfin valider avec « ok ».

5.5 Compiler un projet :

Cette fonctionnalité permet de compiler un projet TinyBuilder. Il faut avoir un projet déjà créé pour que la compilation se réalise correctement. Il suffit de cliquer sur le raccourci  depuis la liste des raccourcis compilation, à partir de la barre de menu, ou en appuyant sur la touche « F9 » de votre clavier.

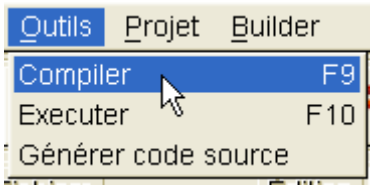

Figure 40 : compilation du projet.

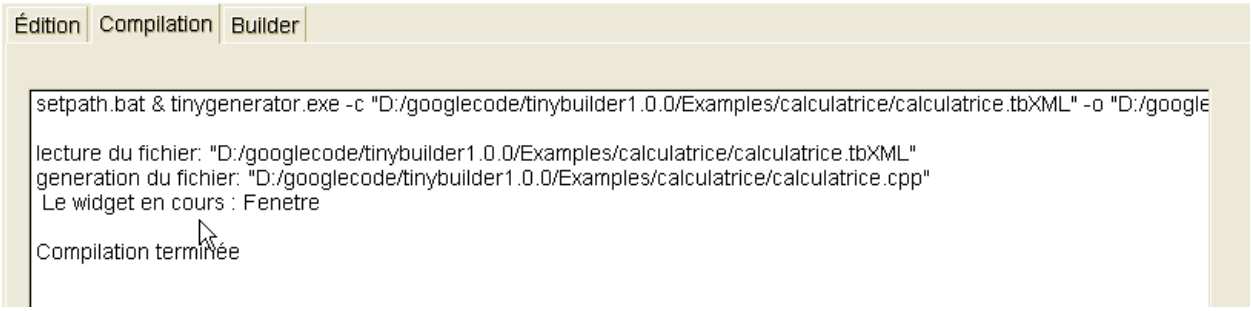

Figure 41 : un exemple de compilation d'un projet.

5.6 Exécuter l'application :

Les projets compilés peuvent être exécuté directement depuis TinyBuilder. Il suffit de cliquer sur le raccourci exécuté  , à partir de la barre de menu « Outils/Exécuter » ou en appuyant sur la touche « F10 » de votre clavier.

5.7 Récupération des erreurs :

Nous avons mis en place un système qui récupère les erreurs depuis la console de message du compilateur, dans le but de retrouver dans le fichier source : le ou les erreurs.

Nous allons ajouter une erreur dans le code et vous faire une démonstration de cette fonctionnalité. Nous allons ouvrir un projet test et introduire du texte dans le code d'initialisation de la widget.

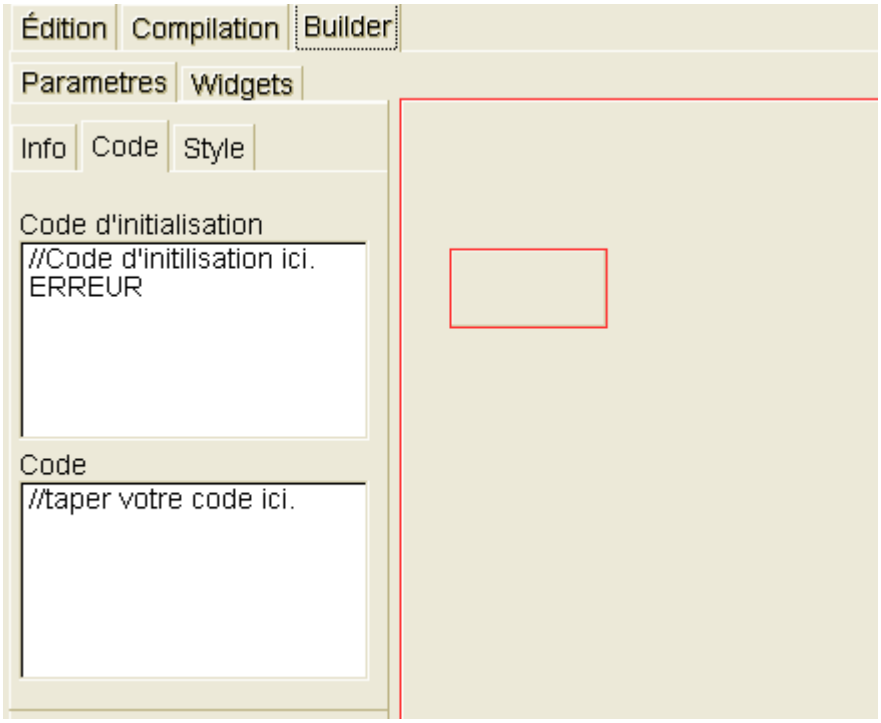

Figure 42 : introduction d'une erreur dans le code

Après la compilation du projet, voici le résultat :

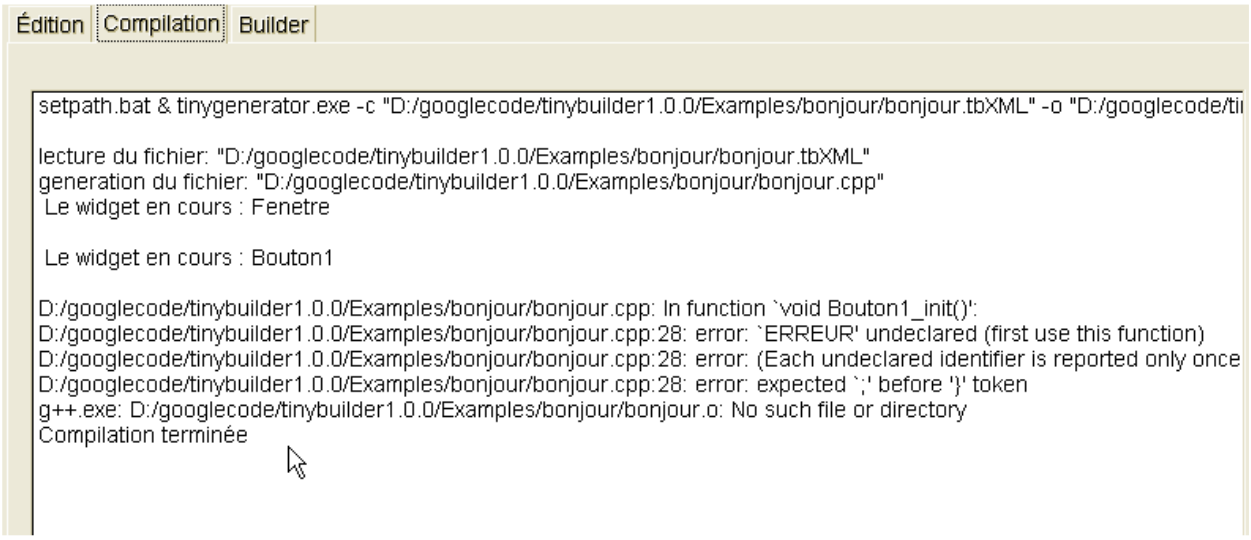

Figure 43 : Message d'erreurs du compilateur.

Si nous voulons savoir ou est ce qu'elle est dans le fichier source, il suffit de cliquer sur la ligne dans l'onglet compilation, ainsi TinyBuilder va sélectionner la ligne en question.

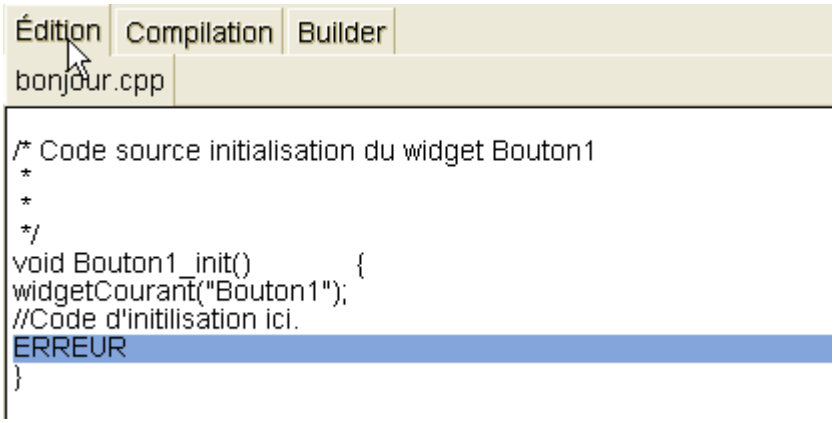

Figure 44 : sélection de l'erreur dans le fichier source.

5.8 Bonjour tout le monde :

Nous allons créer une petite application qui permet de récupérer une chaîne de caractères depuis une entrée texte, et l'afficher dans une sortie texte lorsqu'on clique sur un bouton.

Nous allons commencer par mettre les widgets dans l'espace de travail. Il faut ajouter à la fenêtre principale un bouton et deux entrées textes.

A partir de la palette des widgets, cliquer sur le bouton.

Figure 45 : Ajouter un bouton à la fenêtre principale

Un bouton va s'ajouter dans la fenêtre principale que vous pouvez déplacer comme vous le souhaitez.

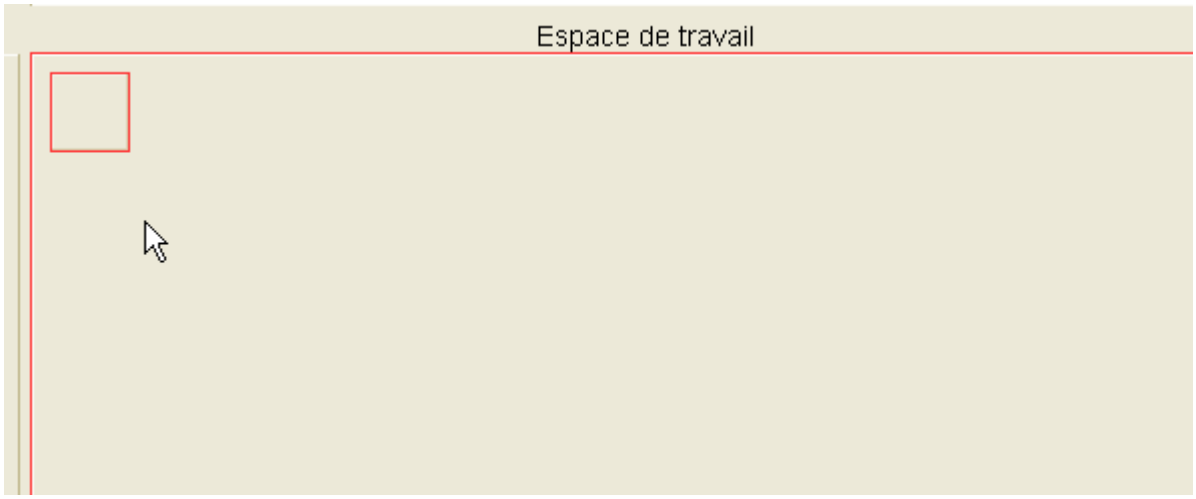

Figure 46 : Ajouter bouton à la fenêtre principale

Il faut faire la même chose avec les autres widgets que nous avons besoin.

Maintenant, nous avons fini la construction de notre interface utilisateur et voici le résultat :

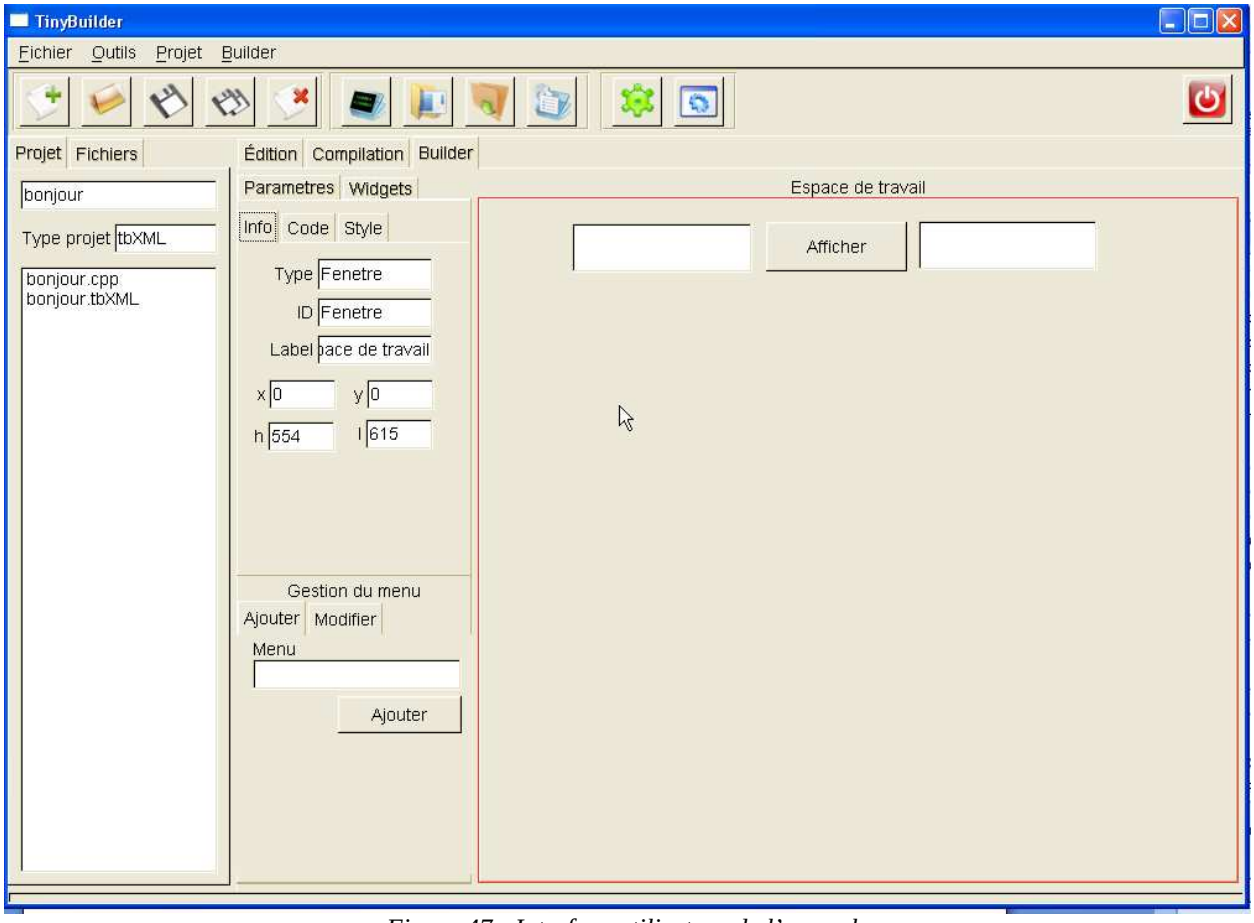

Figure 47 : Interface utilisateur de l'exemple

Nous allons ajouter le code source nécessaire pour copier le contenu de l'entrée texte et l'afficher dans la sortie texte.

```
//taper votre code ici.
char buffer [512];
getTexte("EntreeTexte0",buffer);
setTexte("SortieTexte0", buffer);
```

Figure 48 : code du bouton

Dans ce cas, nous avons deux fonctions : la première getTexte() qui va lire le contenu de la première widget, et la deuxième qui va copier le contenu dans la sortie texte.

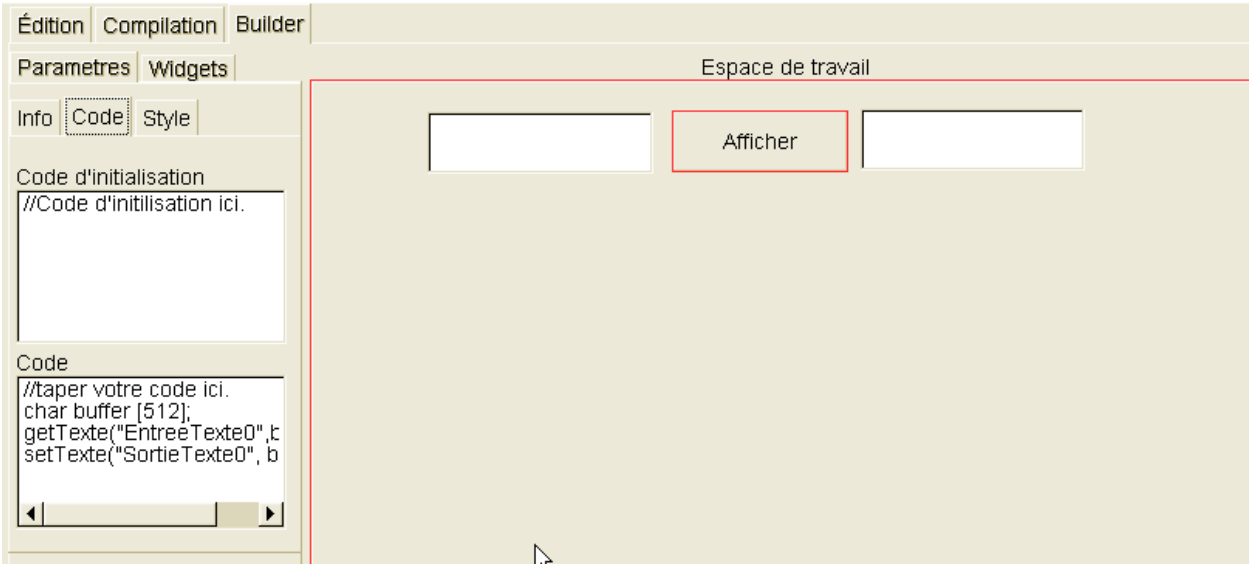

Figure 49 : interface exemple

6 Création d'une calculatrice :

Nous allons créer dans cet exemple une simple calculatrice polonaise.

1. Création d'un nouveau projet

   Nous avons besoin de créer un nouveau projet, car TinyBuilder ne permet la compilation qu'au sein d'un projet.

   Dans la barre d'outils, cliquer sur Projet/Créer nouveau projet

Figure 50 : Création d'un nouveau projet.

Dans la boite de dialogue qui s'ouvre, il faut indiquer les informations suivantes :

- Répertoire du travail : le répertoire de travail
- Nom du projet : nom de projet à créer dans notre cas « calculatrice »
- Type de projet : le type par défaut est « tbXML ». Nous n'allons pas le changer dans ce type de projet, car nous allons créer une interface utilisateur.

Il faut cliquer sur le bouton « Créer projet» pour valider la création du projet.

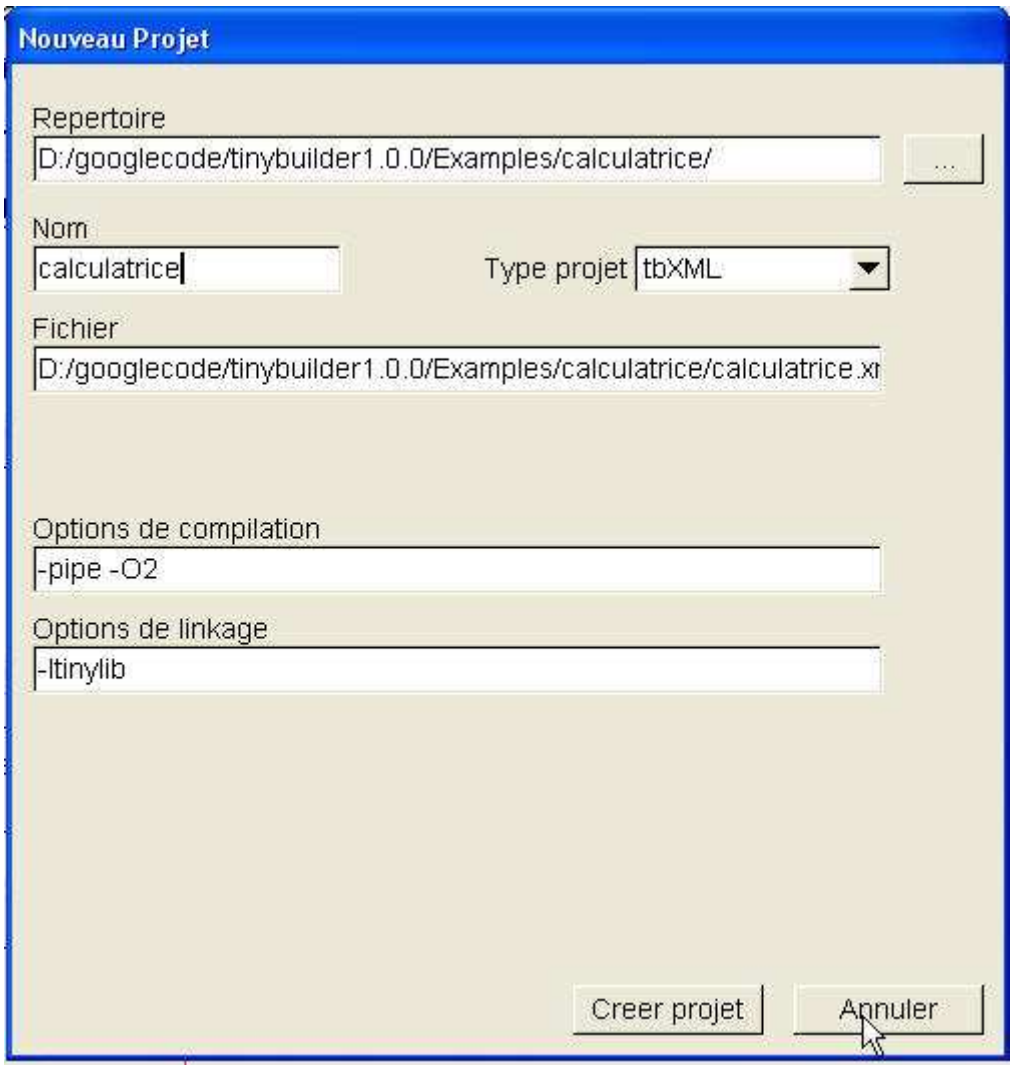

Figure 51 : Boite de dialogue pour la création d'un nouveau projet.

TinyBuilder va nous créer un fichier de sauvegarde pour le projet, et un autre pour la sauvegarde de l'interface utilisateur.

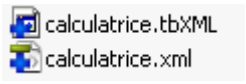

Figure 52 : Les fichiers du projet.

Nous allons maintenant ajouter les widgets à l'interface utilisateur :

- Ajouter une entrée entière pour la saisie et l'affichage des données.

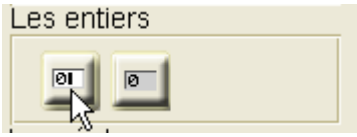

Figure 53 : les widgets des entrées/sorties des entiers.

Le widget vient se placer dans l'espace de travail.

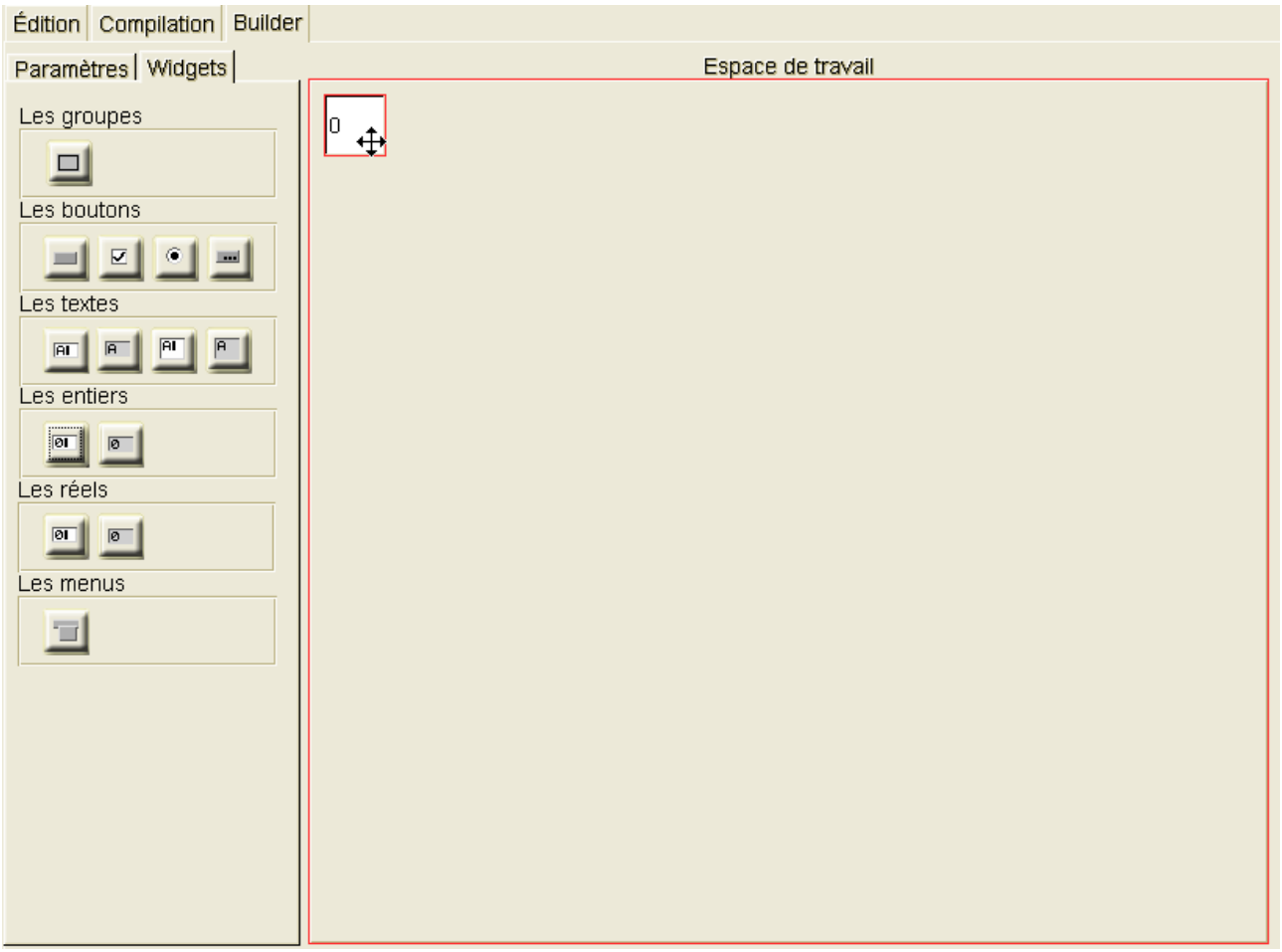

Figure 54 : Ajouter une entrée entière dans l'espace de travail.

Nous allons ensuite la déplacer pour construire notre calculatrice.

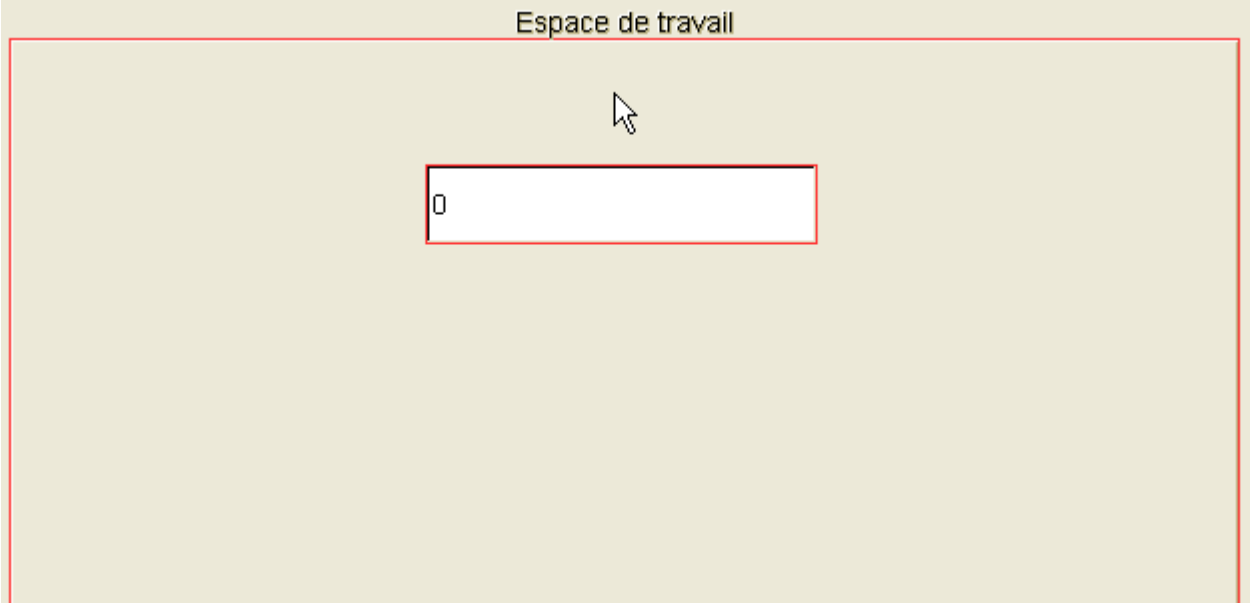

Figure 55 : Mette l'écran de la calculatrice en place.

Nous allons maintenant paramétrer le widget en changeant l'identificateur et le label du notre futur écran de calculatrice.

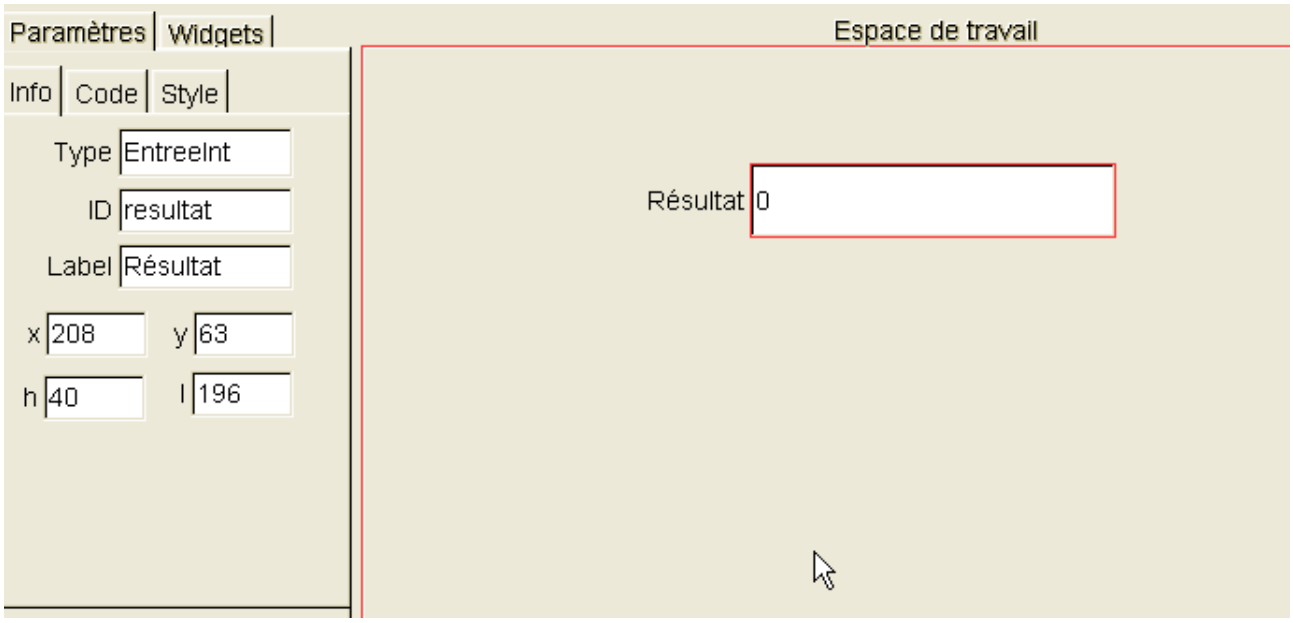

Figure 56 : Mise à jour des informations de l'écran.

- Ajouter les boutons chiffres.

Maintenant, nous allons ajouter les boutons des chiffres. Nous allons ajouter le premier bouton en cliquant sur le bouton depuis la palette des widgets.

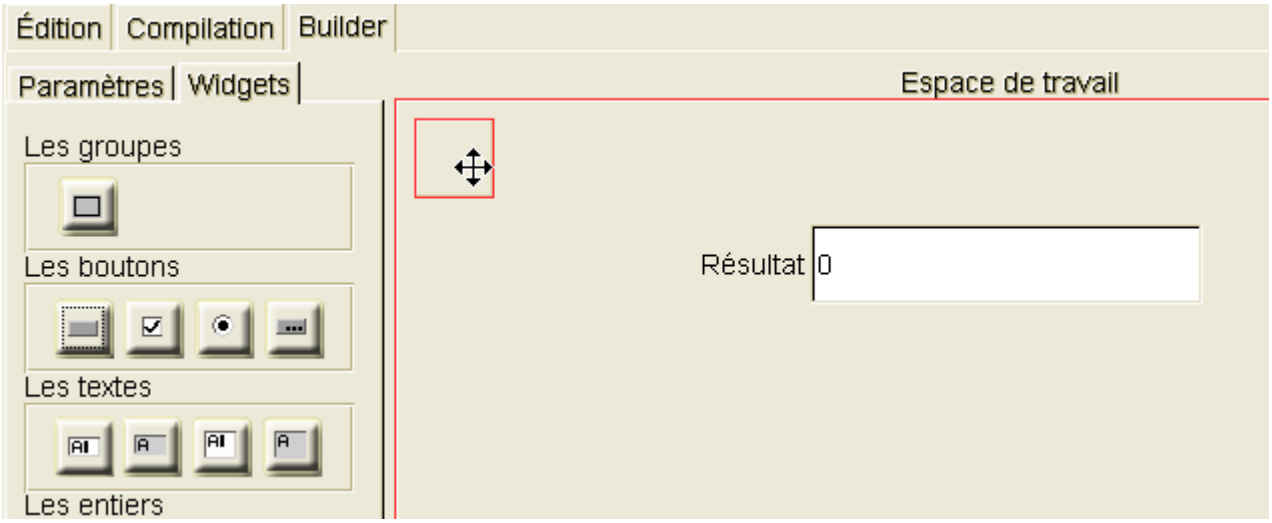

Figure 57 : Ajouter le premier bouton à la calculatrice.

Nous allons ensuite le déplacer et lui donner les paramètres nécessaires pour l'accéder après.

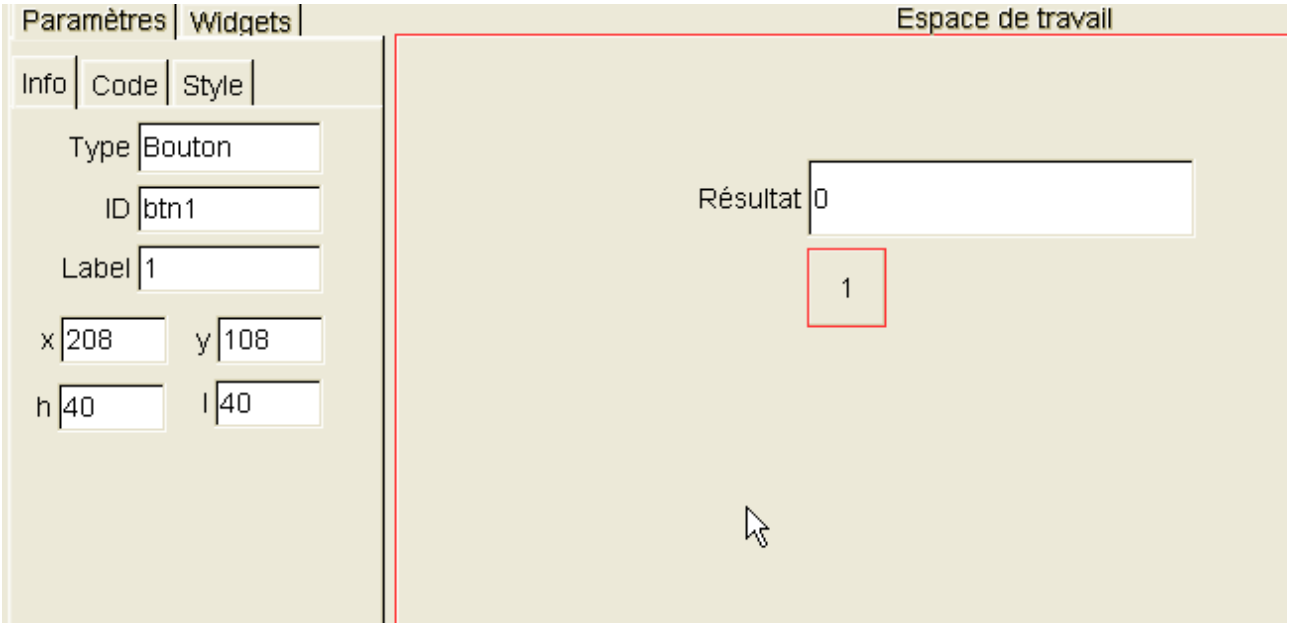

Figure 58 : Mise en place du bouton « 1 ».

Nous allons faire la même chose avec les autres boutons des chiffres.

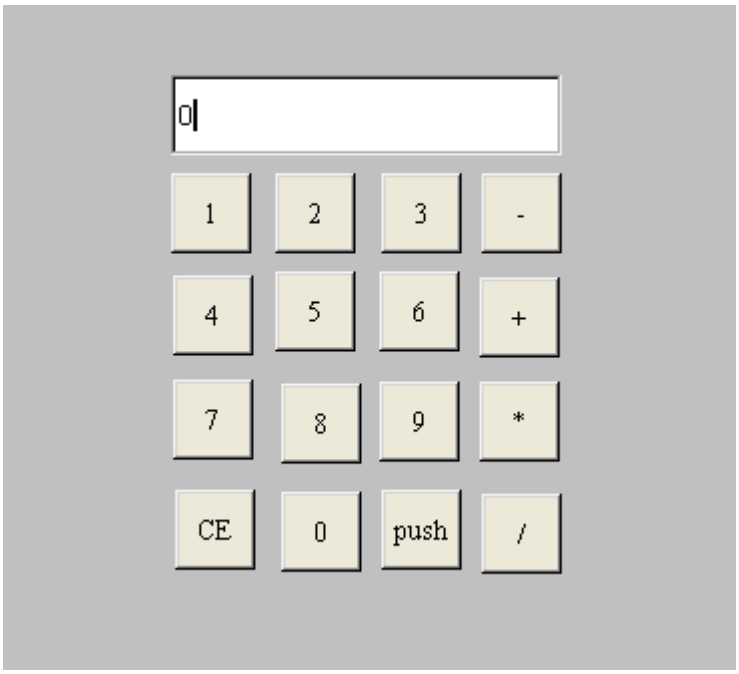

Figure 59 : Vue générale de la calculatrice.

Nous avons fini de construire notre calculatrice. Nous avons mis des identificateurs pour tous les widgets.

Nous allons maintenant implémenter le comportement des widgets dans la calculatrice. Nous avons besoin d'un variable globale, pour stocker les résultats intermédiaires. Nous allons auparavant débuter par l'implémentation des boutons.

Nous allons nous basé sur la librairie TinyLib pour implémenter nos widgets. Nous avons dans cette librairie toutes les fonctions nécessaires à l'implémentation des widgets.

ErreurType	**setEntier** (const string &nomWidget, int entier) *affecter la valeur entiere*
ErreurType	**setEntier** (const char *nomWidget, int entier) *affecter la valeur entiere*
ErreurType	**setEntier** (int entier) *affecter la valeur entiere au widget courant*
ErreurType	**getEntier** (const string &nomWidget, int &entier) *retourner la valeur entiere*
ErreurType	**getEntier** (const char *nomWidget, int *entier) *retourner la valeur entiere*
ErreurType	**getEntier** (int *entier) *retourner la valeur entiere du widget courant*

Figure 60 : Quelques fonctions de la librairie TinyLib.

Principe de fonctionnement :

Nous allons utiliser une pile pour stocker les valeurs intermédiaires ainsi que les résultats. Dans notre cas, nous allons utiliser la classe vector pour gérer les données.

Nous allons commencer par ajouter la librairie vector. Cette déclaration doit être dans le code général de l'application. Autrement dit, dans le code de la fenêtre principale.

```
#include <vector>
Using std ::vector ;
Vector <int> pile ;
```

Figure 61 : Le code de la déclaration du vecteur.

Dans la figure ci-dessus, nous avons déclaré une pile qui va être utilisé pour les traitements futurs des données.

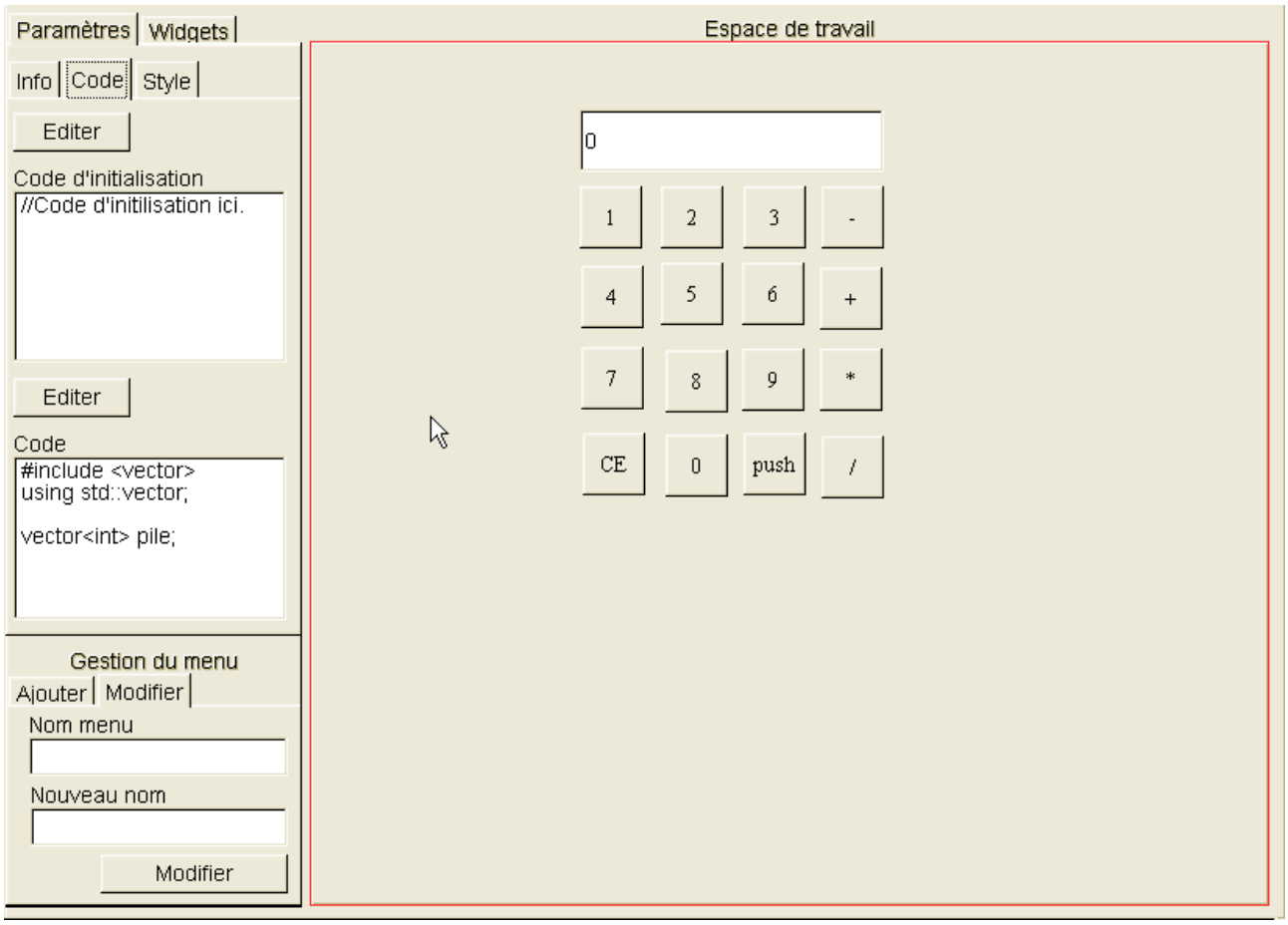

Figure 62 : Vue de la calculatrice ; ajout du code d'initialisation.

Passons maintenant à l'initialisation de l'écran de la calculatrice sur la valeur « 0 ». Sélectionnez le widget écran de la calculatrice et ajoutez la ligne dans la figure 23 dans son code d'initialisation.

```
setEntier(0);
```

Figure 63 : initialisation du widget écran de la calculatrice.

Voici un aperçue :

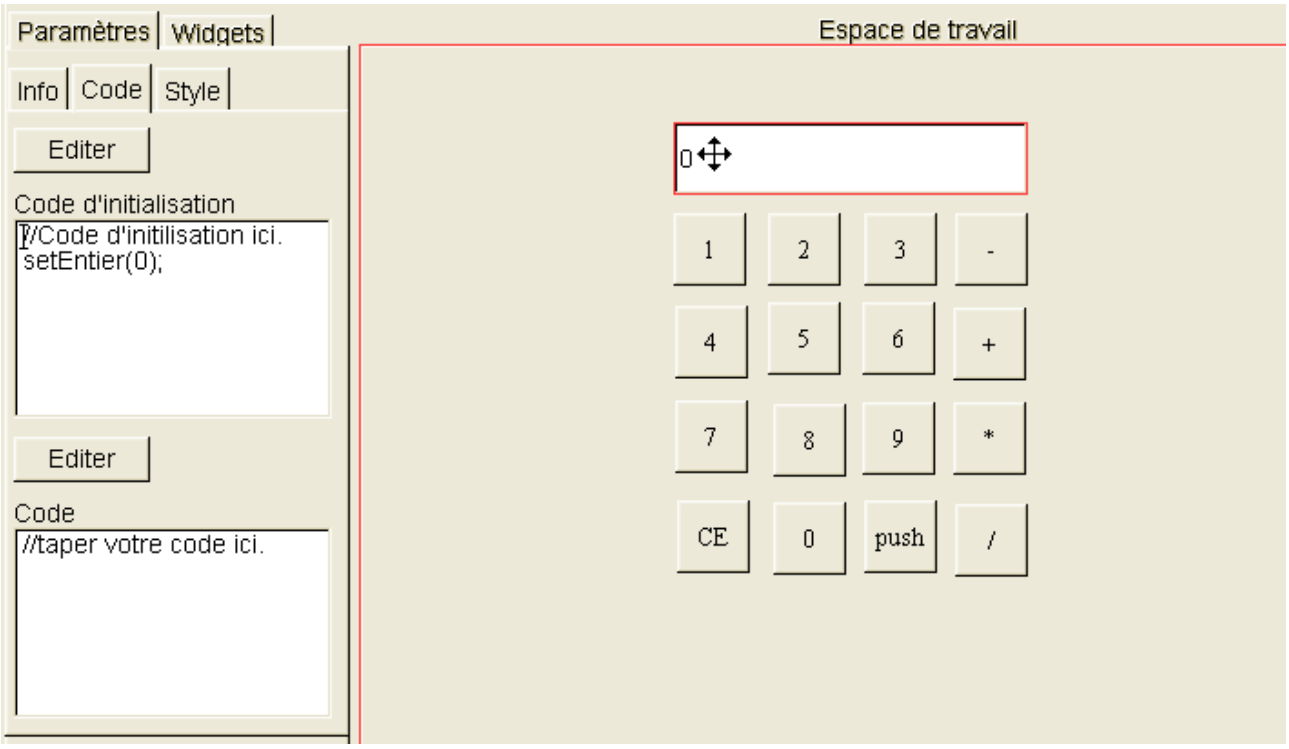

Figure 64 : Ajout du code d'initialisation du widget de l'entrée entière.

Nous allons maintenant passé à l'implémentation du comportement des boutons des chiffres. Il nous faut tout d'abord gérer le problème de concaténation des chiffres pour avoir des nombres. Nous allons déclarer une variable pour récupérer le contenu de l'écran et le concaténer avec la nouvelle valeur. Par exemple pour taper le nombre « 12 », l'utilisateur va taper le chiffre « 1 » ensuite « 2 ». Pour concaténer les deux chiffres, nous allons multiplier le premier chiffre par « 10 » et lui additionner le deuxième chiffre.

Pour finir, il faut afficher le nombre dans l'écran. Voici le code à ajouter dans le premier bouton.

```
if (etatBouton())
        return;
int decalage ;
getEntier("ecranResultat",decalage);
setEntier("ecranResultat",1+(decalage* 10));
```

Figure 65 : Code d'un bouton chiffre (bouton 1).

Pour le reste des boutons chiffres, il suffit de copier le même code en changent la valeur 1 par la valeur du bouton à traiter.

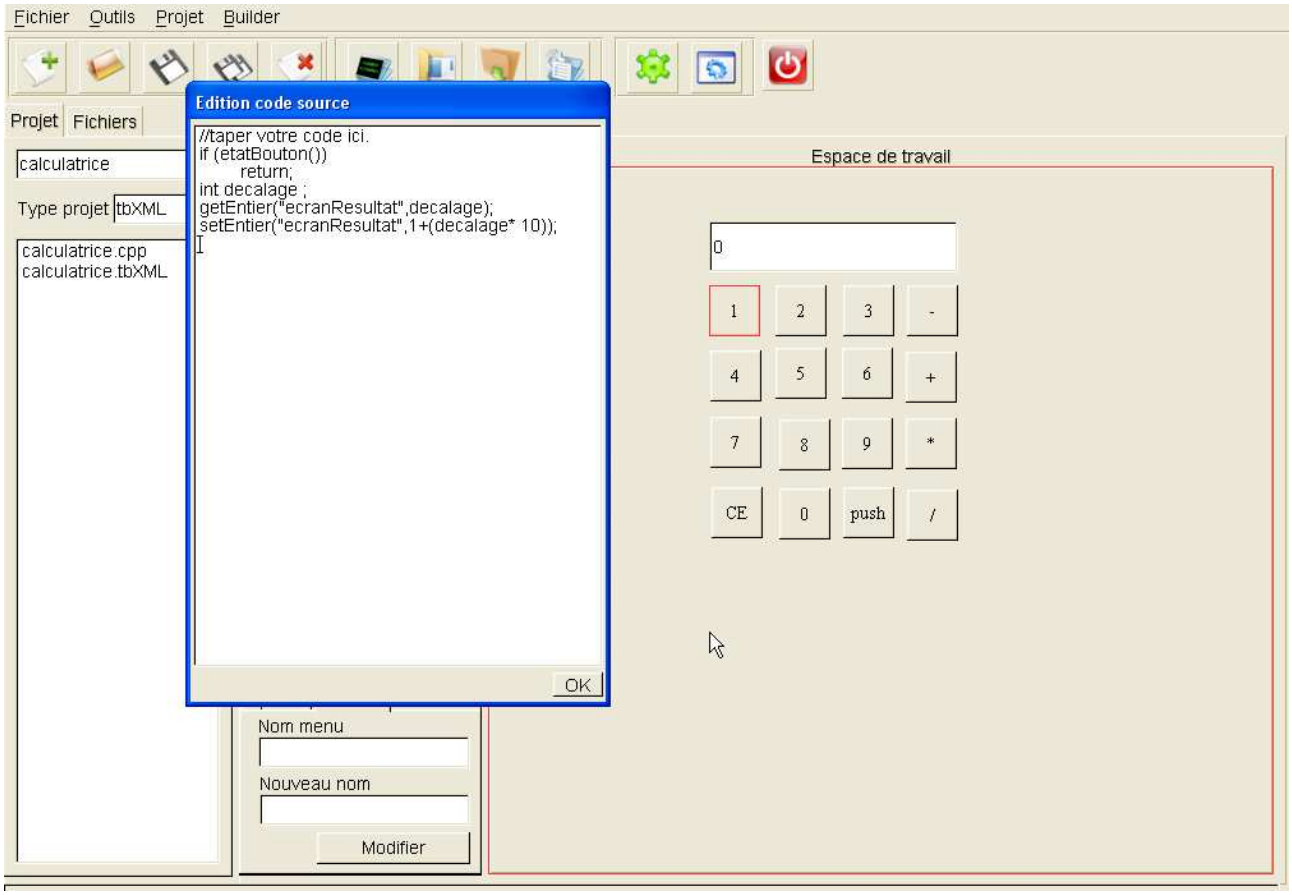

Figure 66 : Édition du code source du bouton.

En ce qui concerne les boutons opérateurs, nous allons vérifier qu'il y a au moins deux valeurs dans la pile sinon on affiche un message d'erreur. Ensuite, il faut dépiler le dernier élément de la pile et le suivant, et finalement calculer le résultat. Voici le code à mettre dans la partie code de chaque bouton opérateur en changent l'opérateur dans le calcul du résultat (voir figure 30).

```
if (etatBouton())
        return;
if(pile.size()<2){
 messageModal("Opération impossible. Il faut entrer deux valeurs.");
 return;
}
```

Figure 70 : Ignorer le double évènement du bouton et vérifier la taille de la pile.

```
int parametreGauche = pile.back();
pile.pop_back();
int parametreDroite = pile.back();
```

Figure 71 : Récupérer les valeurs depuis la pile

```
int resultat = parametreGauche - parametreDroite;
```

Figure 72 : Calcule du résultat (changer le «-» selon l'opérateur).

```
setEntier("ecranResultat",resultat);
```

Figure 73 : afficher le résultat dans l'écran de la calculatrice.

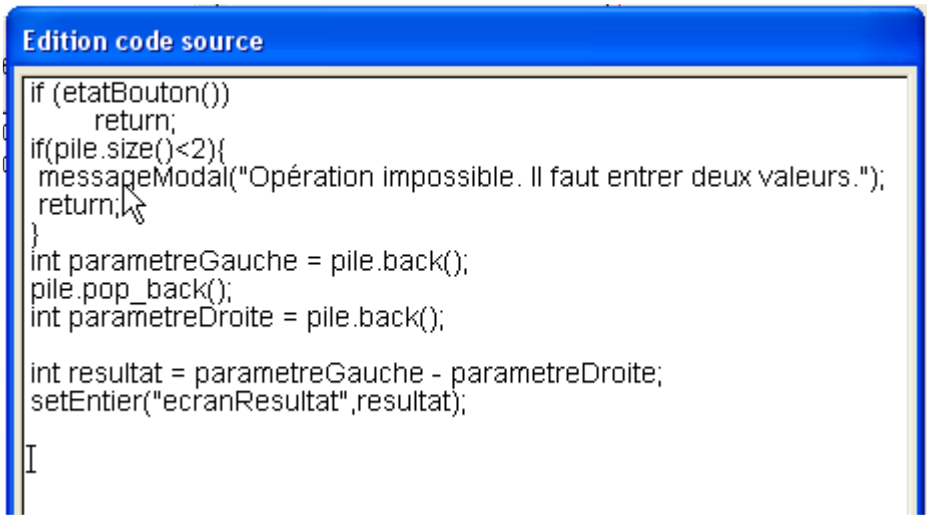

Figure 74 : Capture de l'éditeur du code source.

Le bouton « push » : Ce bouton va servir à empiler les informations. À chaque fois que l'utilisateur tape son nombre, il a besoin de l'empiler. Le code de ce bouton, consiste à récupérer le contenu de l'écran et l'empiler, ensuite il faut initialiser l'écran sur la valeur « 0 ».

```
if (etatBouton())
        return;
int valeur;
getEntier("ecranResultat", valeur);
pile.push_back(valeur);
setEntier("ecranResultat",0);
```

Figure 75 : code source du bouton « push ».

Il nous reste le bouton « CE » qui va nous permettre d'initialiser l'état de la pile ainsi que l'écran de la calculatrice. Voici le code de ce bouton :

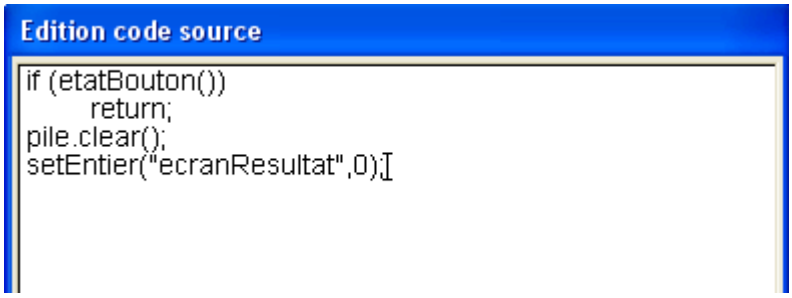

Figure 76 : code source du bouton « CE ».

Nous avons fini la réalisation de notre calculatrice, nous pourrons maintenant compiler et exécuter notre application depuis les raccourcis de compilation.

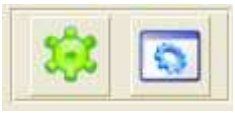

Figure 77 : raccourcis de compilation et exécuter.