« pas un projet banal… c'est presque comme un enfant »

Entretien avec Gábor Kiss-Vámosi, le père de LittlevGL

Questions : **Mathias Claußen** & **Jens Nickel (Elektor)** – Réponses : **Gábor Kiss-Vámosi**

Le développement de logiciels ne saurait se passer de bibliothèques et d'infrastructures (*frameworks*) à code source ouvert. Certaines de ces bibliothèques prennent forme quand de jeunes passionnés de logiciels privés produisent du code dont ils ont régulièrement besoin pour eux-mêmes. Peu à peu leur projet prend de l'ampleur et devient professionnel, comme c'est le cas de la bibliothèque *LittlevGL* de Gábor Kiss-Vámosi, qui met les interfaces graphiques tactiles à la portée des microcontrôleurs à 16 bits pauvres en RAM (il y a un autre article sur ce sujet dans ce numéro). Dans cet entretien, Gábor parle du contexte, de quelques détails intéressants et de l'avenir de sa bibliothèque populaire.

Elektor : Gábor, pourquoi devrions-nous utiliser votre bibliothèque d'interfaces graphiques alors qu'il y en existe déjà qui sont libres à des fins non commerciales comme µGFX ?

Gábor : µGFX et LittlevGL ont des avantages communs, comme l'indépendance du code source ouvert par rapport à la plate-forme, mais LittlevGL gère aussi les animations lisses, l'anti-crénelage, l'opacité et les ombre portées sans double tampon. De sorte qu'avec LittlevGL une seule trame tampon suffit dans la puce du contrôleur d'affichage ou dans le µC et une petite RAM graphique. Un autre avantage est le support intégré dans LittlevGL pour la navigation et la commande par clavier ou avec un encodeur.

Et puis LittlevGL supporte MicroPython. Vous pouvez donc écrire du code Python3 pour créer une interface utilisateur. Ainsi vous pouvez modifier l'interface utilisateur en temps réel sans reconstruire ni flasher le µC.

Elektor : Quelle est votre formation, votre profession ? Pourquoi investissez-vous tant de temps dans un projet logiciel ?

Gábor : Je suis ingénieur électricien et travaille sur un projet lié à la sécurité du matériel informatique dans le cadre de ma profession.

Pendant mes études universitaires, un ami et moi avons passé beaucoup de temps en amateurs sur des amplis, des instruments et d'autres gadgets que nous souhaitions équiper d'un bel écran graphique au lieu des petits LCD habituels. J'ai proposé d'acheter un bel écran TFT pour essayer de dessiner quelque chose. Si je parvenais à programmer un pixel, le reste suivrait facilement. Au bout de deux heures je commandais mon premier pixel, et la suite dure depuis huit ans !

J'aime apprendre. Ce travail est donc comme un passe-temps. Une fois le projet devenu public, j'ai reçu beaucoup de commentaires d'autres développeurs, plus expérimentés. Ils m'ont suggéré des améliorations. J'ai relevé le défi, car je me sens personnellement responsable de mon projet, puisque que des gens l'utilisent et ont confiance en lui.

Elektor : Combien de temps consacrez-vous réellement au codage ? Combien de temps consacrez-vous à d'autres tâches du projet ?

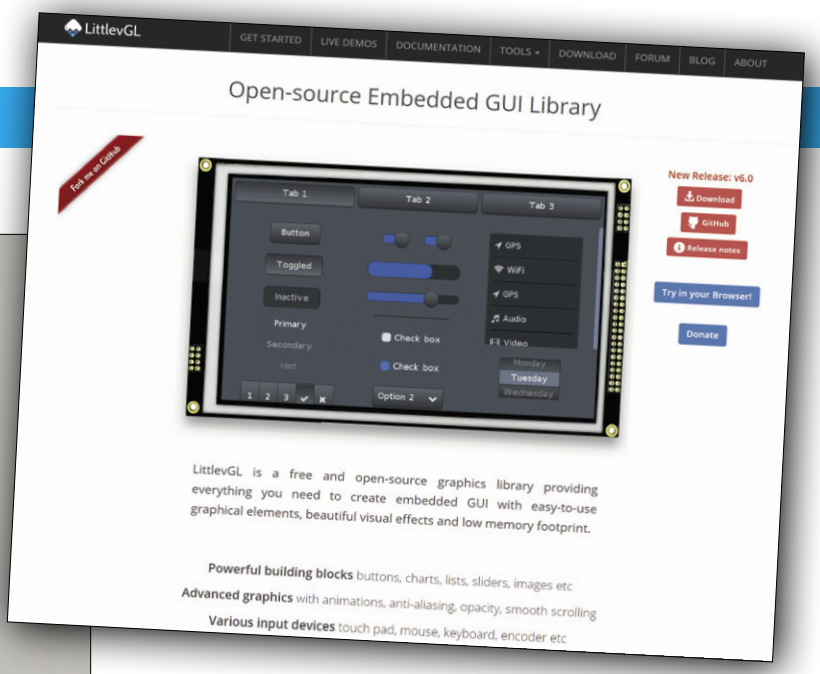
Gábor : C'est devenu moins facile depuis que j'ai une famille, je dois trouver l'équilibre. D'habitude, je me lève tôt, quand ma femme et mon enfant dorment encore, pour répondre aux courriels et aux questions et pour travailler sur les nouvelles fonctions. L'après-midi, si j'ai du temps libre, j'en profite pour essayer d'avancer. Si je compte mes heures, c'est un temps partiel en plus de mon plein temps professionnel. Combiner le travail, LittlevGL, la famille, les amis et les loisirs n'est pas une mince affaire, mais j'y arrive.

Au début, quand j'ai publié LittlevGL, j'avais beaucoup à apprendre sur des choses qui n'ont rien à voir avec la programmation, comme le marketing, l'optimisation pour les moteurs de recherche, le développement web, un peu de conception graphique, etc. Il m'a fallu des mois pour maîtriser les astuces de SEO, et pour ça je passais des journées à analyser des sites. Ces efforts ont porté leurs fruits : recherchez «*embedded GUI*« dans Google, c'est LittlevGL qui sera probablement le premier résultat.

Elektor : Obtenez-vous de l'aide de la communauté ou des entreprises en arrière-plan, pour le codage et d'autres tâches ?

Gábor : Heureusement, au fur et à mesure, de plus en plus de contributeurs se sont joints au projet pour le soutenir. Cer-

Gábor Kiss-Vámosi (29 ans) vit à Budapest (Photos : Ramóna Erdei).



tains d'entre eux partagent leurs améliorations personnelles comme «Hey, j'ai ajouté une fonction de substitut (*placeholder*) à l'objet Zone de texte. Ça vous intéresse ?» Les améliorations proposées par certains sont énormes ; la liaison MicroPython est de «amirgon» et le nouveau convertisseur de polices est de «puzrin». D'autres aident à répondre aux questions des utilisateurs et à corriger des bogues. Par exemple, «embeddedt» passe des heures chaque jour à aider les utilisateurs.
Ce qu'ils font est très important et précieux pour LittlevGL qui sans eux ne serait pas ce qu'il est !

Elektor : Avez-vous obtenu de l'aide de la communauté pour le portage vers Arduino ?
Gábor : En fait, le portage vers Arduino a été délicat. Il a fallu modifier la structure de la bibliothèque pour qu'elle fonctionne avec le compilateur d'Arduino. J'ai fait les premiers changements moi-même mais il y a quelques mois, «Pablo2048» l'a remanié pour sortir une version Arduino. Il m'aide aussi à répondre aux questions connexes, son expérience avec Arduino est plus vaste que la mienne**.**

Elektor : Les utilisateurs peuvent donner de l'argent à votre projet. Les dons sont-ils substantiels pour vous ou n'est-ce qu'un petit soutien ?
Gábor : Les dons sont suffisants pour couvrir les dépenses du projet telles que le serveur, l'achat de cartes de développement et ainsi de suite, mais je ne peux pas en vivre.

Elektor : Comment trouvez-vous de nouvelles idées pour la bibliothèque ? Recevez-vous les commentaires des développeurs ?
Gábor : Parfois, les utilisateurs ont des requêtes explicites. Je traite les questions récurrentes pour améliorer ce qui doit l'être dans la bibliothèque ou dans la documentation. Ainsi la version 6.0 a-t-elle été un changement conceptuel dans la bibliothèque principalement pour résoudre les questions fréquentes.
Outre les suggestions des utilisateurs, j'essaie de suivre les tendances générales en matière de conception graphique et d'interface utilisateur intégrée et de proposer des solutions. J'utilise la bibliothèque moi-même. J'ai réalisé plusieurs projets pour des entreprises en tant que pigiste. Je l'utilise aussi dans mon propre travail professionnel en ce moment.

Elektor : Fréquentez-vous les salons et autres événements ? Avez-vous l'occasion de parler aux utilisateurs de votre bibliothèque ?
Gábor : Ah, les échanges personnels avec les utilisateurs me manquent. Ce serait sans doute enrichissant de les rencontrer, mais ils sont si disséminés….
Certains m'ont invité, plusieurs m'ont offert de nous accueillir si nous visitons leur pays ou leur ville. Nos vacances de l'année prochaine ont été organisées en fonction de la rencontre avec un utilisateur de LittlevGL.
L'arrière-plan de LittlevGL va devenir celui d'un vrai logiciel professionnel et pas seulement d'un projet comme un autre. Dans ce cadre, il est prévu de participer à des conférences et d'organiser des webinaires et des séminaires.

Elektor : Pour utiliser LittlevGL, des µC à 16, 32 ou 64 bits sont nécessaires. Cela implique-t-il l'abandon des 8 bits ?
Gábor : Théoriquement, il n'y a pas de problème avec les 8 bits mais leur RAM est généralement insuffisante pour exécuter LittlevGL. Il faut au moins 16 Ko pour LittlevGL, ce qui est beaucoup demander, même aux µC à 16 bits.

Elektor : Pouvez-vous nous parler de l'idée technique de la bibliothèque ? On voit que l'interface utilisateur fonctionne avec des écrans / scènes…
Gábor : L'idée de base est de créer vos objets (appelés *widgets*) tels que boutons, étiquettes, graphiques, curseurs, etc., et dont vous définissez simplement les propriétés telles que la taille, la position, les styles, la valeur ou l'état et les rappels pour informer l'utilisateur si un événement survient (p. ex. un

Outils en ligne pour un projet logiciel

Dans un projet de logiciel ouvert, le codage est essentiel, mais il vous faut aussi une documentation solide et un solide soutien pour les utilisateurs. C'est ce que Gábor Kiss-Vámosi raconte à propos de son site et d'autres outils pour soutenir sa bibliothèque :

Comme ingénieur électricien, je ne connaissais pas grand-chose au développement web. J'ai essayé WordPress mais quand j'ai essayé de le personnaliser, j'ai découvert qu'il ne me donnerait pas la liberté souhaitée. Puis j'ai trouvé Bootstrap, et j'ai pu créer le site à partir de zéro. Bootstrap est assez facile, même pour un novice, et à la fois beau et rapide.
Par la suite, les «sujets secondaires» ont migré vers des sous-domaines distincts et maintenant ils fonctionnent avec des moteurs différents.
Le blog est hébergé sur GitHub où les messages sont écrits en Markdown (balisage léger), et un moteur appelé Jekyll compile automatiquement un site web statique à partir des fichiers Markdown. De cette façon, n'importe qui peut facilement ajouter des messages sur le blog, il suffit d'envoyer une requête *pull*.
La documentation est également hébergée sur GitHub mais elle est compilée hors ligne avec Sphinx. Les documents peuvent être traduits par les utilisateurs sur Zanata, une plate-forme en ligne.
Le forum dispose d'un moteur Discourse qui fonctionne sur un serveur virtuel DigitalOcean. DigitalOcean a un programme de parrainage pour les projets à code source ouvert. J'ai demandé et obtenu un parrainage annuel, que je peux dépenser comme je l'entends. J'ai donc lancé un VPS pour héberger le forum. L'aide de «seyyyah» de GitHub a été précieuse pour la configuration du VPS.
Mon but était de bâtir une communauté où les gens partagent leurs connaissances, parlent de leurs projets et de leurs expériences. LittlevGL a un blog ouvert où tout le monde peut écrire des messages. Sur le forum, il y a une catégorie «Mes projets» pour partager ce que vous avez créé d'intéressant.

Elektor : Nous pouvons utiliser différents widgets préfabriqués pour l'interface utilisateur, mais si nous avons besoin de quelque chose qui n'existe pas, y a-t-il un guide pour nous aider à les créer ?
Gábor : Tous les widgets fonctionnent de la même manière. Ils ont des données personnalisées, une fonction de conception qui dessine le widget et une fonction de signal qui gère les événements internes de bas niveau. Au fond, si vous examinez le code du widget, vous pouvez créer le vôtre de la même manière. Il existe des fichiers modèles c et h pour faciliter le démarrage avec votre propre widget.

Elektor : L'allocation de RAM est-elle statique pour les éléments de l'interface utilisateur ? Ou la réserve de mémoire est-elle limitée, au risque de ne pas suffire ?
Gábor : LittlevGL a son propre gestionnaire de mémoire avec allocation dynamique de la mémoire pour les widgets et éléments connexes. Il alloue les données dans un grand tableau dont la taille peut être ajustée ou même placée dans une mémoire externe. Contrairement aux standards «malloc» et «free», le gestionnaire de mémoire de LittlevGL permet de surveiller l'utilisation et la fragmentation de la mémoire. Donc oui, la mémoire est limitée, mais vous pouvez surveiller son occupation et ajuster la taille de la mémoire réservée à LittlevGL. Si ça coince, un message d'erreur avec un numéro de ligne s'affiche et le programme s'arrête là. Ainsi, vous décelez facilement l'origine du problème.

Elektor : Comment le dessin est-il géré ? Y a-t-il des astuces pour réduire le dessin multiple sur des objets qui se chevauchent ?
Gábor : De nombreuses astuces contribuent à optimiser le dessin. Au niveau de dessin le plus élevé, LittlevGL accumule les zones à redessiner (p. ex. si un bouton est pressé) et toutes les quelques millisecondes (p. ex. 30), il redessine ces zones. Avant de redessiner, il recherche à partir de l'arrière-plan quel widget est le premier qui couvre la zone à redessiner. Ainsi, si vous modifiez le texte du bouton, seuls le bouton et l'étiquette seront redessinés mais pas le fond sous le bouton. Mais si vous changez la position du bouton, le fond, le bouton et l'étiquette devront également être dessinés.

clic de bouton). De cette façon, tous les dessins et autres éléments sous-jacents sont gérés sous le capot par LittlevGL et l'utilisateur n'a à s'occuper que des éléments de haut niveau. L'objet peut être créé et supprimé en temps réel. Ce qui permet d'optimiser l'utilisation de la mémoire pour ne garder en vie que l'objet requis. Vous avez besoin d'une boîte de message par exemple ? Créez-la puis supprimez-la dès que l'utilisateur clique sur le bouton «Ok». La boîte de message n'a consommé de mémoire que tant qu'elle était affichée.
Les objets ont une hiérarchie parents-enfants. Vous pouvez p. ex. créer un conteneur (parent) et y ajouter trois boutons (enfants). Si vous déplacez le conteneur, les boutons se déplaceront avec, si vous supprimez le conteneur, les boutons seront également supprimés.
Un autre atout de LittlevGL est de ne réaliser qu'un type limité d'orientation d'objet. Chaque objet est dérivé de l'objet de base doté seulement des propriétés les plus communes comme la position, la taille, le parent, etc. Ce mécanisme assez commun en C++ est unique en C.

Le développement commence dès l'aube. Avant même le petit-déjeuner !

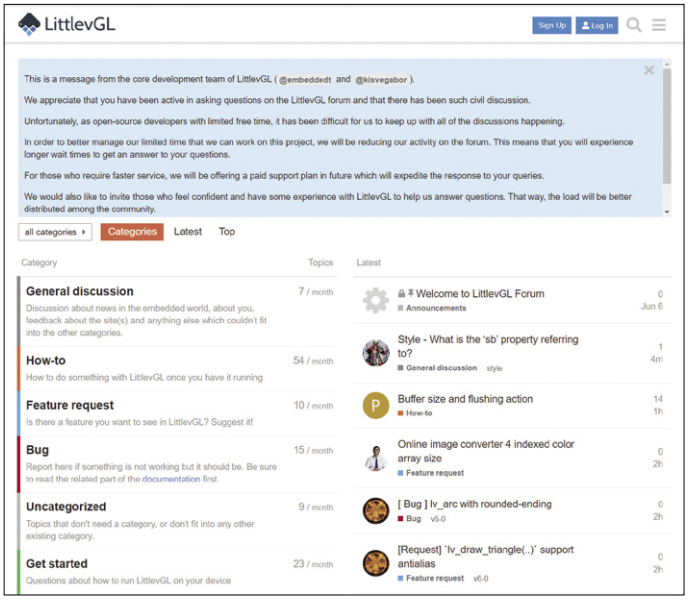
Ce sont-là des optimisations assez banales.
Ce qui est vraiment intéressant dans LittlevGL, c'est qu'il ne dessine pas directement pour l'affichage (comme beaucoup de bibliothèques graphiques simples) ni ne fait de double tampon (comme les bibliothèques graphiques avancées) mais dessine en mosaïque. Il utilise une mémoire plus petite (généralement 1/10e de la taille d'un écran) et y dessine en premier. Lorsque le dessin est prêt, le tampon avec l'image complète est balancé à l'écran d'un coup. Comme le dessin est fait dans le tampon, il n'y a pas de scintillement, les animations sont fluides.
Dans la dernière version de la bibliothèque (qui sera publiée cette année), le moteur de dessin est complètement réécrit. Le moteur de dessin est plus flexible et extensible, la qualité générale est meilleure (amélioration de l'anticrénelage et des ombres p. ex.), mais l'objectif principal était de supporter le masquage. Grâce au masquage des pixels, les coins arrondis peuvent être découpés ou les textes peuvent être écrits avec un fond d'image ou une couleur de dégradé. Sans oublier de nouvelles fonctions comme le gradient horizontal, le décalage de l'ombre et les modes de mélange additif et soustractif.

Elektor : Les polices peuvent être converties en ligne sur votre page ; offrez-vous un convertisseur hors ligne facile à utiliser pour les polices ? Et les symboles ?
Gábor : Le convertisseur de polices est écrit en Node.js, il peut donc fonctionner en ligne et hors ligne aussi. Nous n'avons pas encore d'application de conversion de polices de bureau, mais vous pouvez cloner le convertisseur depuis GitHub et l'utiliser en ligne de commande.
Les symboles peuvent aussi provenir de polices de caractères. *FontAwesome* est une police de caractères bien connue et très populaire dont certains symboles sont inclus par défaut dans LittlevGL. Cependant, lorsque vous créez votre propre police, vous pouvez choisir plusieurs fichiers de polices pour les fusionner en un seul fichier C, en combinant par exemple certaines plages de *Arial* à quelques symboles de *FontAwesome*.

Elektor : La bibliothèque n'est pas sécurisée, est-il prévu d'ajouter le support RTOS à votre bibliothèque ?
Gábor : C'est vrai, elle n'est vraiment pas sécurisée par défaut mais il est facile de la sécuriser. Il suffit de prendre un *mutex* lorsque vous appelez les fonctions liées à LittlevGL et de le libérer dès que c'est fini.

Elektor : LittlevGL dispose d'un environnement basé sur Eclipse pour un développement rapide de l'UI sans avoir à flasher de code sur du matériel tangible. Y aura-t-il une sorte de GUI Designer, pour accélérer encore le développement ?
Gábor : Oui. Ça a déjà été demandé. Cela me comble que certains utilisateurs aient commencé à travailler spontanément sur des concepteurs d'interfaces graphiques. Il existe trois développements indépendants et ils ont tous beaucoup progressé. L'un d'eux pourrait devenir le concepteur officiel de l'interface graphique de LittlevGL.

Elektor : LittlevGL est sous licence MIT, ce qui en facilite l'intégration dans les produits commerciaux. D'après le dernier sondage en ligne effectué, il y avait des questions sur les options de licence commerciale. Qu'est-ce qui va changer ?



Le support aux utilisateurs représente une grande part du travail. Sur le forum, des utilisateurs expérimentés de la bibliothèque répondent également aux questions.

Gábor : La popularité de LittlevGL augmente rapidement. Une très grande entreprise s'est montrée intéressée par LittlevGL, mais il a été très difficile de négocier à titre personnel. La prochaine étape raisonnable est de créer une entreprise qui puisse mieux défendre les intérêts de LittlevGL.
À mesure que la communauté grandit, le développement, le soutien et le marketing sont de moins en moins une activité de temps libre. Il faut pour LittlevGL un modèle économique qui reste acceptable pour les utilisateurs mais permettrait aussi de salarier certaines personnes qui travaillent sur LittlevGL à plein temps. Un support payant avec correction rapide de bogues et aide professionnelle dédiée, ainsi qu'une sorte de licence sont également envisagés. Il reste à trouver le bon modèle économique.

Elektor : Quels conseils donneriez-vous à des jeunes qui développent des projets de logiciels libres intéressants ?
Gábor : Lorsque vous publiez ou démarrez un projet à code source ouvert, ayez un plan, une vision. S'agit-il d'un petit outil qui se passe d'entretien ? Ou de quelque chose qui demandera beaucoup de temps pendant des années ? Quel est votre objectif ? Pourquoi faire ça ? Juste pour le plaisir ? Pour la gloire ? Pour l'expérience ? Pour aider les autres ? Il faut répondre à ces questions avant de se lancer et prendre vos décisions en conséquence. La frustration vous guette si vous laissez les choses arriver sans que vous sachiez comment prendre quelle décision. Sachez dire non à ce qui ne correspond pas à ce que vous voulez et restez ouvert aux occasions favorables qui vous rapprochent de vos objectifs. Comme dit le proverbe, *il n'y a pas de vent favorable pour le marin qui ignore où est son port*.

190353-02

Lien

[1] Page d'accueil de LittlevGL : https://littlevgl.com/