

Avant-propos

Notre culture de consommation nous offre toutes sortes d'opportunités de divertissement, de plaisir et parfois même d'apprentissage. Cependant, dans l'ensemble, il s'agit d'activités passives. Ce n'est pas grave – nous aimons tous nous détendre de temps en temps et nous divertir – mais cela ne devrait pas être une vision globale. À l'attrait de consommer s'ajoute la satisfaction de produire, c'est-à-dire de créer. C'est la joie et la fierté qui résultent lorsque nous faisons un dessin, construisons un modèle réduit d'avion ou cuisons du pain.

Les objets de haute technologie (comme les téléphones portables, les tablettes, les téléviseurs, etc.) que nous utilisons aujourd'hui pour consommer des divertissements et des informations sont des boîtes noires pour la plupart d'entre nous. Leur fonctionnement est incompréhensible et, même si certains d'entre eux possèdent des capacités permettant à l'utilisateur de dessiner des images, de réaliser des vidéos, etc., ils ne sont pas, en soi, des supports de création. En d'autres termes, la plupart des gens ne peuvent pas créer les applications qui s'exécutent sur ces gadgets.

Et si nous pouvions changer cela ? Et si nous pouvions prendre le contrôle créatif de nos gadgets quotidiens, comme les téléphones portables ? Et si créer une application pour votre téléphone portable était aussi simple que de dessiner une image ou de préparer une miche de pain ? Et si nous pouvions réduire le fossé entre les objets de notre culture de consommation et les médias de nos vies créatives ?

D'une part, cela pourrait démystifier ces objets. Plutôt que d'être des boîtes noires, impénétrables à notre vue, elles deviennent des objets bricolables. Ils deviennent des objets capables de notre compréhension. Nous entretenons une relation moins passive et plus créative avec eux, et nous pouvons jouer avec ces appareils d'une manière beaucoup plus profonde et plus significative lorsque nous pouvons réellement construire des choses pour eux.

Lorsque Hal Abelson m'a parlé pour la première fois de l'idée qui est devenue App Inventor, nous avons parlé de la force de motivation unique que les téléphones portables pourraient avoir dans l'éducation.

Il s'est demandé si nous pouvions utiliser cette force motivante pour aider à initier les étudiants aux concepts de l'informatique. Au fur et à mesure que nous l'avons construit et essayé dans des classes comme celle de Dave Wolber, nous avons commencé à réaliser que quelque chose d'encore plus puissant se produisait : App Inventor commençait à transformer les étudiants de consommateurs en créateurs. Les étudiants ont trouvé amusant et exaltant de créer des applications pour leurs téléphones ! Lorsqu'un des étudiants de Dave a créé l'application simple mais puissante No Texting While Driving, nous avons vraiment commencé à imaginer ce qui se passerait si quelqu'un, et pas seulement des ingénieurs logiciels professionnels, pouvait créer une application.

Chez Google, nous avons donc travaillé dur pour rendre App Inventor plus simple, plus amusant à utiliser et toujours plus puissant. Hal et son incroyable équipe du MIT ont pris la relève en 2012 et ont continué à améliorer l'expérience des débutants et des développeurs. La nouvelle version, décrite dans ce livre et communément appelée App Inventor 2, offre une expérience entièrement intégrée au navigateur qui peut vous transformer en créateur d'application en quelques minutes !

Les auteurs de ce livre sont de véritables éducateurs et ingénieurs logiciels de classe mondiale. Je tiens à les remercier personnellement pour leur travail de création, de test et de documentation d'App Inventor et, bien sûr, pour avoir écrit ce merveilleux livre.

Maintenant, libérez votre créativité et créez une application !

—Mark Friedman, responsable technique et responsable du projet App Inventor pour Android,

Google

Préface

Vous êtes sur votre itinéraire de course habituel, vous faites juste du jogging, et une idée pour la prochaine application mobile qui tue vous vient. Sur le chemin du retour, vous ne vous souciez même pas de votre temps, tout ce à quoi vous pouvez penser, c'est de faire connaître votre idée. Mais comment faire exactement ? Vous n'êtes pas un programmeur, et cela prendrait des années, et le temps, c'est de l'argent, et... eh bien, quelqu'un l'a probablement déjà fait de toute façon. Juste comme ça, votre idée est morte dans l'eau.

Imaginez maintenant un monde différent, où la création d'applications ne nécessite pas des années d'expérience en programmation, où les artistes, les scientifiques, les humanitaires, les travailleurs de la santé, les avocats, les combattants, les marathoniens, les entraîneurs de football et les personnes de tous horizons peuvent créer des applications. . Imaginez un monde où vous pouvez transformer des idées en prototypes sans embaucher de programmeurs, où vous pouvez créer des applications qui fonctionnent spécifiquement pour vous, où vous pouvez adapter l'informatique mobile à vos besoins personnels.

C'est le monde d'App Inventor, un outil de programmation visuelle pour créer des applications mobiles. Basé sur une méthode de programmation par « blocs » visuels qui a fait ses preuves même auprès des enfants, App Inventor réduit considérablement les obstacles à la création d'applications pour les téléphones et appareils Android. Que diriez-vous d'un jeu vidéo où les personnages ressemblent à vous et à vos amis ? Ou un « as-tu ramassé le lait ? une application qui te rappelle s'il est plus de 15h et que tu es près de l'épicerie ? Ou une application de quiz que vous offrez à votre proche et qui est en fait une demande en mariage surprise ? « Question 4 : Veux-tu m'épouser ? Appuyez sur le bouton pour accepter en envoyant un message texte. Quelqu'un a vraiment créé une application App Inventor pour proposer un mariage comme celui-ci, et elle a dit oui !

Un langage de blocage pour les téléphones mobiles

App Inventor est un outil visuel de glisser-déposer permettant de créer des applications mobiles sur la plate-forme Android. Vous concevez l'interface utilisateur (l'apparence visuelle) d'une application à l'aide d'un générateur d'interface utilisateur graphique (GUI) basé sur le Web, puis vous spécifiez le comportement de l'application en assemblant des « blocs » comme si vous travailliez sur un puzzle.

La figure P-1 montre les blocs d'une première version d'une application créée par Daniel Finnegan, un étudiant universitaire qui n'avait jamais programmé auparavant. Pouvez-vous dire ce que fait l'application ?



Figure 0-1. Une application qui répond automatiquement aux SMS

L'application est un « répondeur » texte. Vous le lancez lorsque vous conduisez et il répond automatiquement aux SMS que vous recevez.

Parce que les blocs sont plus compréhensibles que le code de programmation traditionnel, vous êtes immédiatement attiré et l'utilitaire du monde réel vous amène à poser des questions telles que : puis-je faire en sorte que les textes reçus soient prononcés à haute voix ? Puis-je faire en sorte que la réponse renvoyée puisse être personnalisée ? Puis-je écrire une application qui permet aux gens de voter pour quelque chose par SMS, comme sur American Idol ? La réponse à toutes ces questions est « oui », et dans ce livre, nous allons vous montrer comment.

Que pouvez-vous faire avec App Inventor ?

Plein de trucs !

Jouer

Créer des applications pour votre téléphone est amusant et App Inventor favorise l'exploration et la découverte. Ouvrez simplement App Inventor dans un navigateur Web, connectez votre téléphone et commencez à assembler des blocs comme ceux de l'application illustrée à la figure P-1. Vous pouvez immédiatement voir et interagir avec l'application que vous créez sur le téléphone.

Donc, vous programmez, mais vous envoyez également un e-mail à votre ami pour qu'il vous envoie un SMS pour tester votre application, ou vous contrôlez un robot LEGO NXT avec l'application que vous venez de créer, ou vous débranchez le téléphone et marchez dehors. pour voir si votre application utilise correctement le capteur de localisation.

Prototype

Vous avez une idée d'application ? Au lieu de l'écrire sur une serviette ou de le laisser flotter dans l'éther, construisez un prototype rapide. Les prototypes sont des modèles fonctionnels incomplets et non raffinés de votre idée. Exprimer une idée dans un texte, c'est comme écrire

a à un ami ou à un proche avec de la prose ; considérez un prototype App Inventor comme de la poésie pour un investisseur en capital-risque. De cette manière, App Inventor peut servir de serviette électronique pour le développement d'applications mobiles.

Créer des applications avec une utilité

personnelle Dans l'état actuel du monde des applications mobiles, nous sommes coincés avec les applications qui nous sont proposées. Qui ne s'est pas plaint d'une application et aurait souhaité qu'elle puisse être personnalisée ou ajustée d'une manière ou d'une autre ? Avec App Inventor, vous pouvez créer une application exactement comme vous le souhaitez. Dans le chapitre 3, vous créerez un jeu MoleMash qui vous permettra de marquer des points en touchant une taupe se déplaçant de manière aléatoire. Mais au lieu d'utiliser l'image de la taupe dans le didacticiel, vous pouvez la personnaliser afin d'écraser une photo de votre frère ou de votre sœur, ce que vous seul pourriez vouloir faire, mais peu importe ? Au chapitre 8, vous écrirez une application de quiz qui pose des questions sur les présidents américains, mais vous pouvez facilement la personnaliser pour poser des questions sur n'importe quel sujet de votre choix, de votre musique préférée à votre histoire familiale.

Développer des applications complètes

App Inventor n'est pas seulement un système de prototypage ou un concepteur d'interface : vous pouvez créer des applications complètes à usage général. Le langage fournit tous les éléments fondamentaux de la programmation, tels que les boucles et les conditions, mais sous forme de blocs.

Enseigner et apprendre

Que vous soyez au collège, au lycée ou à l'université, App Inventor est un excellent outil d'enseignement et d'apprentissage. C'est idéal pour l'informatique, mais c'est également un formidable outil pour les mathématiques, la physique, l'entrepreneuriat et à peu près toutes les autres disciplines. La clé est que vous apprenez en créant. Au lieu de mémoriser des formules, vous créez une application pour, par exemple, trouver l'hôpital (ou le centre commercial !) le plus proche. Au lieu d'écrire un essai sur l'histoire des Noirs, vous créez une application de quiz multimédia avec des vidéos et des discours de Martin Luther King, Jr. et Malcolm X. Nous pensons qu'App Inventor et ce livre peuvent être un excellent outil dans les cours tout au long du programme. .

Pourquoi App Inventor fonctionne

La plupart des gens disent qu'App Inventor est facile à utiliser grâce à son interface visuelle par glisser-déposer. mais qu'est ce que ça veut dire? Pourquoi App Inventor est-il si facile à utiliser ?

Vous n'avez pas besoin de mémoriser et de taper des

instructions. L'une des plus grandes sources de frustration pour les programmeurs débutants vient du fait de taper du code et de voir l'ordinateur recracher des messages d'erreur indéchiffrables.

Cette frustration décourage de nombreux débutants de programmer avant même d'avoir réussi à résoudre des problèmes plus amusants et plus logiques.

Vous choisissez parmi un ensemble d'options

Avec App Inventor, les composants et les blocs sont organisés dans des tiroirs facilement accessibles. Vous programmez en recherchant un bloc (qui permet de spécifier la fonctionnalité que vous souhaitez créer) et en le faisant glisser dans le programme. Vous n'avez pas besoin de vous rappeler quelles sont les instructions ou de vous référer à un manuel de programmation.

Seuls certains blocs se connectent les uns aux autres

Au lieu de réprimander les programmeurs avec des messages d'erreur énigmatiques, le langage de blocage d'App Inventor vous empêche de commettre de nombreuses erreurs en premier lieu. Par exemple, si un bloc fonctionnel attend un nombre, vous ne pouvez pas insérer de texte. Cela n'élimine pas toutes les erreurs, mais cela aide certainement.

Vous gérez directement les événements

Les langages de programmation traditionnels ont été conçus lorsque la programmation revenait à travailler avec des recettes ou des ensembles d'instructions. Mais avec les interfaces graphiques, et notamment avec les applications mobiles où des événements peuvent survenir à tout moment (par exemple, recevoir un SMS ou un appel téléphonique), la plupart des programmes ne sont pas des recettes, mais plutôt des ensembles de gestionnaires d'événements. Un gestionnaire d'événements est une manière de dire : « Lorsque cela se produit, l'application le fait. » Dans un langage traditionnel comme Java, vous devez comprendre les classes, les objets et les objets spéciaux appelés auditeurs pour exprimer un événement simple. Avec App Inventor, vous pouvez dire « Lorsqu'un utilisateur clique sur ce bouton... » ou « Lorsqu'un texte est reçu... » en faisant glisser un bloc « Quand ».

Quel type d'applications pouvez-vous créer ?

Vous pouvez créer de nombreux types d'applications différents avec App Inventor. Utilisez votre imagination et vous pourrez créer toutes sortes d'applications amusantes et utiles.

Jeux

Les gens commencent souvent par créer des jeux comme MoleMash (Chapitre 3) ou des applications qui vous permettent de dessiner des images amusantes sur les visages de vos amis (Chapitre 2). Au fur et à mesure de votre progression, vous pouvez créer vos propres versions de jeux plus complexes comme Pac-Man et Space Invaders. Vous pouvez même utiliser les capteurs du téléphone et déplacer les personnages en inclinant le téléphone (Chapitre 5).

Logiciel éducatif

La création d'applications ne se limite pas aux jeux simples. Vous pouvez également créer des applications qui informent et éduquent. Vous pouvez créer une application de quiz (Chapitre 8) pour vous aider, vous et vos camarades de classe, à étudier en vue d'un examen, ou même une application de création de quiz (Chapitre 10) qui permet aux utilisateurs de votre application de créer leurs propres quiz (pensez à tous les parents qui adoreraient celui-ci pour leurs longs voyages en voiture !).

Applications sensibles à la localisation

Étant donné qu'App Inventor donne accès à un capteur de localisation GPS, vous pouvez créer des applications qui savent où vous vous trouvez. Vous pouvez créer une application pour vous aider à vous rappeler où vous avez garé votre voiture (Chapitre 7), une application qui montre l'emplacement de vos amis ou collègues lors d'un concert ou d'une conférence, ou votre propre application de visite personnalisée de votre école, de votre lieu de travail ou d'un musée.

Applications de

haute technologie Vous pouvez créer des applications qui scannent des codes-barres, parlent, écoutent (reconnaissent des mots), jouent de la musique, créent de la musique (Chapitre 9), lisent des vidéos, détectent l'orientation et l'accélération du téléphone, prennent des photos et passent des appels téléphoniques. Les smartphones sont comme des couteaux suisses en matière de technologie, et App Inventor facilite le contrôle de cette technologie.

Applications d'envoi de SMS

Pas d'envoi de SMS en conduisant (Chapitre 4) n'est qu'un exemple des applications de traitement de SMS que vous pouvez créer. Vous pouvez également écrire une application qui envoie périodiquement des SMS « tu me manques » à tes proches, ou une application comme Broadcast Hub (chapitre 11) qui aide à coordonner de grands événements. Vous voulez une application qui permet à vos amis de voter pour des choses par SMS, comme sur American Idol ? Vous pouvez le créer avec App Inventor.

Applications qui contrôlent les

robots Le chapitre 12 montre comment créer une application qui agit comme un contrôleur pour un robot LEGO. Vous pouvez utiliser le téléphone comme télécommande ou le programmer pour qu'il soit un « cerveau » que le robot transporte avec lui. Le robot et le téléphone communiquent via Bluetooth, et les composants Bluetooth d'App Inventor vous permettent de créer des applications similaires contrôlant d'autres appareils Bluetooth.

Applications complexes

App Inventor abaisse considérablement la barrière d'entrée à la programmation et vous permet de créer des applications flashy et de haute technologie en quelques heures. Mais le langage fournit également des boucles, des conditions et d'autres constructions de programmation et logiques nécessaires pour créer des applications avec une logique complexe. Vous serez surpris de voir à quel point de tels problèmes de logique peuvent être amusants lorsque vous essayez de créer une application.

Applications Web

App Inventor permet également à vos applications de communiquer avec le Web.

Vous pouvez écrire des applications qui extraient des données de Twitter ou d'un flux RSS, ou d'un navigateur Amazon Bookstore qui vous permet de vérifier le coût en ligne d'un livre en scannant son code-barres.

Qui peut créer des applications ?

App Inventor est disponible gratuitement pour tous. Il fonctionne en ligne (au lieu de directement sur votre ordinateur) et est accessible depuis n'importe quel navigateur. Vous n'avez même pas besoin d'un téléphone pour l'utiliser : vous pouvez tester vos applications sur un émulateur Android inclus. En septembre 2014, il y avait 1,9 million d'utilisateurs enregistrés d'App Inventor dans 195 pays.

Ensemble, ils ont créé près de cinq millions d'applications.

Qui sont ces créateurs d'applications ? Étaient-ils déjà programmeurs à leurs débuts ?

Certains d'entre eux l'étaient, mais la plupart ne l'étaient pas.

L'une des expériences les plus révélatrices a été les cours dispensés par le co-auteur David Wolber à l'Université de San Francisco. À l'USF, App Inventor est enseigné dans le cadre d'un cours d'informatique de formation générale ciblant principalement les étudiants en commerce et en sciences humaines. De nombreux étudiants suivent ce cours parce qu'ils ont peur des mathématiques, et le cours répond aux redoutées exigences de base en mathématiques. La grande majorité n'a même jamais rêvé d'écrire un programme informatique.

Bien qu'ils n'aient aucune expérience préalable, les étudiants ont réussi à apprendre App Inventor et à créer d'excellentes applications. Une major anglaise a créé la première application No Texting While Driving, deux majors en communication ont créé Android, Where's My Car ?

(Chapitre 7), et un étudiant en études internationales a créé l'application Broadcast Hub (Chapitre 11).

Lorsqu'un étudiant en art a frappé à la porte du bureau de Wolber un soir, bien après les heures d'ouverture, pour lui demander comment écrire une boucle while , Wolber savait qu'App Inventor avait radicalement changé le paysage de l'enseignement informatique.

Les médias en ont également saisi l'importance. Le New York Times a qualifié App Inventor de « logiciel de création d'applications à faire soi-même ». Le San Francisco Chronicle a rendu compte du travail des étudiants de l'USF dans un article intitulé « Google rend la création d'applications accessible au grand public. » Le magazine Wired a présenté Daniel Finnegan, l'auteur de No Texting While Driving, et a écrit que « l'histoire de Finnegan illustre un point puissant : il est temps que la programmation informatique soit démocratisée ».

Le chat est, comme on dit, sorti du sac (d'ailleurs, votre première application impliquera un chaton). App Inventor est désormais utilisé dans les cours de collège et de lycée du monde entier ; par plus de 2 500 filles dans 28 pays qui ont participé au Technovation Challenge, un programme parascolaire destiné aux lycéennes ; dans des cours pilotes pour le nouveau cours de placement avancé sur les principes de l'informatique au lycée ; et dans de nouveaux cours d'introduction dans plusieurs universités. Il y a maintenant des milliers d'amateurs, d'hommes d'affaires, de proposants de mariage et de bricoleurs qui parcourront les applications de création de sites App Inventor. Vous voulez participer à l'action ? Aucune expérience en programmation n'est requise !

Conventions utilisées dans ce livre

Les conventions typographiques suivantes sont utilisées dans ce livre :

Italique

Indique les nouveaux termes, URL, adresses e-mail, noms de fichiers et extensions de fichiers.

Largeur constante

Utilisé pour les listes de programmes, ainsi que dans les paragraphes pour faire référence à des éléments de programme tels que les noms de variables ou de fonctions, les bases de données, les types de données, les variables d'environnement, les instructions et les mots-clés.

Gras à largeur constante

Affiche les commandes ou tout autre texte qui doit être saisi littéralement par l'utilisateur.

Italique à largeur constante

Affiche le texte qui doit être remplacé par des valeurs fournies par l'utilisateur ou par des valeurs déterminé par le contexte.



Astuce Cet élément signifie un conseil ou une suggestion.



Remarque Cet élément indique les instructions permettant de tester l'application que vous créez.

Utilisation d'exemples de code

Du matériel supplémentaire (exemples de code, exercices, etc.) est disponible en téléchargement sur <https://appinventor.org/bookFiles>.

Ce livre est là pour vous aider à accomplir votre travail. En général, si un exemple de code est proposé avec ce livre, vous pouvez l'utiliser dans vos programmes et votre documentation. Vous n'avez pas besoin de nous contacter pour obtenir une autorisation, sauf si vous reproduisez une partie importante du code. Par exemple, écrire un programme utilisant plusieurs morceaux de code de ce livre ne nécessite aucune autorisation. La vente ou la distribution d'un CD-ROM d'exemples tirés des livres d'O'Reilly nécessite une autorisation. Répondre à une question en citant ce livre et en citant un exemple de code ne nécessite pas d'autorisation. L'incorporation d'une quantité importante d'exemples de code de ce livre dans la documentation de votre produit nécessite une autorisation.

Nous apprécions, mais nous ne demandons pas d'attribution. Une attribution comprend généralement le titre, l'auteur, l'éditeur et l'ISBN. Par exemple : « App Inventor 2 de David Wolber, Hal Abelson, Ellen Spertus et Liz Looney (O'Reilly). Copyright 2015 David Wolber, Hal Abelson, Ellen Spertus et Liz Looney, 978-1-491-90684-2.

Si vous pensez que votre utilisation d'exemples de code ne respecte pas l'usage loyal ou l'autorisation accordée ci-dessus, n'hésitez pas à nous contacter à permissions@oreilly.com.

Livres Safari® en ligne

Safari Books Online est une bibliothèque numérique à la demande qui propose du contenu expert en à la fois sous forme de livres et de vidéos rédigés par les plus grands auteurs mondiaux dans les domaines de la technologie et des affaires.

Les professionnels de la technologie, les développeurs de logiciels, les concepteurs de sites Web et les professionnels des affaires et de la création utilisent Safari Books Online comme principale ressource pour la recherche, la résolution de problèmes, l'apprentissage et la formation de certification.

Safari Books Online propose une gamme de forfaits et de tarifs pour les entreprises, les gouvernements, l'éducation et les individus.

Les membres ont accès à des milliers de livres, de vidéos de formation et de prépublications manuscrits dans une base de données entièrement consultable provenant d'éditeurs comme O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology et des centaines d'autres. Pour plus d'informations sur Safari Books Online, veuillez nous rendre visite en ligne.

Comment nous contacter

Veuillez adresser vos commentaires et questions concernant ce livre à l'éditeur :

- O'Reilly Media, Inc.
- 1005, route Gravenstein Nord
- Sébastopol, CA 95472
- 800-998-9938 (aux États-Unis ou au Canada)
- 707-829-0515 (international ou local)
- 707-829-0104 (télécopieur)

Nous avons une page Web pour ce livre, où nous répertorions les errata, des exemples et toute information supplémentaire. Vous pouvez accéder à cette page à <http://bit.ly/app-inventor2>.

Pour commenter ou poser des questions techniques sur ce livre, envoyez un e-mail à
bookquestions@oreilly.com.

Pour plus d'informations sur nos livres, cours, conférences et actualités, consultez notre site Web à l'adresse <http://www.oreilly.com>.

Retrouvez-nous sur Facebook : <http://facebook.com/oreilly>

Suivez-nous sur Twitter : <http://twitter.com/oreillymedia> Regardez-nous sur YouTube : <http://www.youtube.com/oreillymedia>

Remerciements

La perspective pédagogique qui motive App Inventor considère que l'informatique peut être un moyen de mobiliser des idées puissantes grâce à un apprentissage actif. En tant que tel, App Inventor fait partie d'un mouvement continu dans le domaine des ordinateurs et de l'éducation qui a commencé avec les travaux de Seymour Papert et du MIT Logo Group dans les années 1960, et dont l'influence persiste aujourd'hui à travers de nombreuses activités et programmes conçus pour soutenir la pensée informatique.

La conception d'App Inventor s'appuie sur des recherches antérieures en informatique éducative et sur le travail de Google avec les environnements de développement en ligne. Le cadre de programmation visuelle est étroitement lié au langage de programmation MIT Scratch.

L'implémentation spécifique d'App Inventor 2 est basée sur Blockly, développé chez Google et dirigé par Neil Fraser. Le compilateur qui traduit le langage des blocs visuels pour une implémentation sur Android utilise le Kawa Language Framework et le dialecte Kawa du langage de programmation Scheme, développé par Per Bothner et distribué dans le cadre du système d'exploitation GNU par la Free Software Foundation.

Les auteurs souhaitent remercier Google et l'équipe d'origine d'App Inventor pour leur soutien à notre travail et à nos efforts d'enseignement à l'USF, au Mills College et au MIT. Nous remercions tout particulièrement Mark Friedman, responsable technique d'App Inventor, Karen Parker, chef de projet, ainsi que les ingénieurs Sharon Perl et Debby Wallach.

Nous tenons également à remercier l'équipe MIT App Inventor pour son travail et le développement continu d'App Inventor. Des remerciements particuliers vont au responsable technique Andrew McKinney, au gourou polyvalent Jef Schiller, aux directeurs de l'éducation et de la sensibilisation Shaileen Pokress et Josh Sheldon, au héros méconnu et ingénieur Jose Domínguez, ainsi qu'aux principaux contributeurs « sabbatiques » Franklyn Turbak et Ralph Morelli.

Nous devons également un merci tout particulier à Cayla Shaver, étudiante à l'Université de San Francisco, pour son travail d'édition extraordinaire et pour son aide à la conversion de ce livre pour App Inventor 2.

Enfin, nous tenons à remercier le soutien de nos conjoints respectifs : le mari d'Ellen, Keith Golden ; l'épouse de Hal, Lynn Abelson ; le mari de Liz, Kevin Looney ; et

L'épouse de David, Minerva Novoa. La nouvelle maman Ellen est également reconnaissante pour l'aide de la nounou Neil Fullagar.

BonjourPurr

Ce chapitre vous permet de commencer à créer des applications. Il présente les éléments clés d'App Inventor, du Component Designer et de l'éditeur de blocs, et vous guide à travers les étapes de base de la création de votre première application, HelloPurr. Lorsque vous aurez terminé, vous serez prêt à créer des applications sur le tien.

Un premier programme typique avec un nouveau système informatique imprime le message « Hello World » pour montrer que tout est correctement connecté. Cette tradition remonte aux années 1970 et aux travaux de Brian Kernighan sur le langage de programmation C aux Bell Labs. Avec App Inventor, même les applications les plus simples font plus que simplement afficher des messages : elles émettent des sons et réagissent lorsque vous touchez l'appareil. Nous allons donc commencer tout de suite avec quelque chose de plus excitant : votre première application (comme le montre la figure 1-1) sera « HelloPurr », une image d'un chat qui miaule lorsque vous le touchez et ronronne lorsque vous le secouez. l'appareil sur lequel il est visualisé.



Figure 1-1. L'application HelloPurr

Ce que vous apprendrez

Le chapitre couvre les sujets suivants :

- Créez des applications en sélectionnant les composants et en spécifiant leur comportement.
- Utilisation du Concepteur de composants pour sélectionner des composants. Certains composants sont visibles sur l'écran de l'appareil et certains ne le sont pas.
- Ajout de médias (sons et images) aux applications en les téléchargeant depuis votre ordinateur.
- Utilisation de l'éditeur de blocs pour assembler des blocs qui définissent les comportements des composants.
- Test d'applications avec les tests en direct d'App Inventor. Cela vous permet de voir à quoi ressembleront et se comporteront les applications sur l'appareil, étape par étape, même pendant que vous les créez.
- Empaquetage des applications que vous créez et les télécharger sur un appareil.

2 Chapitre 1 : HelloPurr

L'environnement App Inventor

Vous pouvez commencer à programmer avec App Inventor en ouvrant un navigateur sur ai2.appinventor.mit.edu. Cela ouvre la dernière version d'App Inventor, publiée en décembre 2013.

Certaines personnes l'appellent App Inventor 2, mais elle s'appelle officiellement App Inventor et la version précédente s'appelle App Inventor Classic. Dans ce livre, vous utiliserez la nouvelle version.

L'environnement de programmation App Inventor comprend trois éléments clés :

- Le concepteur de composants (Figure 1-2). Vous l'utilisez pour sélectionner des composants pour votre app et spécifiez leurs propriétés.
- L' éditeur de blocs (Figure 1-3). Vous l'utilisez pour spécifier comment les composants se comporteront (par exemple, ce qui se passe lorsqu'un utilisateur clique sur un bouton).
- Un appareil Android avec lequel vous pouvez réellement exécuter et tester votre application pendant que vous la développez. Si vous ne disposez pas d'un appareil Android à portée de main, vous pouvez tester les applications que vous créez à l'aide de l'émulateur Android fourni avec le système.

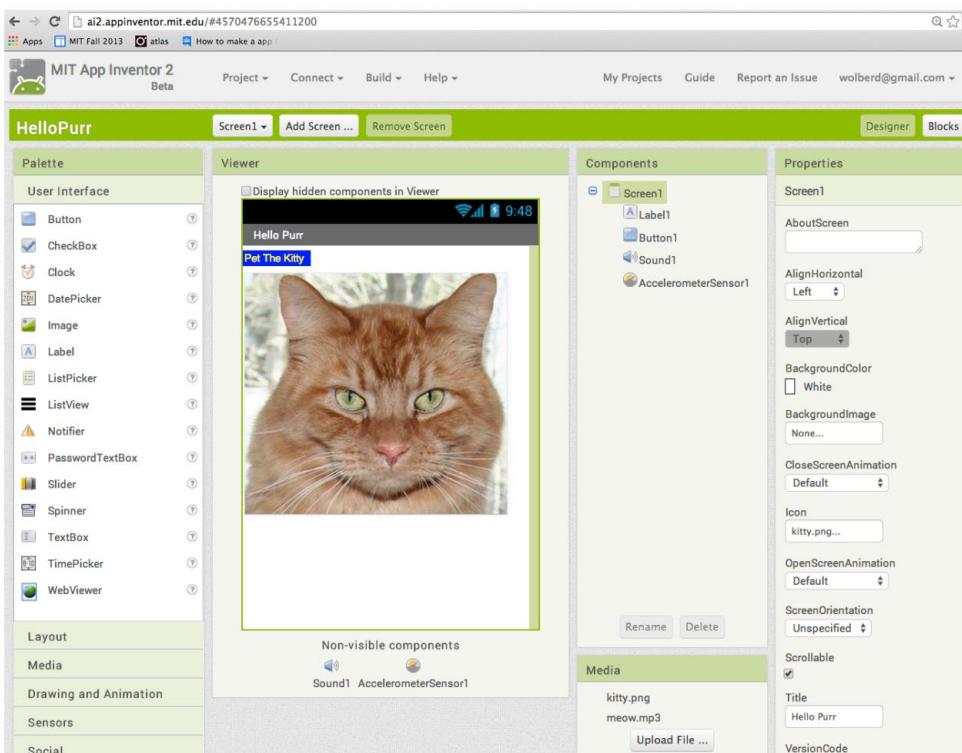


Figure 1-2. Le concepteur de composants pour spécifier à quoi ressemblera l'application

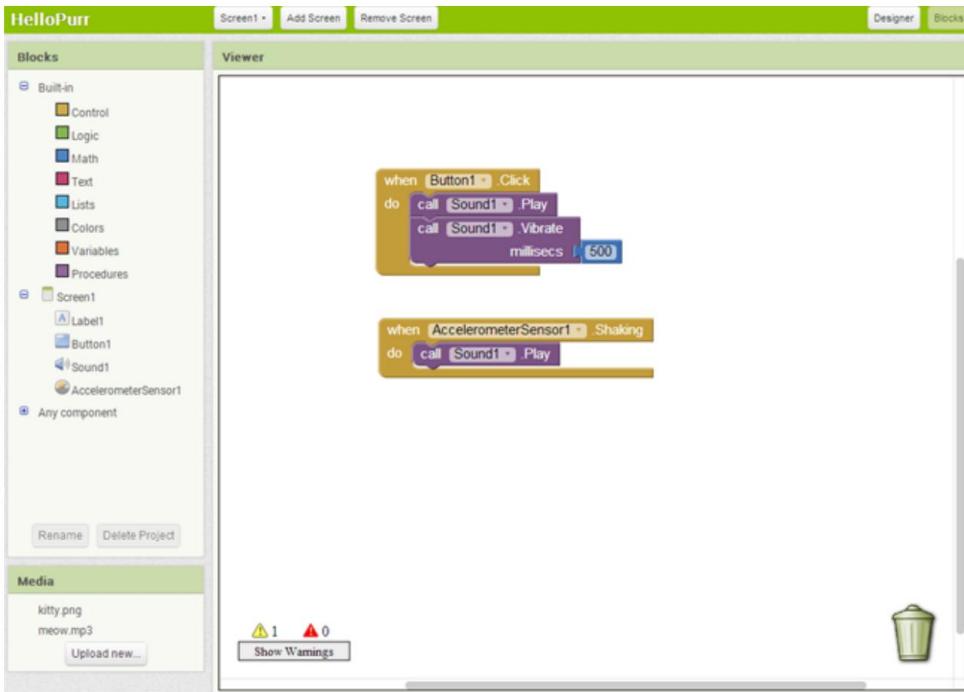


Figure 1-3. L'éditeur de blocs pour spécifier le comportement de l'application

La première fois que vous accédez à ai2.appinventor.mit.edu, vous verrez la page Projets, qui sera en grande partie vide car vous n'avez pas encore créé de projets. Pour créer un projet, en haut à gauche de la page, cliquez sur « Nouveau projet », saisissez le nom du projet « HelloPurr » (un mot sans espaces), puis cliquez sur OK.

La première fenêtre qui s'ouvre est le Concepteur de composants. L'éditeur de blocs est disponible en cliquant sur le bouton « Blocs » dans le coin supérieur droit de la fenêtre.

App Inventor est un outil de cloud computing, ce qui signifie que votre application est stockée sur un serveur en ligne pendant que vous travaillez. Ainsi, si vous fermez App Inventor, votre application sera là à votre retour ; vous n'avez pas besoin de sauvegarder quoi que ce soit sur votre ordinateur comme vous le feriez avec, par exemple, un fichier Microsoft Word.

Conception des composants

Le premier outil que vous utiliserez est le Component Designer (ou simplement Designer). Les composants sont les éléments que vous combinez pour créer des applications, comme les ingrédients d'une recette.

Certains composants sont très simples, comme un composant Label , qui affiche du texte à l'écran, ou un composant Button , sur lequel vous appuyez pour lancer une action. D'autres composants sont plus élaborés : un canevas de dessin pouvant contenir des images fixes ou des animations ; un accéléromètre, qui est un capteur de mouvement qui détecte lorsque vous déplacez ou secouez le

4 Chapitre 1 : HelloPurr

appareil; ou des composants qui créent ou envoient des messages texte, lisent de la musique et des vidéos, obtiennent des informations à partir de sites Web, etc.

Lorsque vous ouvrez Designer, il apparaît comme indiqué dans la figure 1-4.

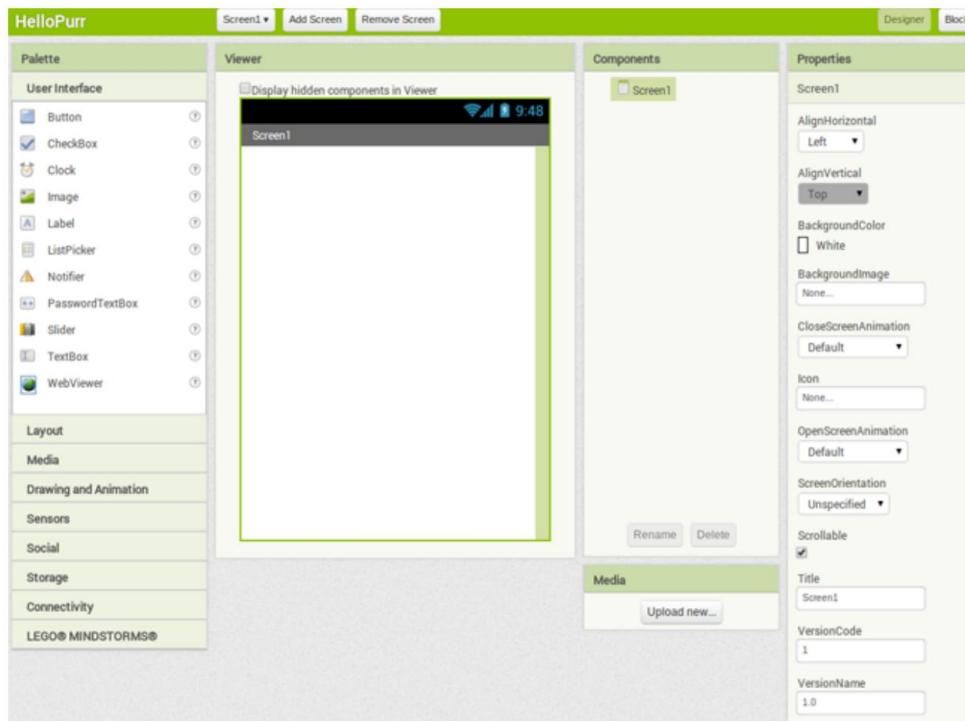


Figure 1-4. Le concepteur de composants App Inventor

Le Designer est divisé en plusieurs zones :

- Vers le centre se trouve une zone blanche appelée Visionneuse. C'est ici que vous placez composants et organisez-les pour définir à quoi vous souhaitez que votre application ressemble. La visionneuse n'affiche qu'une indication approximative de l'apparence de l'application. Par exemple, une ligne de texte peut s'interrompre à un endroit différent de votre appareil que sur la visionneuse. Pour voir à quoi ressemblera réellement votre application , vous devrez la tester sur votre appareil ou sur l'émulateur (nous vous montrerons comment procéder sous peu).
- À gauche du visualiseur se trouve la palette, qui est une liste de composants parmi lesquels vous pouvez sélectionner. La palette est divisée en sections ; à ce stade, seuls les composants de l'interface utilisateur sont visibles, mais vous pouvez voir les composants dans d'autres sections de la palette en cliquant sur les en-têtes intitulés Mise en page, Média, etc.
- À droite du visualiseur se trouve la liste Composants , qui répertorie les composants de votre projet. Tout composant que vous faites glisser dans la visionneuse apparaîtra également

dans cette liste. Actuellement, le projet ne comporte qu'un seul composant répertorié : Screen1, qui représente l'écran de l'appareil lui-même.

- Sous la liste des composants se trouve une zone qui affiche les médias (images et son) dans le projet. Ce projet n'a pas encore de média, mais vous en ajouterez bientôt.
- À l'extrême droite se trouve une section qui affiche les propriétés des composants ; lorsque vous cliquez sur un composant dans la visionneuse, vous verrez ses propriétés répertoriées ici. Les propriétés sont des détails sur chaque composant que vous pouvez modifier. (Par exemple, lorsque vous cliquez sur un composant Label , vous pouvez voir les propriétés liées à la couleur, au texte, à la police, etc.) À l'heure actuelle, il affiche les propriétés de l'écran (appelé Screen1), qui incluent une couleur d'arrière-plan, une image d'arrière-plan et un titre.

Pour l'application HelloPurr, vous aurez besoin de deux composants visibles (considérez-les comme des composants que vous pouvez réellement voir dans l'application) : le composant Label indiquant « Pet the Kitty » et un composant Button avec l'image d'un chat dedans. Vous aurez également besoin d'un composant sonore non visible capable de jouer des sons, tels que « miaou », et d'un composant accéléromètre pour détecter lorsque l'appareil est secoué. Ne vous inquiétez pas, nous vous guiderons à travers chaque composant, étape par étape.

FAIRE UNE ÉTIQUETTE

Le premier composant à ajouter est un Label :

1. Accédez à la palette, ouvrez le tiroir de l'interface utilisateur s'il n'est pas ouvert, cliquez sur Étiquette (qui apparaît environ six endroits plus bas dans la liste des composants) et faites-la glisser vers la visionneuse. Vous verrez une forme rectangulaire apparaître sur la visionneuse, contenant les mots « Texte pour Label1 ».
2. Regardez la zone Propriétés sur le côté droit du Designer. Il montre le propriétés de l'étiquette. À peu près à mi-chemin, il y a une propriété appelée Text, avec une zone pour le texte de l'étiquette. Remplacez le texte par « Pet the Kitty » et appuyez sur Retour. Vous verrez le texte changer dans la visionneuse.
3. Modifiez la BackgroundColor de l'étiquette en cliquant sur la case qui est actuellement lit Aucun, pour sélectionner une couleur dans la liste qui apparaît. Sélectionnez Bleu. Changez également le TextColor de l'étiquette en Jaune. Enfin, modifiez FontSize à 20.

Le Designer devrait maintenant apparaître comme indiqué dans la figure 1-5.

6 Chapitre 1 : HelloPurr

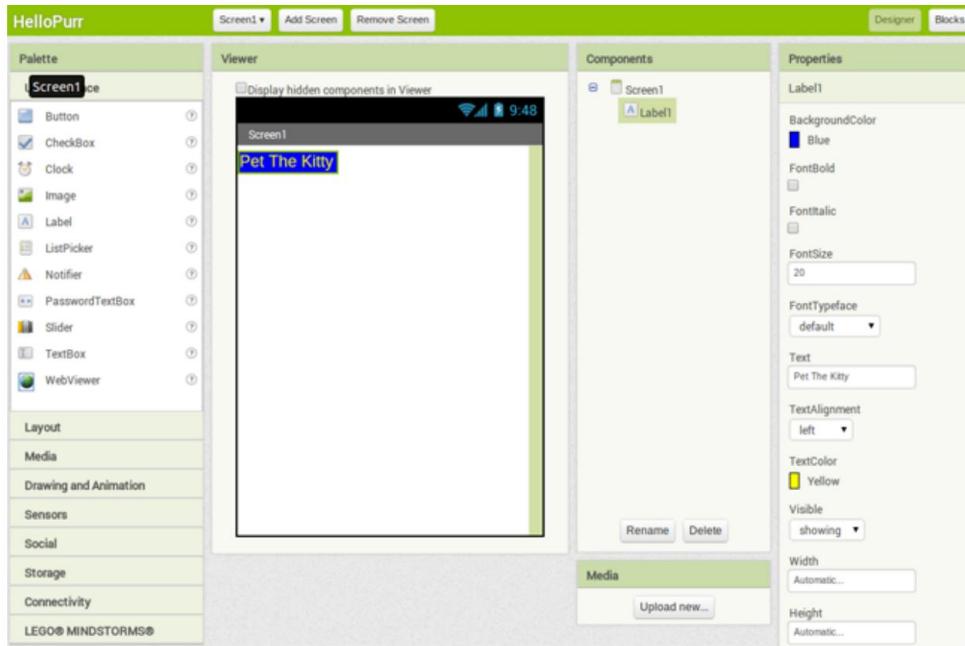


Figure 1-5. L'application a désormais une étiquette

AJOUT DU BOUTON

Le chaton pour HelloPurr est implémenté en tant que composant Button : vous créez un bouton normal, puis remplacez l'image du bouton par le chaton. Pour créer d'abord le bouton de base, accédez à la palette dans le concepteur et cliquez sur Bouton (en haut de la liste des composants). Faites-le glisser sur la visionneuse, en le plaçant sous l'étiquette. Vous verrez un bouton rectangulaire apparaître sur la visionneuse.

Vous disposez désormais d'un bouton que vous utiliserez pour déclencher l'effet sonore lorsque quelqu'un tape dessus, mais nous voulons vraiment qu'il ressemble à l'image du chaton, pas à un vieux rectangle simple. Pour que le bouton ressemble au chat :

1. Tout d'abord, vous devez télécharger une photo du chat et l'enregistrer sur le bureau de votre ordinateur. Vous pouvez le télécharger sur <http://appinventor.org/bookFiles/HelloPurr/kitty.png>. L'extension .png est un format d'image standard similaire à .jpg et .gif ; tous ces types de fichiers fonctionneront dans App Inventor, tout comme la plupart des fichiers audio standard tels que .mpg ou .mp3. Vous pouvez également télécharger le fichier audio dont vous aurez besoin pour faire miauler le chat sur <http://appinventor.org/bookFiles/HelloPurr/meow.mp3>.
2. La zone Propriétés doit afficher les propriétés du bouton. Si ce n'est pas le cas, cliquez sur le bouton dans la visionneuse pour afficher les propriétés du bouton sur la droite. Dans la zone Propriétés, cliquez sur la zone sous Image (qui indique actuellement « Aucune... »).

3. Cliquez sur « Télécharger le fichier ». Ensuite, cliquez sur « Choisir un fichier » et parcourez pour sélectionner le fichier kitty.png que vous avez téléchargé précédemment sur votre ordinateur, puis cliquez sur OK.
4. Après le téléchargement de l'image, kitty.png doit être répertorié comme option pour la propriété image du bouton. Cliquez sur OK pour le choisir. Vous verrez également le fichier répertorié dans la zone Média de la fenêtre Designer, juste en dessous de la liste des composants. Et si vous regardez le bouton dans le concepteur, vous verrez l'image du chat affichée : le bouton ressemble maintenant à un chat.
5. Vous avez peut-être également remarqué que l'image du chat porte toujours les mots « Texte pour Bouton 1» affiché dessus. Vous ne voulez probablement pas cela dans votre application, alors allez-y et videz la propriété Text de Button1.

Le Designer devrait maintenant apparaître comme indiqué dans la figure 1-6.

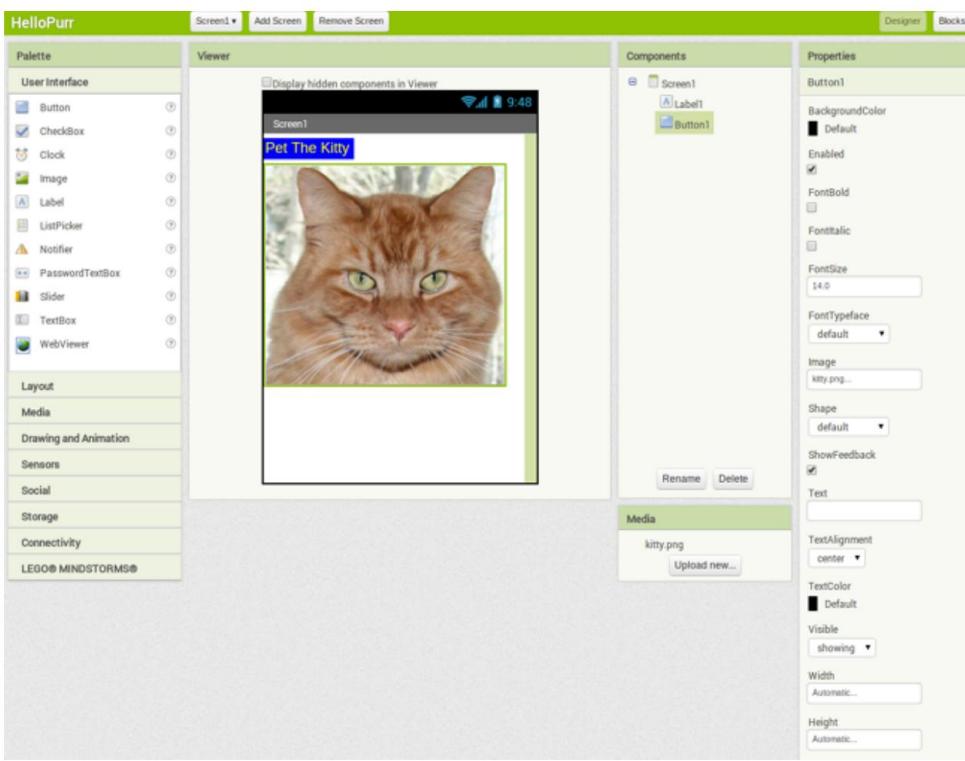


Figure 1-6. L'application avec une étiquette et un bouton avec une image dessus

AJOUTER LE SON MIAOU

Dans votre application, vous souhaitez que le chat miaule lorsque vous appuyez sur le bouton. Pour cela, vous devrez ajouter le son miaou et programmer le comportement du bouton pour qu'il joue ce son lorsque vous cliquez sur le bouton :

8 Chapitre 1 : HelloPurr

1. Si vous n'avez pas téléchargé le fichier meow.mp3 sur le bureau de votre ordinateur, faites-le maintenant en utilisant ce lien : <http://appinventor.org/bookFiles/HelloPurr/meow.mp3>.
2. Accédez à la palette à gauche de la fenêtre Designer et cliquez sur l'en-tête marqué Média pour développer la section Médias. Faites glisser un composant Sound et placez-le dans la visionneuse. Peu importe où vous le déposez, il apparaîtra dans la zone située à le bas de la visionneuse porte la mention « Composants non visibles ». Non visible les composants sont des objets qui font des choses pour l'application mais n'apparaissent pas dans le visuel interface utilisateur.
3. Cliquez sur Sound1 pour afficher ses propriétés. Cliquez sur la propriété Source , puis allez à travers les étapes pour télécharger et choisir le fichier meow.mp3 que vous avez téléchargé plus tôt. Lorsque vous avez terminé, vous devriez voir kitty.png et meow.mp3 répertoriés dans la section Média du Designer.

Le tableau 1-1 répertorie les composants que vous avez rassemblés jusqu'à présent pour votre application.

Tableau 1-1. Les composants que vous avez ajoutés à l'application HelloPurr

Type de composant	Groupe de palettes	Nom du composant	Objectif
Bouton	Bouton de l'interface utilisateur 1		Appuyez pour faire miauler le chat.
Étiquette	Étiquette de l'interface utilisateur1		Affiche le texte « Pet the Kitty ».
Son	Médias	Son1	Jouez le son miaou.

Tests en direct

Avec App Inventor, vous pouvez afficher et tester votre application sur un appareil Android pendant que vous créez-le. Tester votre application de manière incrémentielle est une pratique utilisée par les plus efficaces. développeurs de logiciels et vous fera gagner des heures de travail.

Si vous disposez d'un appareil Android et d'une connexion Internet avec WiFi, vous pouvez configurer tests en direct en quelques minutes et vous n'avez pas besoin de télécharger de logiciel sur votre ordinateur (juste une application sur votre téléphone). Si vous n'avez pas d'appareil Android, vous devez effectuer une configuration supplémentaire afin d'utiliser l'émulateur, les détails de qui sont couverts sur <http://appinventor.mit.edu/explore/ai2/setup.html>.

Si vous possédez un appareil Android, procédez comme suit :

1. Sur votre appareil, téléchargez et installez l'application « MIT AI2 Companion » depuis le Google Play Store. Lancez l'application une fois installée.
2. Connectez votre ordinateur et votre appareil à la même connexion WiFi.

3. Dans App Inventor (dans le navigateur), dans le menu supérieur, sélectionnez Connecter, puis choisissez AI Companion, comme illustré dans la figure 1-7.

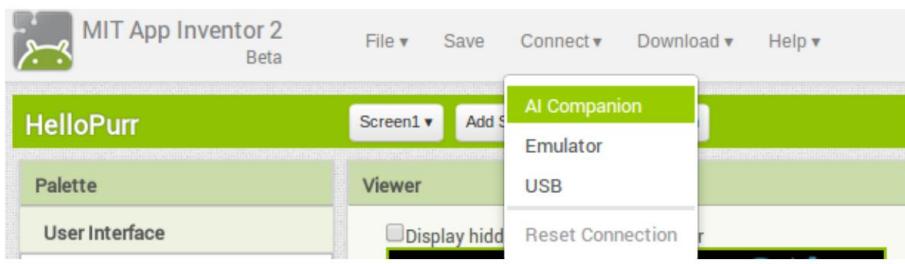


Figure 1-7. Cliquez sur Connecter, puis sélectionnez AI Companion

4. Sur votre appareil, lancez l'application que vous avez installée, le MIT AI2 Companion, comme illustré à la figure 1-8. Sélectionnez « Scanner le code QR », puis placez votre appareil devant le code QR sur l'écran de l'ordinateur pour le scanner.



Figure 1-8. Sur votre appareil, ouvrez l'application Companion et cliquez sur « Scanner le code QR »

Si tout se passe bien, vous devriez voir l'application HelloPurr s'exécuter sur votre appareil, y compris tous les composants que vous avez ajoutés. Lorsque vous apportez des modifications dans App Inventor Designer ou dans l'éditeur de blocs, ces modifications apparaîtront également sur l'appareil.



Configuration des tests en direct Si vous rencontrez des difficultés pour configurer les tests en direct, visitez <http://appinventor.mit.edu/explore/ai2/setup.html>.

Si votre application apparaît sur l'appareil, continuez et appuyez sur le bouton. Penses-tu quelque chose va arriver ? Ce ne sera pas le cas, car vous n'avez pas encore demandé au bouton de faire quoi que ce soit. C'est le premier point important à comprendre à propos d'App Inventor : par exemple

Pour chaque composant que vous ajoutez dans le Designer, vous devez passer à l'éditeur de blocs et créer le code pour que ce composant fasse tout ce que vous voulez qu'il fasse.

Ajout de comportements aux composants

Vous venez d'ajouter les composants Button, Label et Sound comme éléments de base de votre première application. Maintenant, faisons miauler le chat lorsque vous appuyez sur le bouton. Vous faites cela avec l'éditeur de blocs. En haut à droite du Concepteur de composants, cliquez sur « Blocs ».

Regardez la fenêtre de l'éditeur de blocs. C'est ici que vous indiquez aux composants quoi faire et quand le faire. Vous allez demander au bouton du chat de jouer un son lorsque l'utilisateur l'appuie. Si les composants sont des ingrédients dans une recette, vous pouvez considérer les blocs comme des instructions de cuisson.

FAIRE LE MIAOU DU CHAT

En haut à gauche de la fenêtre, sous l'en-tête Blocs, vous verrez une colonne qui comprend un tiroir intégré et un tiroir pour chaque composant que vous avez créé dans Designer : Button1, Label1, Screen1 et Sound1. Lorsque vous cliquez sur un tiroir, vous obtenez un certain nombre d'options (blocs) pour ce composant. Cliquez sur le tiroir pour Button1. Le tiroir s'ouvre et affiche une sélection de blocs que vous pouvez utiliser pour créer le comportement du bouton, en commençant par Button1.Click en haut, comme le montre la figure 1-9.

12 Chapitre 1 : HelloPurr

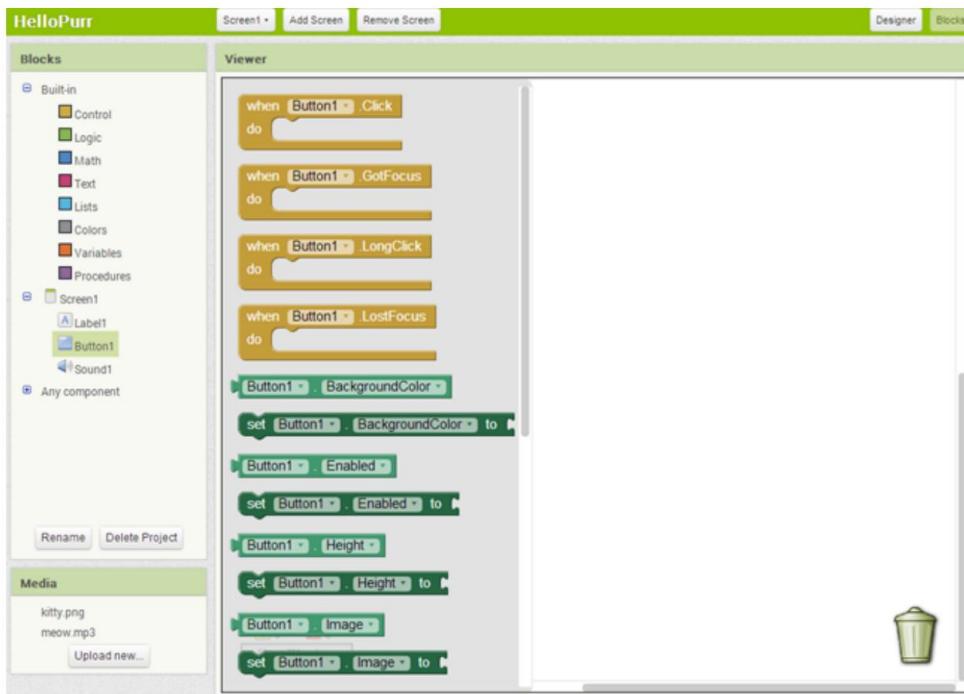


Figure 1-9. Cliquer sur Button1 affiche les blocs du composant

Cliquez sur le bloc intitulé Button1.Click et faites-le glisser dans l'espace de travail. Vous remarquerez que le mot «quand» est inclus dans le bloc Button1.Click . Les blocs comprenant le mot « quand » sont appelés gestionnaires d'événements ; ils précisent ce que les composants doivent faire lorsqu'un événement particulier se produit. Dans ce cas, l'événement qui nous intéresse se produit lorsque l'utilisateur de l'application appuie sur l'image du chat (qui est en réalité un bouton), comme le montre la figure 1-10. Vous allez ensuite ajouter quelques blocs pour programmer ce qui se passera en réponse à cet événement.

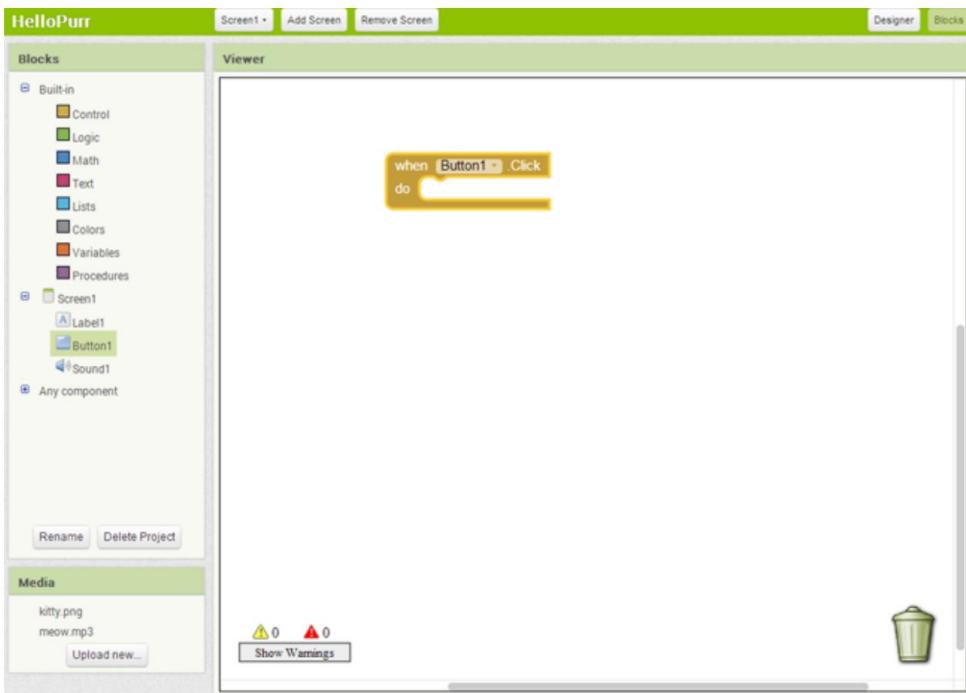


Figure 1-10. Vous spécifiez une réponse à l'utilisateur cliquant dans le bloc Button.Click

Cliquez sur Sound1 pour ouvrir le tiroir du composant audio, puis faites glisser le appelez le bloc Sound1.Play . (Rappelez-vous, plus tôt, nous avons défini la propriété de Sound1 sur le fichier sonore miaou que vous avez téléchargé sur votre ordinateur.) À ce stade, vous avez peut-être remarqué que le bloc d'appel Sound1.Play est conçu de manière à pouvoir s'insérer dans un espace marqué « faire ». » dans le bloc Button1.Click . App Inventor est configuré de manière à ce que seuls certains blocs soient assemblés ; de cette façon, vous savez toujours que vous connectez des blocs qui fonctionnent réellement ensemble. Dans ce cas, les blocs avec le mot « call » incitent les composants à faire des choses. Les deux blocs doivent s'emboîter pour former une unité, comme le montre la figure 1-11, et vous entendrez un claquement lorsqu'ils se connectent.

14 Chapitre 1 : HelloPurr

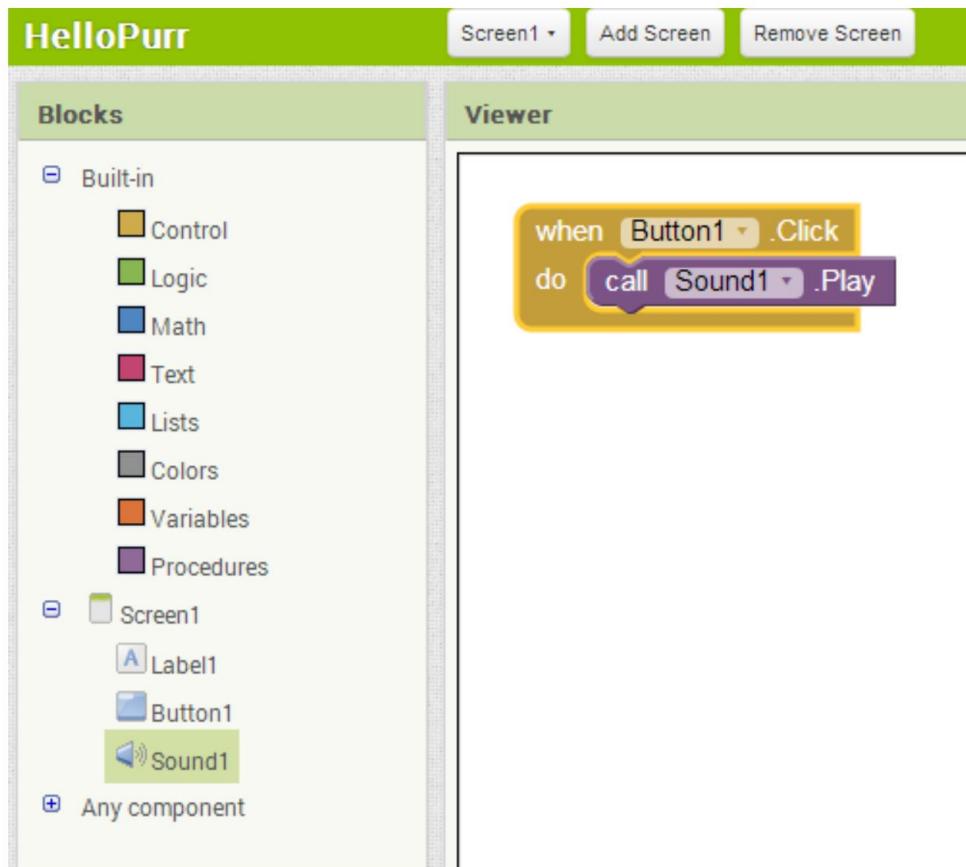


Figure 1-11. Désormais, lorsque quelqu'un clique sur le bouton, le son miaou est joué

Contrairement au code de programmation traditionnel (qui ressemble souvent à un fouillis de "mots" charabia), les blocs de réponse aux événements dans App Inventor épellent les comportements que vous essayez de créer de manière simple et compréhensible. Dans ce cas, nous disons essentiellement : « Hé, App Inventor, lorsque quelqu'un appuie sur le bouton du chat, joue le son miaou. »



Testez votre application Vérifiez que tout fonctionne correctement : il est important de tester votre application chaque fois que vous ajoutez quelque chose de nouveau. Appuyez sur le bouton de l'appareil (ou cliquez dessus si vous utilisez l'émulateur). Vous devriez entendre le chat miauler.

Félicitations, votre première application est en cours d'exécution !

AJOUTER UN RONRONNEMENT

Nous allons maintenant faire ronronner et miauler le chat lorsque vous appuyez sur le bouton. Nous allons simuler le ronronnement en faisant vibrer l'appareil. Cela peut sembler difficile, mais en fait, c'est facile à faire car le composant Son que nous avons utilisé pour jouer le son de miaulement peut également faire vibrer l'appareil. App Inventor vous aide à exploiter ce type de fonctionnalités de base de l'appareil sans avoir à vous soucier de la façon dont l'appareil vibre réellement. Vous n'avez rien besoin de faire de différent dans le Designer ; vous pouvez simplement ajouter un deuxième bloc d'appel de fonction au clic sur le bouton dans l'éditeur de blocs :

1. Accédez à l'éditeur de blocs et cliquez sur Sound1 pour ouvrir le tiroir.
2. Sélectionnez appeler Sound1.Vibrate et faites-le glisser sous le bloc d'appel Sound1.Play dans la fente Button1.Click . Le bloc doit s'enclencher, comme le montre la figure 1-12. Si ce n'est pas le cas, essayez de le faire glisser de manière à ce que la petite encoche sur le bord supérieur de l'appel Sound1.Vibrate touche la petite bosse en bas de l'appel Sound1.Play.

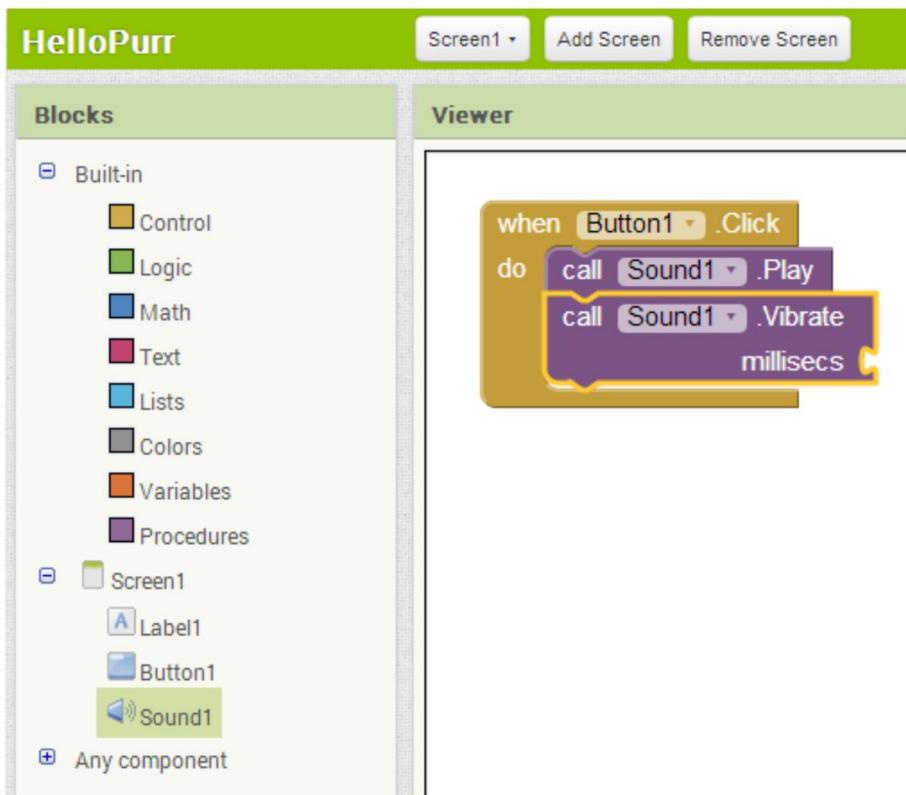


Figure 1-12. Jouer le son et vibrer sur l'événement Click

16 Chapitre 1 : HelloPurr

3. Vous avez peut-être remarqué que le bloc d'appel Sound1.Vibrate inclut le texte « millisecs » en bas à droite, et à côté se trouve une prise ouverte dépassant vers l'intérieur du bord du bloc. Un socket ouvert dans un bloc signifie que vous devez y brancher quelque chose pour spécifier davantage le fonctionnement du comportement. Dans ce cas, vous devez indiquer au bloc Vibreur combien de temps il doit vibrer. Vous devez spécifier cette durée en millièmes de seconde (millisecondes), ce qui est assez courant pour de nombreux langages de programmation. Ainsi, pour faire vibrer l'appareil pendant une demi-seconde, vous devez saisir une valeur de 500 millisecondes. Pour ce faire, vous devez saisir un bloc numérique. Cliquez sur le tiroir Math et vous verrez apparaître une liste de blocs bleus, comme le montre la figure 1-13.

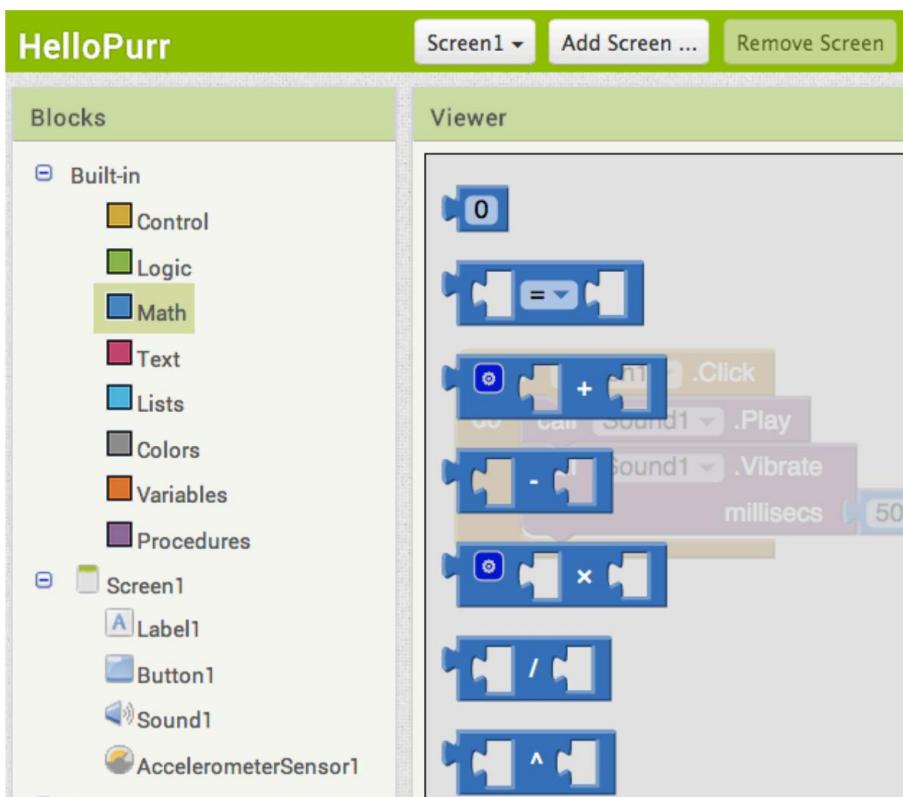


Figure 1-13. Ouvrir le tiroir Math

4. En haut de la liste, vous devriez voir un bloc avec un « 0 ». Vous pouvez faire glisser ce bloc, puis remplacer le 0 par le nombre de votre choix. Continuez et faites glisser le bloc numérique, comme indiqué dans la figure 1-14.

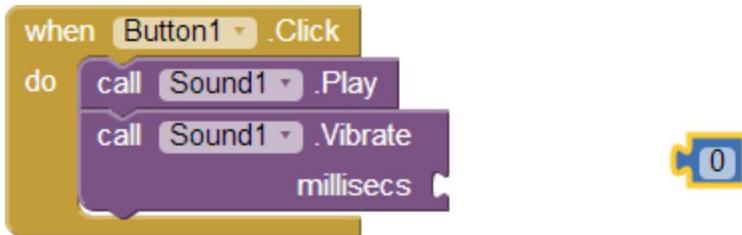


Figure 1-14. Choisir un bloc numérique (0 est la valeur par défaut)

5. Cliquez sur 0 et saisissez la nouvelle valeur, 500, comme indiqué dans la Figure 1-15.



Figure 1-15. Changer la valeur à 500

6. Branchez le bloc de numéros 500 dans la prise située sur le côté droit de call Sound1.Vibrate, comme indiqué sur la figure 1-16.



Figure 1-16. Brancher la valeur 500 dans la prise millisecs



Testez votre application Essayez-la ! Appuyez sur le bouton de l'appareil et vous ressentirez le ronronnement pendant une demi-seconde.

SECOUER L'APPAREIL

Maintenant, ajoutons un élément final qui exploite une autre fonctionnalité intéressante d'Android : faire miauler le chat lorsque vous secouez l'appareil. Pour ce faire, vous utiliserez un composant appelé AccelerometerSensor qui peut détecter lorsque vous secouez ou déplacez l'appareil.

18 Chapitre 1 : HelloPurr

1. Dans le Designer, dans la liste des composants de la palette, développez la zone Capteurs et faites glisser un AccelerometerSensor. Ne vous inquiétez pas de l'endroit où vous le faites glisser. Comme pour tout composant non visible, peu importe où vous le placez dans la visionneuse, il sera déplacé vers la section « Composants non visibles » en bas de la visionneuse.
2. Vous souhaiterez traiter quelqu'un qui secoue l'appareil comme un événement différent et distinct à partir du bouton cliquez. Cela signifie que vous avez besoin d'un nouveau gestionnaire d'événements. Accédez à l'éditeur de blocs. Il devrait y avoir un nouveau tiroir pour AccelerometerSensor1. Ouvrez-le et faites glisser le bloc AccelerometerSensor1.Shaking . Ce devrait être le deuxième bloc de la liste.
3. Tout comme vous l'avez fait avec le son et le clic sur le bouton, faites glisser un appel Bloc Sound1.Play et insérez-le dans l'espace de AccelerometerSensor1.Shaking. Essayez-le en secouant l'appareil.

La figure 1-17 montre les blocs de l'application HelloPurr terminée.

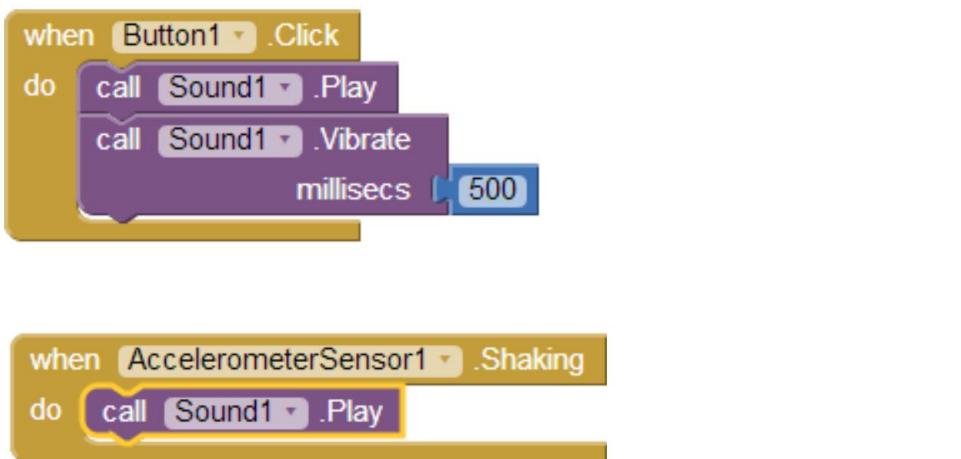


Figure 1-17. Les blocs pour HelloPurr

Téléchargement de l'application sur votre appareil Android

La fonction de test en direct d'App Inventor vous permet de tester facilement l'application lorsque vous êtes connecté à votre appareil. Le seul problème est que si vous déconnectez votre appareil d'App Inventor, l'application exécutée sur l'appareil s'arrêtera et vous ne trouverez l'application nulle part sur l'appareil car elle n'a jamais été réellement installée ; il fonctionnait simplement dans l'application App Inventor Companion.

Vous pouvez télécharger et installer l'application terminée pour qu'elle fonctionne sur n'importe quel appareil, même lorsqu'elle n'est pas connectée à l'ordinateur. Pour vous préparer à cela, définissez d'abord le

icône afin que lorsque vous l'installez sur un appareil, il apparaisse avec une image distinctive dans la liste des applications. Vous pouvez le faire dans le Designer en sélectionnant le composant Screen , en cliquant sur sa propriété Icône , puis en téléchargeant un fichier image comme icône (par exemple, l'image du chaton).

Ensuite, assurez-vous que votre appareil permet de télécharger des applications à partir d'autres endroits. que l'Android Market. Pour la plupart des appareils Android, pour ce faire, accédez à Paramètres → Applications, puis cochez la case à côté de « Sources inconnues ».

Ensuite, de retour dans App Inventor, dans Designer, cliquez sur Créer et sélectionnez « Application (fournir le code QR pour .apk) ». Vous devriez voir un message « Barre de progression » dans la fenêtre, un processus qui prend jusqu'à une minute. Lorsque le code QR de l'application terminée s'affiche, scannez-le sur votre appareil avec une application Barcode Scanner.¹ Après avoir scanné le code QR, l'appareil peut vous demander de saisir votre mot de passe pour votre compte Google. Lorsque vous aurez fini de saisir votre mot de passe, votre application commencera à se télécharger sur votre appareil et vous verrez une icône de téléchargement dans les notifications de votre appareil. Accédez à vos notifications, attendez la fin du téléchargement, puis choisissez l'application pour l'installer.

Après l'avoir installé, regardez les applications disponibles sur votre appareil et vous verrez maintenant HelloPurr, l'application que nous venons de créer. Vous l'exécutez comme n'importe quelle autre application. (Assurez-vous d'exécuter votre nouvelle application, et non l'application App Inventor Companion.) Vous pouvez maintenant arrêter l'application Companion ou débrancher votre appareil de l'ordinateur, et votre nouvelle application packagée sera toujours là.

Il est important de comprendre que cela signifie que votre application packagée est désormais séparée du projet sur App Inventor. Vous pouvez effectuer davantage de travail sur le projet dans App Inventor en connectant l'appareil à AI Companion comme auparavant. Mais cela ne changera pas l'application packagée désormais installée sur votre appareil. Si vous apportez d'autres modifications à votre application dans App Inventor, vous souhaiterez empaqueter le résultat et télécharger la nouvelle version pour remplacer l'ancienne sur l'appareil.

Partager l'application

Vous pouvez partager votre application de plusieurs manières. Pour partager l'application exécutable (le fichier .apk), cliquez d'abord sur Créer et choisissez « Application (enregistrer sur mon ordinateur) ». Cela créera un fichier avec une extension .apk sur votre ordinateur. Vous pouvez partager ce fichier avec d'autres personnes en leur envoyant une pièce jointe à un e-mail, qu'ils ouvriront ensuite avec leur application de messagerie sur leur appareil. Ou vous pouvez télécharger le fichier .apk quelque part sur le Web (par exemple, sur Dropbox). Assurez-vous simplement d'informer les personnes qui installent votre application qu'elles doivent

¹ Il existe de nombreux scanners de codes QR pour Android. Si vous n'en avez pas sur votre appareil, accédez au Play Store et installez-en un.

20 Chapitre 1 : HelloPurr

autoriser les « sources inconnues » dans les paramètres d'application de leur appareil afin d'installer des applications qui ne proviennent pas de l'Android Market.

Vous pouvez également créer un code QR pour l'application afin que les utilisateurs puissent le scanner sur leur appareil à partir du Web ou même d'une affiche physique. Il existe de nombreux outils qui génèrent un code QR à partir d'une URL (par exemple, consultez qrcode.kaywa.com). Vous pouvez ensuite couper et coller le code QR dans une page Web ou un document pour l'imprimer et le publier.

Vous pouvez également partager le code source (blocs) de votre application avec un autre développeur App Inventor. Pour ce faire, cliquez sur Mes projets, cochez l'application que vous souhaitez partager (dans ce cas, HelloPurr), puis sélectionnez Projet → Exporter le projet sélectionné. Le fichier créé sur votre ordinateur aura une extension .aia. Vous pouvez envoyer ce fichier par e-mail à quelqu'un, qui peut ouvrir App Inventor, choisir Projet → Importer le projet, puis sélectionner le fichier .aia. Cela donnera à l'utilisateur une copie complète de votre application, qui pourra ensuite être modifiée et personnalisée sans affecter votre version.

App Inventor disposera bientôt de sa propre galerie d'applications où vous pourrez partager vos applications et remixez les applications de développeurs du monde entier.

Variantes

Après avoir créé les applications de ce livre, vous réfléchirez probablement à des moyens de les améliorer. À la fin de chaque chapitre, nous vous proposerons également des idées de personnalisation à essayer. La personnalisation des applications vous amènera à explorer les composants et blocs disponibles et à apprendre à programmer par vous-même sans les instructions détaillées fournies dans les didacticiels.

Voici quelques choses à essayer pour HelloPurr :

- Lorsque vous secouez l'appareil, les miaulements semblent étranges, comme s'ils faisaient écho. En effet, le capteur de l'accéléromètre déclenche l'événement de secousses plusieurs fois par seconde en réponse à chaque mouvement individuel de haut en bas, de sorte que les miaulements se chevauchent. Si vous examinez le composant Son dans le Concepteur, vous verrez une propriété appelée Intervalle minimum. Cette propriété détermine à quelle distance les sons successifs peuvent commencer. Elle est actuellement réglée à un peu moins d'une demi-seconde (400 millisecondes), ce qui est inférieur à la durée d'un seul miaulement. En ajustant l'intervalle minimum, vous pouvez modifier le degré de chevauchement des miaulements.
- Si vous exécutez l'application packagée et que vous la promenez avec l'appareil dans votre poche, votre appareil miaule à chaque fois que vous bougez brusquement, ce qui pourrait vous embarrasser. Les applications Android sont généralement conçues pour continuer à fonctionner même lorsque vous ne les regardez pas ; votre application continue de communiquer avec l'accéléromètre et le miaulement continue. Pour vraiment quitter l'application, affichez

HelloPurr et appuyez sur le bouton de menu de l'appareil. Une option vous sera proposée pour arrêter l'application. Sélectionnez ceci pour fermer complètement l'application.

Résumé

Voici quelques-uns des concepts que nous avons abordés dans ce chapitre :

- Vous créez des applications en sélectionnant des composants dans le Designer, puis dans les Blocs. Editeur, vous indiquez aux composants quoi faire et quand le faire.
- Certains composants sont visibles et d'autres non. Les visibles apparaissent dans l'interface utilisateur de l'application. Les non-visibles font des choses comme jouer des sons.
- Vous définissez le comportement des composants en assemblant des blocs dans l'éditeur de blocs. Vous faites d'abord glisser un gestionnaire d'événements, tel que Button1.Click, puis vous y placez des blocs de commandes tels que Sound.Play . Tous les blocs dans Button1.Click seront effectués lorsque l'utilisateur appuie sur le bouton.
- Certaines commandes nécessitent des informations supplémentaires pour fonctionner. Un exemple est Vibrer, qui doit savoir pendant combien de millisecondes vibrer. Ces valeurs sont appelées arguments ou paramètres.
- Les nombres sont représentés sous forme de blocs de nombres. Vous pouvez les brancher sur les commandes qui prennent des chiffres comme arguments.
- App Inventor comporte des composants de capteur. L' AccelerometerSensor peut détecter lorsque l'appareil est déplacé ou secoué.
- Vous pouvez regrouper les applications que vous créez et les télécharger sur le téléphone, où ils fonctionnent indépendamment d'App Inventor.

Pot de peinture

Ce didacticiel présente le composant Canvas permettant de créer des graphiques simples en deux dimensions (2D). Vous allez créer PaintPot, une application qui permet à l'utilisateur de dessiner sur l'écran dans différentes couleurs, puis la mettre à jour afin que l'utilisateur puisse prendre une photo et dessiner dessus à la place. D'un point de vue historique, PaintPot a été l'un des premiers programmes développés pour démontrer le potentiel des ordinateurs personnels, dès les années 1970. À l'époque, créer quelque chose comme cette simple application de dessin était une entreprise très complexe et les résultats étaient plutôt imparfaits. Mais désormais, avec App Inventor, n'importe qui peut rapidement créer une application de dessin plutôt sympa, ce qui constitue un excellent point de départ pour créer de la 2D.

Jeux.

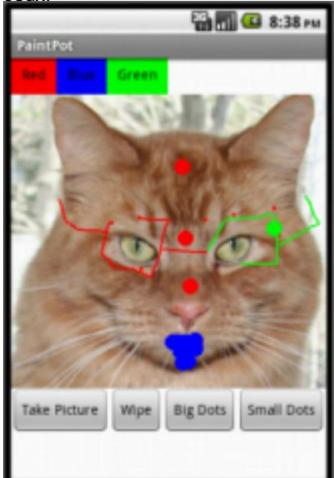


Figure 2-2. L'application PaintPot

Figure 2-1.



Avec l'application PaintPot illustrée à la figure 2-1, vous pouvez :

- Trempez votre doigt dans un pot de peinture virtuel pour dessiner dans cette couleur.
- Faites glisser votre doigt le long de l'écran pour tracer une ligne.
- Piquez l'écran pour faire des points.
- Utilisez le bouton en bas pour essuyer l'écran.
faire le ménage.
- Changez la taille du point en grand ou petit avec les boutons au fond.
- Prenez une photo avec l'appareil photo, puis dessinez dessus.
image.

Ce que vous apprendrez

Ce didacticiel introduit les concepts suivants :

- Utilisation du composant Canvas pour le dessin.
 - Gestion des événements de toucher et de glisser sur la surface de l'appareil.
 - Contrôle de la disposition de l'écran avec les composants de disposition.
 - Utilisation de gestionnaires d'événements dotés d'arguments.
 - Définir des variables pour mémoriser des éléments tels que la taille du point choisi par l'utilisateur.
- dessin.

Commencer

Accédez au site Web App Inventor. Démarrez un nouveau projet et nommez-le « PaintPot ». Cliquez sur Connecter et configurez votre appareil (ou émulateur) pour des tests en direct (voir <http://appinventor.mit.edu/explore/ai2/setup> pour obtenir de l'aide sur cette configuration).

Ensuite, à droite du Designer, accédez au panneau Propriétés et modifiez le titre de l'écran en « PaintPot ». Vous devriez voir ce changement sur l'appareil, avec le nouveau titre affiché dans la barre de titre de votre application.

Si vous craignez de confondre le nom de votre projet et le nom d'écran, ne vous inquiétez pas ! Il existe trois noms clés dans App Inventor :

- Le nom que vous choisissez pour votre projet au fur et à mesure que vous y travaillez. Ce sera également le nom de l'application lorsque vous la conditionnerez pour l'appareil. Notez que vous pouvez cliquer sur Projet et Enregistrer sous dans le Concepteur de composants pour démarrer une nouvelle version ou renommer un projet.
- Le nom du composant, Screen1, que vous verrez dans le panneau Composants. Vous ne pouvez pas modifier le nom de cet écran initial dans la version actuelle d'App Inventor.
- Le titre de l'écran, c'est ce que vous verrez dans la barre de titre de l'application. Cela commence par être le même que le nom du composant, Screen1, que vous avez utilisé dans HelloPurr. Mais vous pouvez le modifier, comme nous venons de le faire pour PaintPot.

Conception des composants

Vous utiliserez ces composants pour créer l'application :

- Trois composants Bouton pour sélectionner la peinture rouge, bleue ou verte, et un Composant HorizontalArrangement pour les organiser.
- Un composant Bouton pour effacer le dessin, deux pour modifier la taille des points dessinés et un pour appeler l'appareil photo pour prendre une photo.
- Un composant Canvas , qui est la surface de dessin. La toile a un Propriété BackgroundImage , que vous définirez sur le fichier kitty.png à partir du didacticiel HelloPurr du chapitre 1. Plus loin dans ce chapitre, vous modifierez l'application afin que l'arrière-plan puisse être défini sur une photo prise par l'utilisateur.

CRÉATION DES BOUTONS DE COULEUR

Tout d'abord, créez les trois boutons de couleur en suivant ces instructions :

1. Faites glisser un composant Button sur le Viewer, changez son attribut Text en « Rouge », puis rendez son BackgroundColor rouge.
2. Dans le Viewer, dans la liste des composants, cliquez sur Button1 pour le mettre en surbrillance (cela pourrait déjà mis en surbrillance) et cliquez sur Renommer pour changer son nom de Button1 à RedButton. Notez que les espaces ne sont pas autorisés dans les noms de composants. Il est donc courant de mettre en majuscule la première lettre de chaque mot du nom.
3. De même, créez deux autres boutons pour le bleu et le vert, nommés BlueButton et GreenButton, en les plaçant verticalement sous le bouton rouge. Vérifiez votre travail jusqu'à présent par rapport à la figure 2-2.

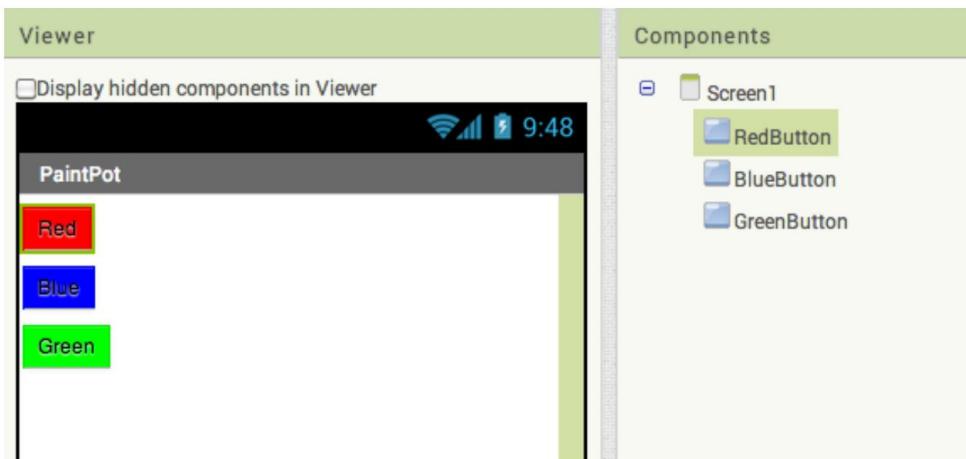


Figure 2-3. Le Viewer montrant les trois boutons créés

Notez que dans ce projet, vous modifiez les noms des composants plutôt plutôt que de les laisser comme noms par défaut, comme vous l'avez fait avec HelloPurr. Utiliser davantage

des noms significatifs rendent vos projets plus lisibles, et cela vous aidera vraiment lorsque vous passerez à l'éditeur de blocs et que vous devrez faire référence aux composants par leur nom. Dans ce livre, nous utiliserons la convention selon laquelle le nom du composant se termine par son type (par exemple, RedButton).



Testez votre application Si vous n'avez pas cliqué sur Connecter et connecté votre appareil, faites-le maintenant et vérifiez à quoi ressemble votre application sur votre appareil ou dans l'émulateur.

UTILISER DES ARRANGEMENTS POUR DE MEILLEURES DISPOSITIONS

Vous devriez maintenant avoir trois boutons empilés les uns sur les autres. Mais pour cette application, vous souhaitez qu'ils soient tous alignés côté à côté, en haut de l'écran, comme le montre la figure 2-3. Pour ce faire, vous utilisez un composant HorizontalArrangement :

1. Depuis le tiroir Disposition de la palette, faites glisser un composant HorizontalArrangement et placez-le sous les boutons.
2. Dans le panneau Propriétés, modifiez la largeur du HorizontalArrangement en « Fill parent » afin qu'il remplisse toute la largeur de l'écran.
3. Déplacez les trois boutons un par un dans l' arrangement horizontal composant. Astuce : vous verrez une ligne verticale bleue indiquant où ira la pièce que vous faites glisser.

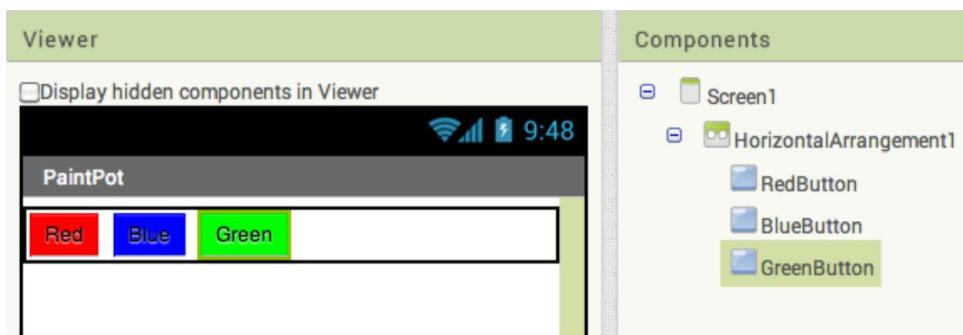


Figure 2-4. Les trois boutons dans une disposition horizontale

Si vous regardez dans la liste des composants du projet, vous verrez les trois boutons en retrait sous le composant HorizontalArrangement . Cela indique que les boutons sont désormais des sous-composants du composant HorizontalArrangement . Notez que tous les composants sont en retrait sous Screen1.

Vous pouvez centrer toute la rangée de boutons sur l'écran en modifiant la valeur de Screen1 Propriété AlignHorizontal sur « Centre ».



Testez votre application Sur l'appareil, vous devriez voir vos trois boutons alignés en haut de l'écran, même si les choses peuvent ne pas ressembler exactement à ce qu'elles sont sur le Designer. Par exemple, le contour autour de HorizontalArrangement apparaît dans la visionneuse mais pas sur l'appareil.

En général, vous utilisez la disposition des écrans pour créer des dispositions simples verticales, horizontales ou tabulaires. Vous pouvez également créer des présentations plus complexes en insérant (ou en imbriquant) des composants de disposition d'écran les uns dans les autres.

AJOUT DE LA TOILE

L'étape suivante consiste à configurer le canevas sur lequel le dessin aura lieu :

1. Depuis le tiroir Dessin et Animation de la Palette, faites glisser un composant Canvas sur le Visualiseur. Changez son nom en DrawingCanvas. Définissez sa largeur sur « Remplir le parent » afin qu'elle s'étende sur toute la largeur de l'écran. Réglez sa hauteur sur 300 pixels, ce qui laissera de la place aux deux rangées de boutons.
2. Si vous avez suivi le tutoriel HelloPurr (Chapitre 1), vous avez déjà téléchargé le fichier kitty.png . Si ce n'est pas le cas, vous pouvez le télécharger sur <http://appinventor.org/bookFiles/HelloPurr/kitty.png> .
3. Définissez BackgroundImage de DrawingCanvas sur le fichier kitty.png . Dans le Dans la section Propriétés du Concepteur de composants, BackgroundImage sera défini sur Aucun. Cliquez sur le champ et téléchargez le fichier kitty.png .
4. Définissez PaintColor de DrawingCanvas sur rouge afin que lorsque l'utilisateur démarre l'application mais n'a pas encore cliqué sur un bouton, sa couleur par défaut soit le rouge. Vérifiez que ce que vous avez construit ressemble à la figure 2-4.

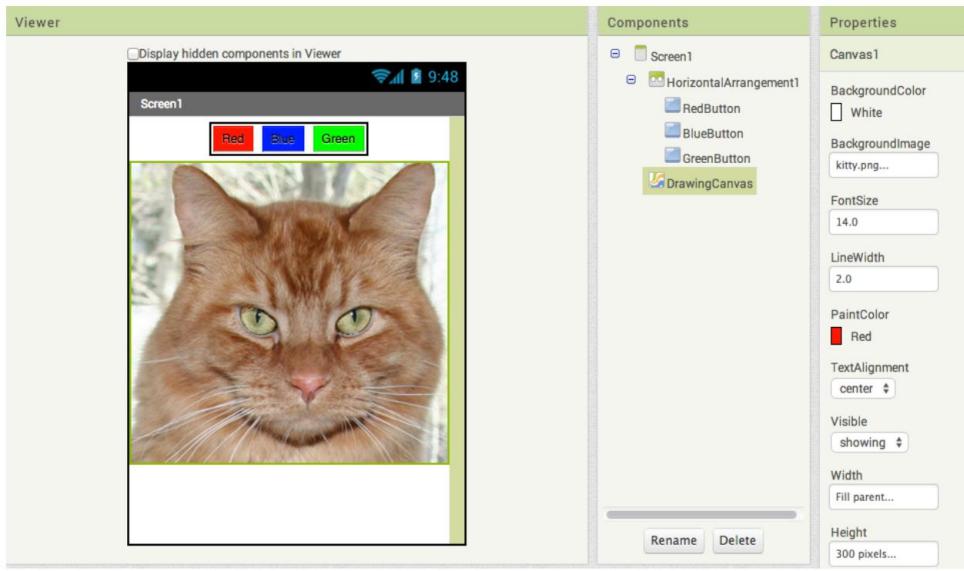


Figure 2-5. Le composant DrawingCanvas a une BackgroundImage de l'image du chaton

DISPOSITION DES BOUTONS INFÉRIEURS ET DU COMPOSANT DE LA CAMÉRA

1. Depuis la palette, faites glisser un deuxième arrangement horizontal et placez-le sous le canevas.
Ensuite, faites glisser deux autres composants Button sur l'écran et placez-les dans cet arrangement horizontal inférieur. Changez le nom du premier bouton en TakePictureButton et sa propriété Text en « Take Picture ». Changez le nom du deuxième bouton en WipeButton et sa propriété Text en « Wipe ».
2. Faites glisser deux autres composants Button de la palette vers l' arrangement horizontal, en les plaçant à côté de WipeButton.
3. Nommez les boutons BigButton et SmallButton et définissez leur texte sur « Big Dots ». et « Petits points », respectivement.
4. Depuis le tiroir Média, faites glisser un composant Caméra dans la Visionneuse. Ce sera apparaissent dans la zone des composants non visibles.

Vous avez maintenant terminé les étapes pour définir l'apparence de votre application, qui devrait ressembler à la figure 2-5.

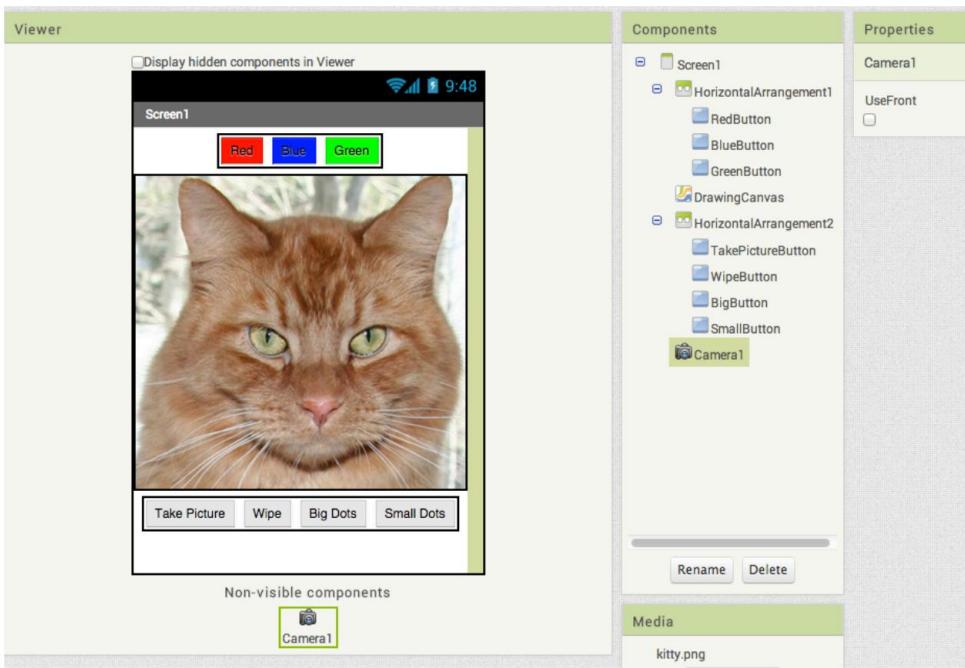


Figure 2-6. L'interface utilisateur complète pour PaintPot



Testez votre application Vérifiez l'application sur l'appareil. L'image du chat apparaît-elle maintenant sous la rangée supérieure de boutons ? La rangée inférieure de boutons est-elle en place sous l'image ?

Ajout de comportements aux composants

L'étape suivante consiste à définir le comportement des composants. Créer un programme de peinture peut sembler fastidieux, mais soyez assuré qu'App Inventor a fait le gros du travail pour vous : il existe des blocs faciles à utiliser pour gérer les actions de toucher et de glissement de l'utilisateur, ainsi que pour dessiner et prendre des photos.

Dans Designer, vous avez ajouté un composant Canvas nommé DrawingCanvas. Comme tous les composants Canvas, DrawingCanvas possède un événement Touched et un événement Dragged . Vous allez programmer l' événement DrawingCanvas.Touched de sorte qu'un cercle soit dessiné en réponse au fait que l'utilisateur touche son doigt sur l'écran. Vous allez programmer l' événement DrawingCanvas.Dragged de sorte qu'une ligne soit tracée lorsque l'utilisateur fait glisser son doigt sur le canevas. Vous programmerez ensuite les boutons pour changer la couleur du dessin, effacer le canevas et changer l'arrière-plan du canevas en une photo prise avec l'appareil photo.

AJOUTER L'ÉVÉNEMENT TOUCH POUR DESSINER UN POINT

Tout d'abord, vous allez organiser les choses de manière à ce que lorsque vous touchez le DrawingCanvas, vous dessinez un point au point de contact :

1. Dans l'éditeur de blocs, sélectionnez le tiroir du DrawingCanvas , puis faites glisser le bloc DrawingCanvas.Touched vers l'espace de travail. Le bloc a des paramètres pour x, y et touchedSprite, comme le montre la figure 2-6. Ces paramètres fournissent des informations sur l'emplacement du toucher.

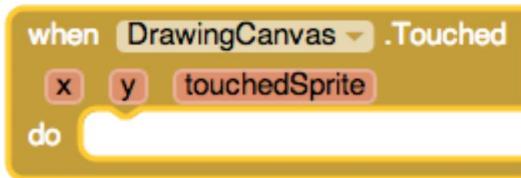


Figure 2-7. L'événement contient des informations sur l'endroit où l'écran est touché



Remarque Si vous avez terminé l'application HelloPurr au chapitre 1, vous êtes familier avec les événements Button.Click , mais pas avec les événements Canvas. Les événements Button.Click sont assez simples car il n'y a rien à savoir sur l'événement autre que le fait qu'il s'est produit.

Cependant, certains gestionnaires d'événements fournissent des informations sur l'événement appelées arguments. L' événement DrawingCanvas.Touched fournit les coordonnées x et y du toucher dans DrawingCanvas. Il vous permet également de savoir si un objet dans le

DessinToile (dans App Inventor, cela s'appelle un sprite) était touché, mais nous n'en aurons pas besoin avant le chapitre 3. Les coordonnées x et y sont les arguments que nous utiliserons pour identifier où le l'utilisateur a touché l'écran. Nous pouvons alors dessiner le point à ce moment-là position.

2. Dans le tiroir DrawingCanvas, faites glisser une commande DrawingCanvas.DrawCircle et placez-la dans le gestionnaire d'événements DrawingCanvas.Touched , comme illustré dans la figure 2-7.

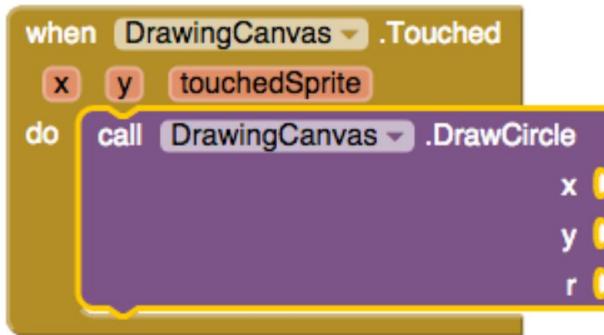


Figure 2-8. Lorsque l'utilisateur touche le canevas, l'application dessine un cercle

Sur le côté droit du bloc DrawingCanvas.DrawCircle , vous verrez trois sockets pour les arguments que nous devons brancher : x, y et r. Les arguments x et y spéfient l'emplacement où le cercle doit être dessiné, et r détermine le rayon (ou la taille) du cercle. Ce gestionnaire d'événements peut être un peu déroutant car l'événement DrawingCanvas.Touched a également un x et un y ; gardez simplement à l'esprit que les x et y de l'événement DrawingCanvas.Touched indiquent l'endroit où l'utilisateur a touché, tandis que les x et y de l'événement DrawingCanvas.DrawCircle sont des sockets ouverts vous permettant de spécifier où le cercle doit être dessiné. Étant donné que vous souhaitez dessiner le cercle où l'utilisateur a touché, vous intégrerez les valeurs x et y de DrawingCanvas.Touched comme valeurs à utiliser pour les paramètres x et y dans DrawingCanvas.DrawCircle.



Remarque Vous pouvez accéder aux valeurs des paramètres d'événement en passant la souris dessus dans le bloc when , comme illustré dans la figure 2-8.

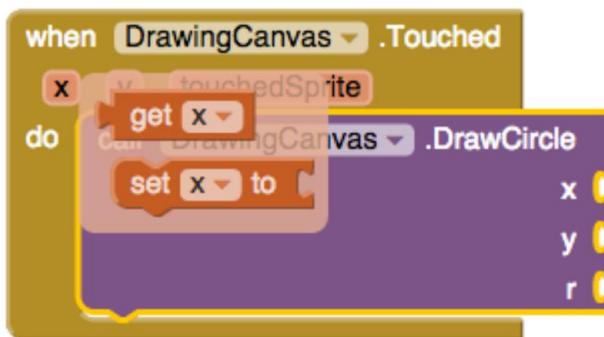


Figure 2-9. Passez la souris sur un paramètre d'événement pour faire glisser un bloc get permettant d'obtenir la valeur

3. Faites glisser les blocs pour les valeurs x et y et branchez-les dans les prises le bloc DrawingCanvas.DrawCircle , comme illustré dans la figure 2-9.



Figure 2-10. L'application sait maintenant où dessiner (x, y), mais nous devons encore préciser la taille du cercle.

4. Vous devez maintenant spécifier le rayon, r, du cercle à dessiner. Le rayon est mesuré en pixels, qui est le plus petit point pouvant être dessiné sur l'écran d'un appareil. Pour l'instant, réglez-le sur 5 : cliquez dans une zone vide de l'écran, tapez 5 puis appuyez sur Retour (cela créera automatiquement un bloc numérique), puis branchez-le sur la prise r . Lorsque vous le faites, la case jaune dans le coin inférieur gauche reviendra à 0 car tous les emplacements sont désormais remplis. La figure 2-10 illustre à quoi devrait ressembler le gestionnaire d'événements DrawingCanvas.Touched final .



Remarque Si vous tapez un « 5 » dans l'éditeur de blocs et appuyez sur Retour, un bloc numérique contenant un « 5 » apparaîtra. Cette fonctionnalité est appelée blocage de type : si vous commencez à taper, l'éditeur de blocs affiche une liste de blocs dont les noms correspondent à ce que vous tapez ; si vous tapez un nombre, cela crée un bloc numérique.

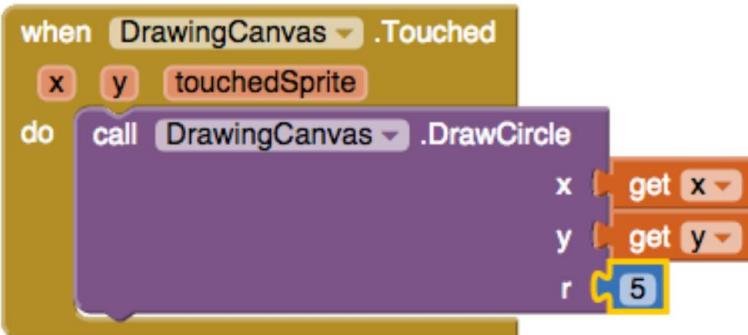


Figure 2-11. Lorsque l'utilisateur touche le DrawingCanvas, un cercle de rayon 5 est dessiné à l'emplacement du toucher (x,y)



Testez votre application Essayez ce que vous avez jusqu'à présent sur l'appareil.
Lorsque vous touchez DrawingCanvas, votre doigt doit laisser un point à chaque endroit que vous touchez. Les points seront rouges si vous définissez la propriété DrawingCanvas.PaintColor sur rouge dans le Concepteur de composants (sinon, ils sont noirs, car c'est la valeur par défaut).

AJOUT DE L'ÉVÉNEMENT DRAG QUI TIRE UNE LIGNE

Ensuite, vous ajouterez le gestionnaire d'événements glisser. Voici la différence entre un toucher et un glissement :

- Un toucher , c'est lorsque vous placez votre doigt sur le DrawingCanvas puis que vous le soulevez sans le déplacer.
- Un glissement , c'est lorsque vous placez votre doigt sur le DrawingCanvas et que vous le déplacez. tout en le gardant en contact avec l'écran.

Dans un programme de peinture, faire glisser votre doigt en arc de cercle sur l'écran semble tracer une ligne courbe qui suit le chemin de votre doigt. Ce que vous faites en réalité, c'est dessiner de nombreuses petites lignes droites ; chaque fois que vous bougez votre doigt, même un tout petit peu, vous tracez la ligne depuis la dernière position de votre doigt jusqu'à sa nouvelle position.

1. Depuis le tiroir DrawingCanvas, faites glisser le bloc DrawingCanvas.Dragged vers l'espace de travail.

Vous devriez voir le gestionnaire d'événements tel qu'il est illustré dans la figure 2-11.

L' événement DrawingCanvas.Dragged est livré avec les arguments suivants :

- startX, startY : la position de votre doigt au point où le glisser commencé.

- currentX, currentY : la position actuelle de votre doigt
- prevX, prevY : la position immédiatement précédente de votre doigt.
- draggedSprite : une valeur booléenne, elle sera vraie si l'utilisateur glisse directement sur un sprite d'image. Nous n'utiliserons pas cet argument dans ce tutoriel.

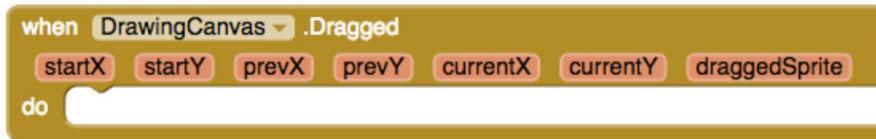


Figure 2-12. Un événement Dragged a encore plus d'arguments qu'un événement Touched

2. Dans le tiroir DrawingCanvas, faites glisser le bloc DrawingCanvas.DrawLine dans le bloc DrawingCanvas.Dragged , comme illustré dans la figure 2-12.

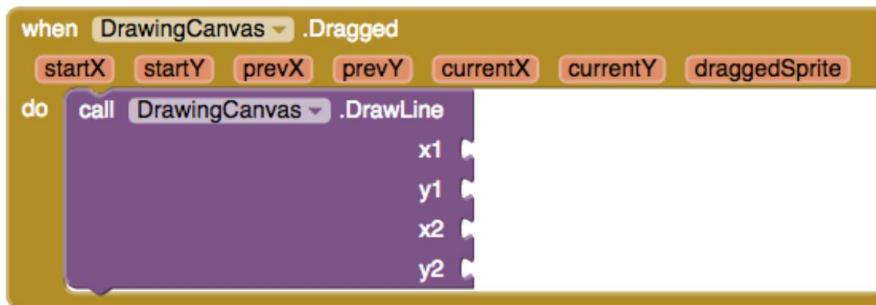


Figure 2-13. Ajout de la possibilité de tracer des lignes

Le bloc DrawingCanvas.DrawLine a quatre arguments, deux pour chaque point qui détermine la ligne. (x_1, y_1) est un point, tandis que (x_2, y_2) est l'autre. Pouvez-vous déterminer quelles valeurs doivent être insérées dans chaque argument ? N'oubliez pas que l'événement Dragged sera appelé plusieurs fois lorsque vous faites glisser votre doigt sur DrawingCanvas. L'application trace une petite ligne à chaque fois que votre doigt bouge, de ($prevx, prevy$) à ($currentX, currentY$).

3. Faites glisser les blocs get pour les arguments dont vous avez besoin. A obtenir prevX et obtenir prevY doit être branché respectivement sur les prises x1 et y1. Un get currentX et un get currentY doivent être branchés respectivement sur les prises x2 et y2, comme indiqué dans la figure 2-13.

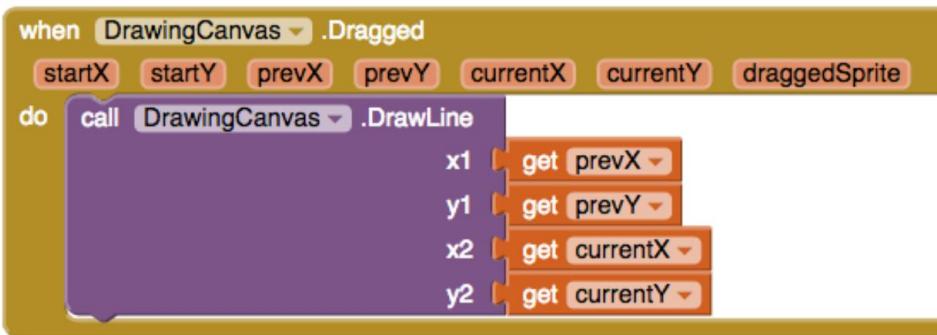


Figure 2-14. Au fur et à mesure que l'utilisateur fait glisser, l'application tracera une ligne entre le point précédent et le point actuel.



Testez votre application Essayez ce comportement sur l'appareil. Faites glisser votre doigt sur l'écran pour tracer des lignes et des courbes. Touchez l'écran pour créer des points.

CHANGER LA COULEUR

L'application que vous avez créée permet à l'utilisateur de dessiner, mais jusqu'à présent, tout était en rouge. Ensuite, ajoutez des gestionnaires d'événements pour les boutons de couleur afin que les utilisateurs puissent modifier la couleur de la peinture, et un autre pour le WipeButton afin de leur permettre d'effacer l'écran et de recommencer.

Dans l'éditeur de blocs :

1. Ouvrez le tiroir de RedButton et faites glisser le bloc RedButton.Click .
2. Ouvrez le tiroir DrawingCanvas. Faites glisser l' ensemble DrawingCanvas.PaintColor à bloquer (vous devrez peut-être faire défiler la liste des blocs dans le tiroir pour le trouver) et placez-le dans la section « faire » de RedButton.Click.
3. Ouvrez le tiroir Couleurs et faites glisser le bloc pour la couleur rouge et branchez-le dans l' ensemble DrawingCanvas.PaintColor à bloquer.
4. Répétez les étapes 2 à 4 pour les boutons bleu et vert.
5. Le dernier bouton à configurer est WipeButton. Faites glisser un WipeButton.Click depuis le tiroir WipeButton. Dans le tiroir DrawingCanvas, faites glisser DrawingCanvas.Clear et placez-le dans le bloc WipeButton.Click . Confirmez que vos blocs apparaissent comme dans la figure 2-14.

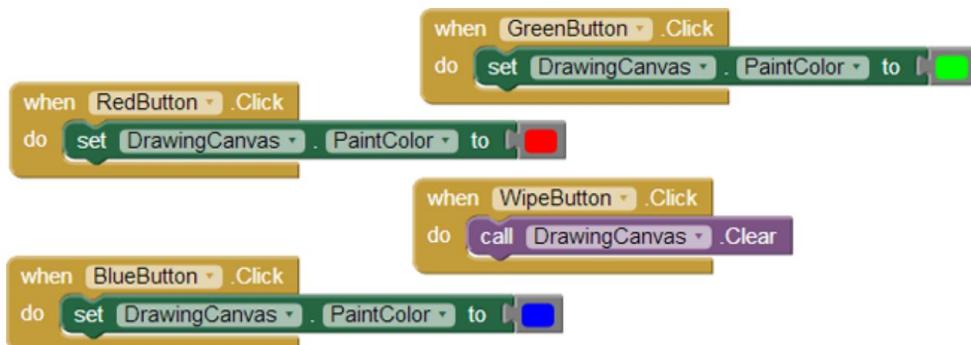


Figure 2-15. Cliquer sur les boutons de couleur modifie la PaintColor de DrawingCanvas ; cliquer sur Wipe efface l'écran



Testez votre application Essayez les comportements en cliquant sur chacun des boutons de couleur et voyez si vous pouvez dessiner des cercles de couleurs différentes. Ensuite, cliquez sur le bouton Effacer pour voir si le canevas est effacé.

LAISSER L'UTILISATEUR PRENDRE UNE PHOTO

Les applications App Inventor peuvent interagir avec les fonctionnalités puissantes d'un appareil Android, y compris l'appareil photo. Pour pimenter l'application, laissez l'utilisateur définir l'arrière-plan du dessin en prenant une photo avec l'appareil photo.

Le composant Caméra comporte deux blocs clés. Le bloc Camera.TakePicture lance l'application appareil photo sur l'appareil. L'événement Camera.AfterPicture est déclenché lorsque l'utilisateur a fini de prendre la photo. Vous ajouterez des blocs dans le gestionnaire d'événements Camera.AfterPicture pour définir DrawingCanvas.BackgroundImage sur l'image que l'utilisateur vient de prendre.

1. Ouvrez le tiroir TakePictureButton et faites glisser le gestionnaire d'événements TakePictureButton.Click dans l'espace de travail.
2. Depuis Camera1, faites glisser Camera1.TakePicture et placez-le dans le gestionnaire d'événements TakePictureButton.Click .
3. Depuis Camera1, faites glisser le gestionnaire d'événements Camera1.AfterPicture dans le espace de travail.
4. Depuis DrawingCanvas, faites glisser l' ensemble DrawingCanvas.BackgroundImage pour bloquer et placez-le dans le gestionnaire d'événements Camera1.AfterPicture .
5. Camera1.AfterPicture a un argument nommé image, qui est la photo qui vient d'être prise. Vous pouvez en obtenir une référence en utilisant un bloc get du

Bloc Camera1.AfterPicture , puis branchez-le dans DrawingCanvas.BackgroundImage.

Les blocs devraient ressembler à la figure 2-15.

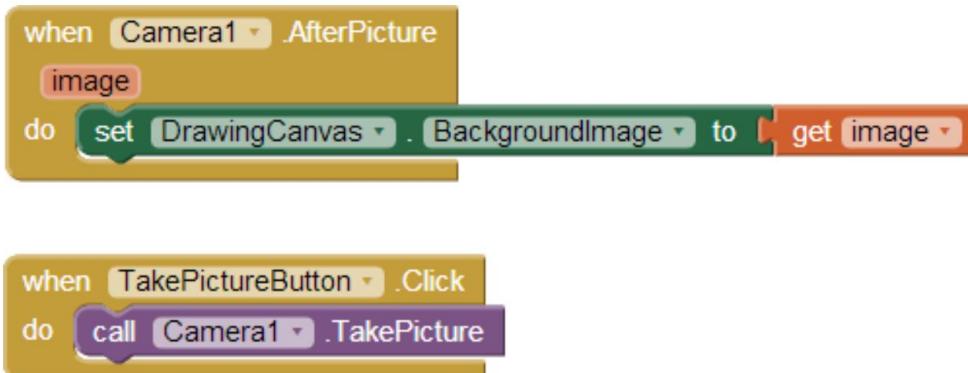


Figure 2-16. Lorsque la photo est prise, elle est définie comme image d'arrière-plan pour DrawingCanvas.



Testez votre application Essayez ce comportement en cliquant sur Prendre une photo sur votre appareil et en prenant une photo. L'image du chat devrait devenir la photo que vous venez de prendre, et vous pourrez ensuite dessiner sur cette photo. (S'inspirer du professeur Wolber est un passe-temps favori de ses étudiants, comme le montre la figure 2-16.)

MODIFIER LA TAILLE DES POINTS

La taille des points dessinés sur DrawingCanvas est déterminée lors de l'appel à DrawingCanvas.DrawCircle, où l'argument de rayon r est défini sur 5. Pour modifier la taille, vous pouvez saisir une valeur différente pour r. Pour tester cela, essayez de remplacer le 5 par un 10 et de le tester sur l'appareil pour voir à quoi il ressemble.

Le problème ici est que l'utilisateur est limité à la taille que vous définissez dans le rayon argument. Que se passe-t-il si l'utilisateur souhaite modifier la taille des points ? Modifions le programme pour que l'utilisateur, et pas seulement le programmeur, puisse modifier la taille des points. Nous allons programmer le bouton intitulé « Gros points » pour changer la taille du point à 8, et programmer le bouton intitulé « Petits points » pour ajuster la taille à 2.

Pour utiliser différentes valeurs pour l'argument rayon, l'application doit savoir laquelle nous voulons appliquer. Nous devons lui demander d'utiliser une valeur spécifique, et il doit stocker (ou mémoriser) cette valeur d'une manière ou d'une autre pour pouvoir continuer à l'utiliser. Lorsque votre application doit mémoriser quelque chose qui n'est pas une propriété, vous pouvez définir une variable. Une variable est une cellule mémoire ; vous pouvez la considérer comme un compartiment dans lequel vous pouvez stocker des données qui peuvent varier,

qui dans ce cas est la taille actuelle du point (pour plus d'informations sur les variables, voir le chapitre 16).



Figure 2-17. L'application PaintPot avec une photo « annotée » du professeur Wolber

Commençons par définir une variable nommée `dotSize` :

1. Dans l'éditeur de blocs, depuis le tiroir Variables des blocs intégrés, faites glisser un nom global d'initialisation à bloquer. Dans le bloc d'initialisation, remplacez le texte « nom » par « `dotSize` ».
2. Notez que le `dotSize` global initialisé à bloquer a un socket ouvert. C'est ici que vous pouvez spécifier la valeur initiale de la variable ou la valeur par défaut au démarrage de l'application. (Cela est souvent appelé « initialisation d'une variable » en termes de programmation.) Pour cette application, initialisez le `dotSize` à 2 en créant un bloc numéro 2 (utilisez la fonction de blocage de type : tapez un « 2 » dans l'éditeur de blocs, puis appuyez sur Return), puis en le branchant sur `initialize global dotSize`, comme le montre la figure 2-17.



Figure 2-18. Initialisation de la variable `dotSize` avec une valeur de 2

RÉFÉRENCEMENT DE LA VARIABLE DOTSIZE DANS DRAWCIRCLE

Ensuite, nous souhaitons modifier l'argument de `DrawingCanvas.DrawCircle` dans le gestionnaire d'événements `DrawingCanvas.Touched` afin qu'il utilise la valeur de `dotSize` plutôt que

toujours en utilisant un numéro fixe. (Il peut sembler que nous avons "fxé" dotSize à la valeur 2 plutôt que de l'avoir rendu variable parce que nous l'avons initialisé de cette façon, mais vous verrez dans une minute comment nous pouvons changer la valeur de dotSize et, par conséquent, changer la valeur de dotSize. taille du point dessiné.)

1. Faites glisser un bloc get de la taille de point globale d'initialisation vers le bloc. Vous devriez voir un bloc get global dotSize qui fournit la valeur de la variable.
2. Accédez au bloc DrawingCanvas.DrawCircle , faites glisser le bloc numéro 5 hors de l'emplacement r , puis placez-le dans la corbeille. Ensuite, remplacez-le par le bloc get global dotSize (voir Figure 2-18). Lorsque l'utilisateur touche le DrawingCanvas, l'application déterminera désormais le rayon à partir de la variable dotSize.

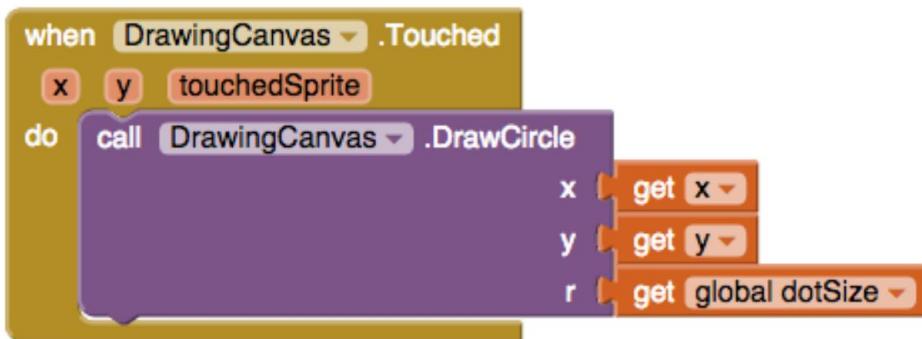


Figure 2-19. Désormais, la taille de chaque cercle dépend de ce qui est stocké dans la variable dotSize

MODIFIER LA VALEUR DE LA TAILLE DES POINTS

Votre application dessinera désormais des cercles dont la taille est basée sur la valeur de la variable dotSize, mais vous avez toujours besoin de code pour que dotSize change (il reste actuellement à 2) en fonction du choix de l'utilisateur. Vous allez implémenter ce comportement en programmant les gestionnaires d'événements SmallButton.Click et BigButton.Click :

1. Faites glisser un gestionnaire d'événements SmallButton.Click du tiroir SmallButton. Ensuite, passez la souris sur la « taille du point » dans le bloc global d'initialisation et faites glisser la taille du point globale définie à bloquer. Branchez-le sur SmallButton.Click. Enfin, créez un bloc numéro 2 et branchez-le dans le dotSize global défini pour bloquer.
2. Créez un gestionnaire d'événements similaire pour BigButton.Click, mais définissez dotSize sur 8. Les deux gestionnaires d'événements devraient maintenant apparaître dans l'éditeur de blocs, comme le montre la figure 2-19.



Remarque Le « global » dans set global dotSize fait référence au fait que la variable peut être utilisée dans tous les gestionnaires d'événements du programme (globalement). Dans App Inventor, vous pouvez également définir des variables « locales » pour une partie particulière du programme (voir le chapitre 21 pour plus de détails).

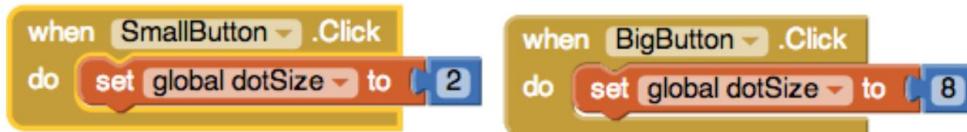


Figure 2-20. Cliquer sur les boutons modifie la taille du point ; les touches par la suite dessineront à cette taille



Testez votre application Essayez de cliquer sur les boutons de taille, puis de toucher DrawingCanvas. Les cercles sont-ils dessinés avec des tailles différentes ? Les lignes sont-elles ? La taille de la ligne ne devrait pas changer car vous avez programmé dotSize pour qu'il soit utilisé uniquement dans le bloc DrawingCanvas.DrawCircle. Sur cette base, pouvez-vous penser à la manière dont vous modifieriez vos blocs afin que les utilisateurs puissent également modifier la taille de la ligne ? (Indice : DrawingCanvas possède une propriété nommée LineWidth.)

L'application complète : PaintPot

La figure 2-20 illustre notre application PaintPot terminée.

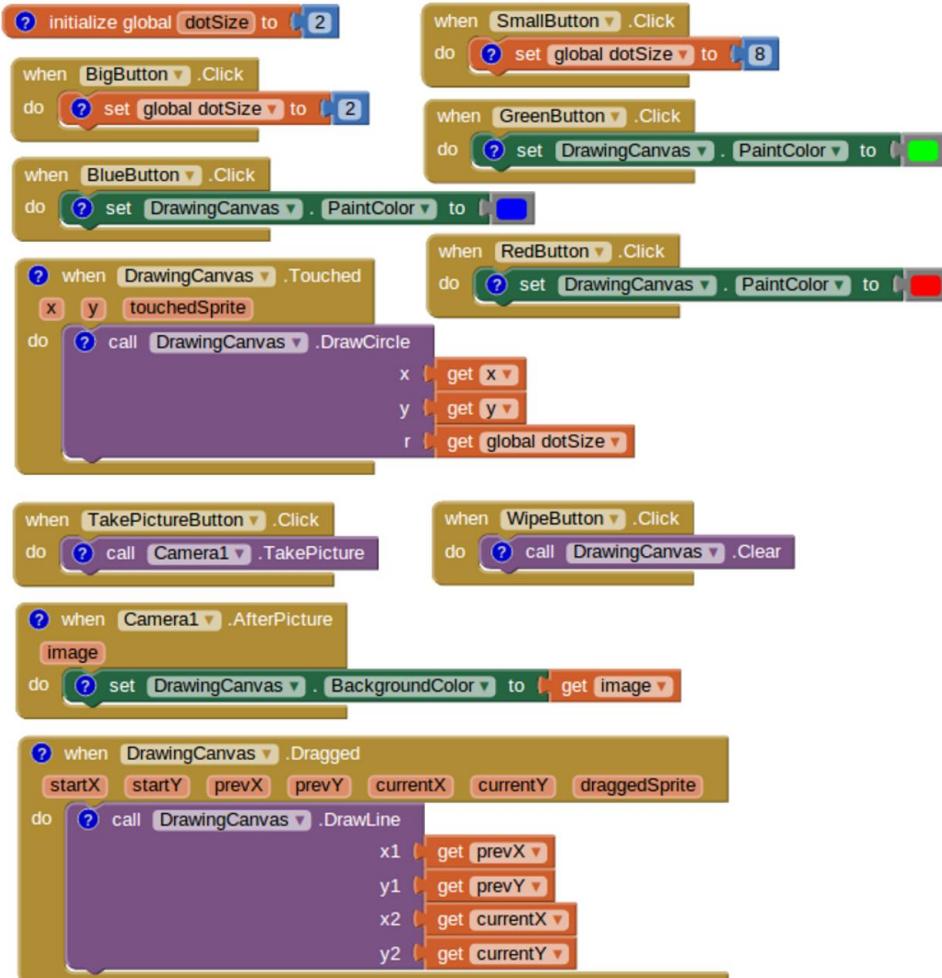


Figure 2-21. Le dernier ensemble de blocs pour PaintPot

Variantes

Voici quelques variantes que vous pouvez explorer :

- L'interface utilisateur de l'application ne fournit pas beaucoup d'informations sur les paramètres actuels (par exemple, la seule façon de connaître la taille ou la couleur actuelle du point est de dessiner quelque chose). Modifiez l'application pour que ces paramètres soient affichés à l'écran.
- Laissez l'utilisateur spécifier des valeurs autres que 2 et 8 pour la taille du point à l'aide d'un curseur composant.

Résumé

Voici quelques-unes des idées que nous avons abordées dans ce chapitre :

- Le composant Canvas vous permet de dessiner dessus. Il peut également détecter les touches et les glissements, et vous pouvez mapper ces événements aux fonctions de dessin.
- Vous pouvez utiliser des composants de disposition d'écran pour organiser la disposition des composants au lieu de simplement les placer les uns au-dessous des autres.
- Certains gestionnaires d'événements fournissent des informations sur l'événement, comme le coordonnées de l'endroit où l'écran a été touché. Ces informations sont représentées par des arguments. Lorsque vous faites glisser un gestionnaire d'événements comportant des arguments, App Inventor crée des éléments get et set dans le bloc à utiliser pour les référencier.
arguments.
- Vous créez des variables en utilisant l'initialisation du nom global aux blocs du Tiroir de variables. Les variables permettent à l'application de mémoriser des informations, telles que la taille des points, qui ne sont pas stockées dans une propriété de composant.
- Pour chaque variable que vous définissez, App Inventor fournit automatiquement une référence globale d'obtention qui donne la valeur de la variable, ainsi qu'un bloc global défini pour modifier la valeur de la variable. Pour y accéder, passez la souris sur le nom de la variable dans son bloc d'initialisation.

Ce chapitre a montré comment utiliser le composant Canvas pour un programme de peinture. Vous pouvez également l'utiliser pour programmer des animations, comme celles que vous trouverez dans les jeux 2D. Pour en savoir plus, consultez le jeu MoleMash au chapitre 3, le jeu Ladybug Chase au chapitre 5 et la discussion sur l'animation au chapitre 17.

MoleMash



Ce chapitre vous montre comment créer MoleMash, un jeu inspiré du classique d'arcade Whac-A-Mole™, dans lequel des créatures mécaniques sortent des trous et où les joueurs marquent des points lorsqu'ils réussissent à les frapper avec un maillet. MoleMash a été créé par un membre de l'équipe App Inventor, apparemment pour tester la fonctionnalité sprite (qu'elle a implémentée), mais en réalité parce qu'elle est fan du jeu.

Lorsqu'Ellen Spertus a rejoint l'équipe App Inventor chez Google, elle était impatiente d'ajouter un support pour la création de jeux, elle s'est donc portée volontaire pour implémenter des sprites. Le terme, inventé à l'origine pour décrire des créatures mythologiques telles que les fées et les lutins, a été adopté par la communauté informatique dans les années 1970 pour désigner des images capables de bouger sur un écran d'ordinateur (pour les jeux vidéo). Ellen a d'abord travaillé avec des sprites lorsqu'elle a participé à un camp informatique au début des années 1980 et qu'elle a programmé une TI 99/4. Son travail sur les sprites et MoleMash était motivé par une double nostalgie : à la fois pour les ordinateurs et les jeux de son enfance.

Ce que vous construirez

Pour l'application MoleMash illustrée à la figure 3-1, vous allez implémenter la fonctionnalité suivante :

- Une taupe apparaît à des endroits aléatoires sur l'écran, se déplaçant une fois par seconde.
- Taper sur la taupe fait vibrer l'appareil, incrémenter l'affichage des « hits » (en l'augmentant d'un) et déplace immédiatement la taupe vers un nouvel emplacement.
- En appuyant sur l'écran mais en manquant la taupe, l'affichage des « manqués » s'affiche.
- Appuyer sur un bouton Réinitialiser réinitialise le nombre de réussites et d'échecs.



Figure 3-1. L'interface utilisateur de MoleMash

Ce que vous apprendrez

Le didacticiel couvre les composants et concepts suivants :

- Le composant ImageSprite pour les images mobiles tactiles.
- Le composant Canvas , qui agit comme une surface sur laquelle placer le ImageSprite.
- Le composant Clock pour déplacer le sprite une fois par seconde.
- Le composant Son pour produire une vibration lorsque la taupe est tapée.
- Le composant Button pour démarrer une nouvelle partie.
- Procédures pour mettre en œuvre un comportement répété, comme déplacer la taupe.
- Générer des nombres aléatoires.
- Utiliser les blocs d'addition (+) et de soustraction (-).

Commencer

Connectez-vous au site Web App Inventor et démarrez un nouveau projet. Nommez-le « MoleMash » et définissez également le titre de l'écran sur « MoleMash ». Cliquez sur Connecter et connectez votre appareil ou émulateur pour des tests en direct.

Téléchargez l'image d'une taupe depuis <http://appinventor.org/bookFiles/MoleMash/taupe.png>. Dans Component Designer, dans la section Média, cliquez sur « Télécharger le fichier », recherchez l'emplacement du fichier sur votre ordinateur, puis téléchargez-le sur App Inventor.

Conception des composants

Vous utiliserez ces composants pour créer MoleMash :

- Un Canvas qui sert de terrain de jeu.
- Un ImageSprite qui affiche une image d'une taupe et peut se déplacer et sentir quand le grain de beauté est touché.
- Un son qui vibre lorsque le grain de beauté est touché.
- Des étiquettes qui affichent « Hits : », « Misses : » et le nombre réel de hits et manque.
- Arrangements horizontaux pour positionner correctement les étiquettes.
- Un bouton pour remettre à 0 le nombre de réussites et d'échecs.

- Une horloge pour faire bouger la taupe une fois par seconde.

Le tableau 3-1 montre tous les composants que vous utiliserez.

Tableau 3-1. La liste complète des composants pour MoleMash

Type de composant	Groupe de palettes	Comment vous l'appellerez	But
Toile	Dessin et Animation	Toile1	Le conteneur pour ImageSprite.
ImageSprite	Dessin et Animation	Taupe	L'utilisateur essaiera de toucher cela.
Bouton	Bouton de réinitialisation de l'interface utilisateur		L'utilisateur appuiera sur cette touche pour réinitialiser le score.
Horloge	Horloge de l'interface utilisateur1		Contrôlez le mouvement de la taupe.
Son	Médias	Son1	Vibrez lorsque le grain de beauté est touché.
Étiquette	Interface utilisateur HitsLabel		Afficher « Hits : ».
Étiquette	Interface utilisateur HitsCountLabel		Afficher le nombre de hits.
Disposition de l'arrangement horizontal		Horizontal Disposition1	Positionnez HitsLabel à côté de HitsCountLabel.
Étiquette	Interface utilisateur MissesLabel		Afficher « Manqués : ».
Étiquette	Interface utilisateur MissesCountLabel		Afficher le nombre d'échecs.
Disposition de l'arrangement horizontal		Horizontal Disposition2	Positionner MissesLabel à côté de MissesCountLabel.

PLACEMENT DES COMPOSANTS D'ACTION

Dans cette section, vous placerez les composants nécessaires à l'action du jeu. Dans le

Dans la section suivante, vous placerez les composants permettant d'afficher la partition.

1. Dans le tiroir Dessin et Animation, faites glisser un composant Canvas , en laissant avec le nom par défaut Canvas1. Définissez sa propriété Largeur sur « Fill parent » afin qu'elle est aussi large que l'écran et définissez sa hauteur sur 300 pixels.
2. Toujours dans le tiroir Dessin et Animation, faites glisser un ImageSprite composant, en le plaçant n'importe où sur Canvas1. Au bas des composants liste, cliquez sur Renommer et changez son nom en « Mole ». Définissez sa propriété Picture sur mole.PNG, que vous avez téléchargé plus tôt.
3. Dans le tiroir Interface utilisateur, faites glisser un composant Bouton et placez-le sous Canvas1. Renommez-le « ResetButton » et définissez sa propriété Text sur « Reset ».

46 Chapitre 3 : MoleMash

4. Également à partir du tiroir Interface utilisateur, faites glisser un composant Horloge . Il apparaîtra en bas du Viewer dans la section « Composants non visibles ».

5. Dans le tiroir Média, faites glisser un composant Son . Il apparaîtra également dans la section « Composants non visibles ».

Votre écran devrait maintenant ressembler à la figure 3-2 (bien que votre grain de beauté puisse être dans une position différente).

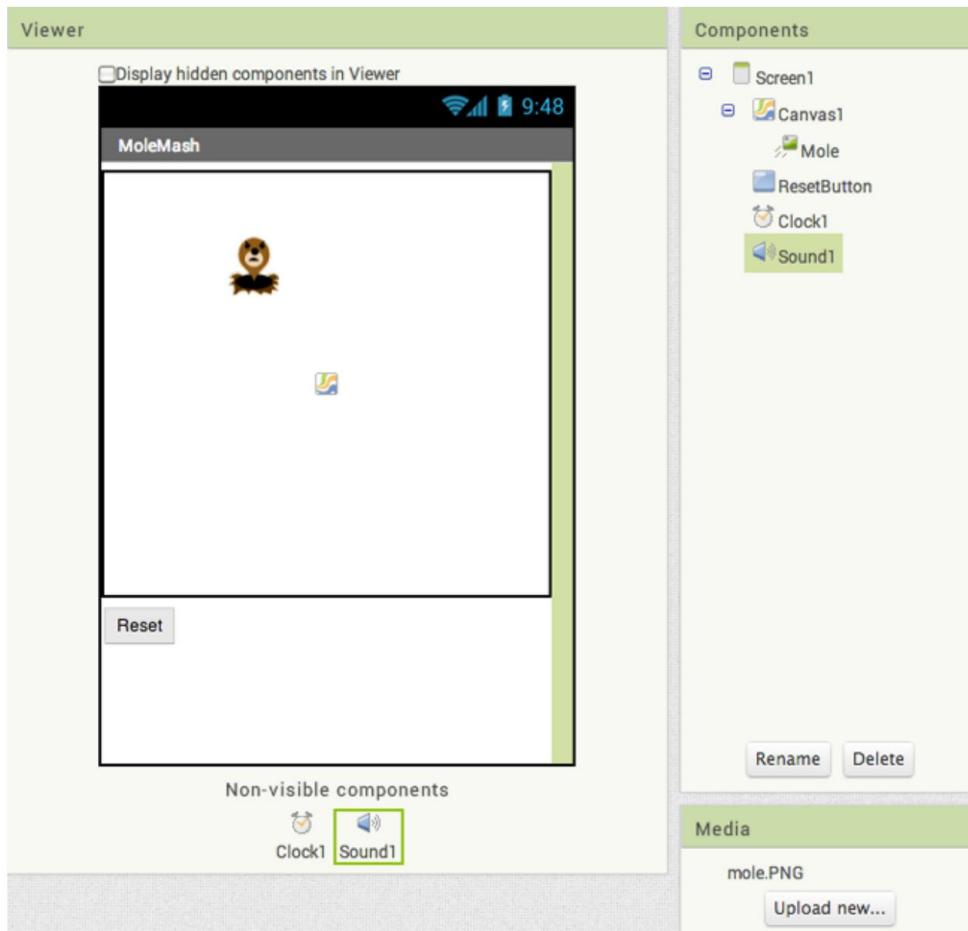


Figure 3-2. La vue Component Designer des composants « action »

PLACEMENT DES COMPOSANTS DE L'ÉTIQUETTE

Vous allez maintenant placer des composants pour afficher le score de l'utilisateur, plus précisément le nombre de succès et d'échecs.

1. Dans le tiroir Mise en page, faites glisser un HorizontalArrangement, en le plaçant sous le bouton et en conservant le nom par défaut de HorizontalArrangement1.
2. Dans le tiroir de l'interface utilisateur, faites glisser deux étiquettes dans Disposition horizontale1.
 - Renommez l' étiquette de gauche « HitsLabel » et définissez sa propriété Text sur « Hits : » (en veillant à inclure un espace après les deux points).
 - Renommez l' étiquette de droite « HitsCountLabel » et définissez sa propriété Text sur la valeur numéro 0.
3. Faites glisser un deuxième HorizontalArrangement, en le plaçant sous HorizontalArrangement1.
4. Faites glisser deux étiquettes dans HorizontalArrangement2.
 - Renommez l' étiquette de gauche « MissesLabel » et définissez sa propriété Text sur « Misses : » (en veillant à inclure un espace après les deux points).
 - Renommez l' étiquette de droite « MissesCountLabel » et définissez sa propriété Text sur le chiffre 0.

Votre écran devrait maintenant ressembler à quelque chose comme la figure 3-3.

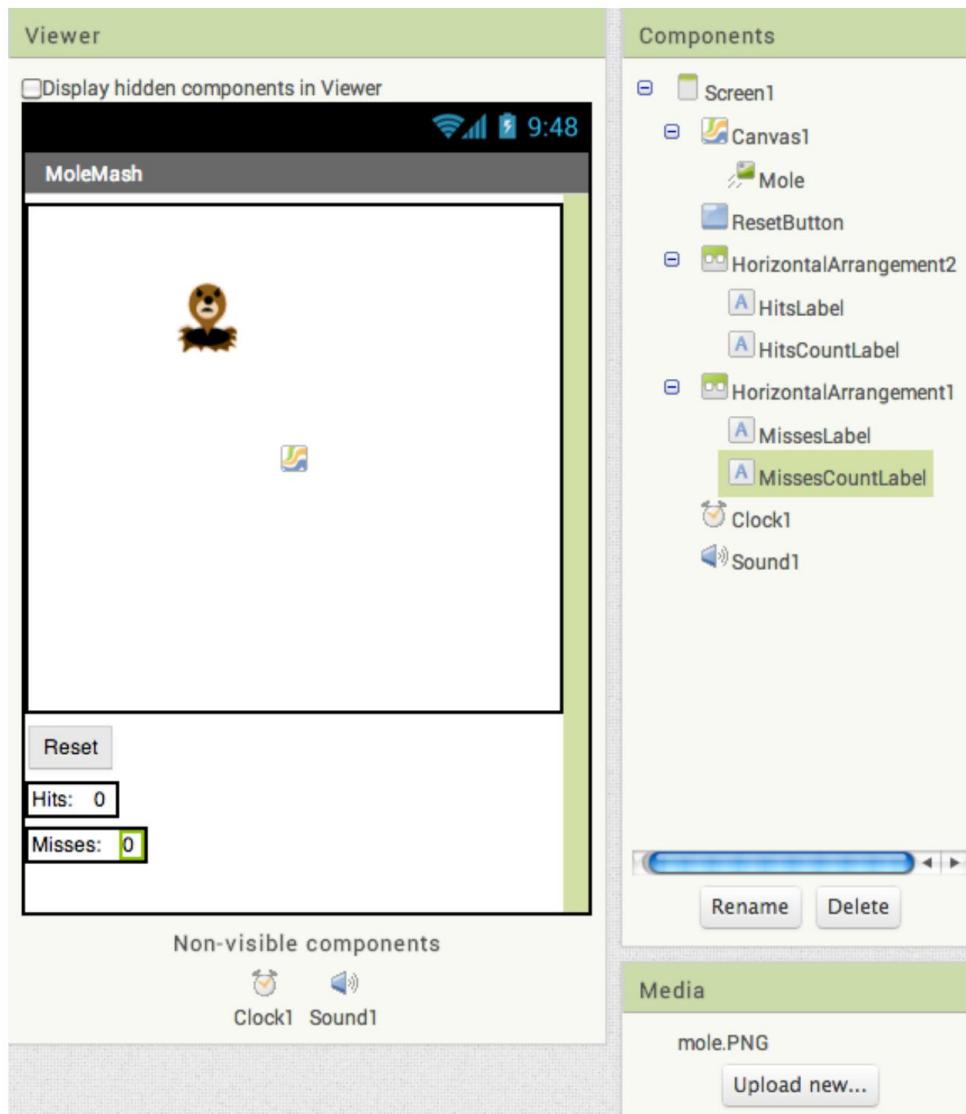


Figure 3-3. La vue Component Designer de tous les composants MoleMash

Ajout de comportements aux composants

Après avoir créé les composants précédents, passons à l'éditeur de blocs pour implémenter le comportement du programme. Plus précisément, nous voulons que la taupe se déplace vers un emplacement aléatoire sur la toile chaque seconde. L'objectif de l'utilisateur est d'appuyer sur la taupe partout où elle apparaît, et l'application affichera le nombre de fois où l'utilisateur frappe ou manque la taupe. (Remarque : nous vous recommandons d'utiliser votre doigt, pas un maillet !) En appuyant sur le bouton Réinitialiser, le nombre de coups et d'échecs est réinitialisé à 0.

DÉPLACER LA TAUPE

Dans les programmes que vous avez écrits jusqu'à présent, vous avez appelé des procédures intégrées telles que Vibrer dans HelloPurr. Ne serait-il pas bien si App Inventor disposait d'une procédure permettant de déplacer un ImageSprite vers un emplacement aléatoire sur l'écran ? La mauvaise nouvelle : ce n'est pas le cas. La bonne nouvelle : vous pouvez créer vos propres procédures ! Tout comme les procédures intégrées, votre procédure apparaîtra dans un tiroir et vous pourrez l'utiliser n'importe où dans l'application.

Plus précisément, nous allons créer une procédure pour déplacer la taupe vers un emplacement aléatoire sur l'écran, que nous nommerons MoveMole. Nous souhaitons appeler MoveMole au début du jeu, lorsque l'utilisateur réussit à appuyer sur la taupe, et une fois par seconde.

CRÉATION DE LA PROCÉDURE MOVEMOLE

Pour comprendre comment déplacer la taupe, nous devons examiner le fonctionnement des graphiques Android.

Le canevas (et l'écran) peuvent être considérés comme une grille avec des coordonnées x (horizontales) et y (verticales), où les coordonnées (x, y) du coin supérieur gauche sont (0, 0).

La coordonnée x augmente à mesure que vous vous déplacez vers la droite et la coordonnée y augmente à mesure que vous descendez, comme le montre la figure 3-4. Les propriétés x et y d'un ImageSprite indiquent où est positionné son coin supérieur gauche ; ainsi, la taupe dans le coin supérieur gauche de la figure 3-4 a des valeurs x et y de 0.

Pour déterminer les valeurs x et y maximales disponibles afin que la taupe apparaisse à l'écran, nous devons utiliser les propriétés Largeur et Hauteur de Mole et Canvas1. (Les propriétés Largeur et Hauteur de la taupe sont les mêmes que la taille de l'image que vous avez téléchargée. Lorsque vous avez créé Canvas1, vous définissez sa hauteur sur 300 pixels et sa largeur sur « Remplir le parent », qui copie la largeur de son élément parent, qui dans ce cas, c'est l'écran.) Si la taupe a une largeur de 36 pixels et que la toile a une largeur de 200 pixels, la coordonnée x du côté gauche de la taupe peut être aussi basse que 0 (tout à gauche) ou aussi haute comme 164 (200 – 36, ou Canvas1.Width – Mole.Width) sans que la taupe ne s'étende sur le bord droit de l'écran. De même, la coordonnée y du haut de la taupe peut aller de 0 à Canvas1.Height – Mole.Height.

La figure 3-5 montre la procédure que vous allez créer, annotée avec une description commentaires (que vous pouvez éventuellement ajouter à votre procédure).

Pour placer la taupe au hasard, nous voudrons sélectionner une coordonnée x comprise entre 0 et Canvas1.Width – Mole.Width. De même, nous voudrons que la coordonnée y soit comprise entre 0 et Canvas1.Height – Mole.Height. Nous pouvons générer un nombre aléatoire grâce à la procédure intégrée Random Integer, que vous pouvez trouver dans le tiroir Math. Vous devrez modifier le paramètre par défaut « from » de 1 à 0 et remplacer les paramètres « to », comme indiqué dans la figure 3-5.

50 Chapitre 3 : MoleMash

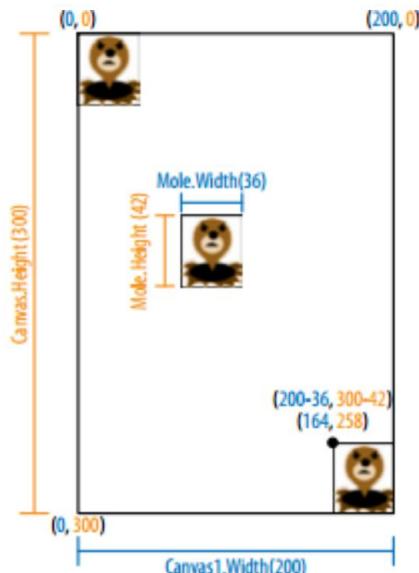


Figure 3-4. Positions de la taupe sur l'écran, avec informations sur les coordonnées, la hauteur et la largeur ; Les coordonnées x et les largeurs sont affichées en bleu, tandis que les coordonnées y et les hauteurs sont en orange.

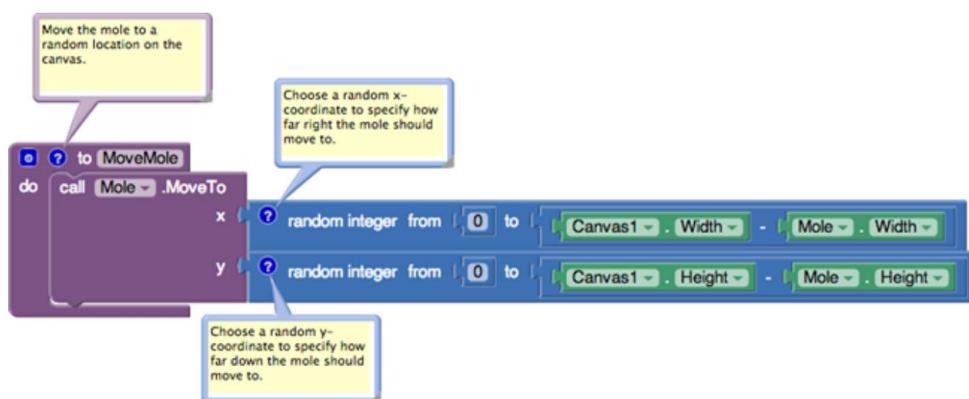


Figure 3-5. La procédure MoveMole, qui place la taupe à un endroit aléatoire

Pour créer la procédure :

1. Dans l'éditeur de blocs, cliquez sur le tiroir Procédure.
2. Faites glisser le bloc de procédure vers (contenant « faire » et non « résultat »).
3. Sur le nouveau bloc, cliquez sur le texte « procédure » et tapez « MoveMole » pour définir le nom de la procédure.

4. Parce que nous voulons déplacer la taupe, cliquez sur le tiroir Mole et faites glisser l'appel Mole.MoveTo dans la procédure, à droite de « faire ». Notez les sockets ouvertes sur le côté droit qui indiquent que nous devons fournir les coordonnées x et y.
5. Pour spécifier que la nouvelle coordonnée x de la taupe doit être comprise entre 0 et Canvas1.Width – Mole.Width, comme indiqué précédemment, procédez comme suit :
 - Depuis le tiroir Math, faites glisser l' entier aléatoire du bloc, en plaçant la fiche (saillie) sur son côté gauche dans la prise « x » lors de l'appel.
Taupe.MoveTo.
 - Changez le bloc numéro « 1 » sur la prise « from » en cliquant dessus puis en saisissant le chiffre « 0 ».
 - Supprimez le nombre « 100 » en cliquant dessus et en appuyant sur la touche Suppr ou Suppr de votre clavier, ou en le faisant glisser vers la corbeille.
 - Depuis le tiroir Math, faites glisser un bloc de soustraction (–) et placez-le dans la prise « vers ».
 - Dans le tiroir Canvas1, sélectionnez le bloc Canvas1.Width et faites-le glisser vers le côté gauche des opérations de soustraction.
 - De même, cliquez sur le tiroir Mole et faites glisser Mole.Width dans l'espace de travail. Ensuite, branchez-le sur le côté droit du bloc de soustraction.
6. Suivez une procédure similaire pour spécifier que la coordonnée y doit être aléatoire. entier compris entre 0 et Canvas1.Height – Mole.Height.
7. Vérifiez vos résultats par rapport à la figure 3-5.

APPELER MOVEMOLE AU DÉMARRAGE DE L'APPLICATION

Maintenant que vous avez écrit la procédure MoveMole , utilisons-la. Parce qu'il est si courant que les programmeurs souhaitent que quelque chose se produise au démarrage d'une application, il existe un bloc spécialement prévu à cet effet : Screen1.Initialize.

1. Cliquez sur le tiroir Screen1 et faites glisser Screen1.Initialize.
2. Cliquez sur le tiroir Procédures, dans lequel vous verrez un bloc d'appel MoveMole . (C'est plutôt cool que vous ayez créé un nouveau bloc, n'est-ce pas ?!) Faites-le glisser et placez-le dans Screen1.Initialize, comme le montre la figure 3-6.

52 Chapitre 3 : MoleMash



Figure 3-6. Appel de la procédure MoveMole au démarrage de l'application

APPEL DE MOVEMOLE CHAQUE SECONDE

Faire bouger la taupe toutes les secondes nécessitera le composant Clock . Nous avons laissé la propriété TimerInterval pour Clock1 à sa valeur par défaut de 1 000 (millisecondes) ou 1 seconde. Cela signifie que chaque seconde, quelle que soit la valeur spécifiée dans un bloc Clock1.Timer , aura lieu. Voici comment configurer cela :

1. Cliquez sur le tiroir Clock1 et faites glisser Clock1.Timer.
2. Cliquez sur le tiroir Procédures et faites glisser un bloc d'appel MoveMole dans le bloc Clock1.Timer , comme illustré dans la figure 3-7.



Figure 3-7. Appel de la procédure MoveMole lorsque le timer expire (chaque seconde)

Si c'est trop rapide ou trop lent pour vous, vous pouvez modifier la propriété TimerInterval pour Clock1 dans le Component Designer pour le faire bouger plus ou moins fréquemment.

GARDER LE SCORE

Comme vous vous en souvenez peut-être, vous avez créé deux étiquettes, HitsCountsLabel et MissesCountsLabel , qui avait des valeurs initiales de 0. Nous aimerais incrémenter les nombres dans ces étiquettes chaque fois que l'utilisateur réussit à appuyer sur la taupe (un coup) ou à appuyer sur l'écran sans toucher la taupe (un échec). Pour ce faire, nous utiliserons le bloc Canvas1.Touched , qui indique que le canevas a été touché, les coordonnées x et y de l'endroit où il a été contacté (que nous n'avons pas besoin de connaître) et si un sprite a été touché (ce qui nous avons besoin de savoir). La figure 3-8 montre le code que vous allez créer.

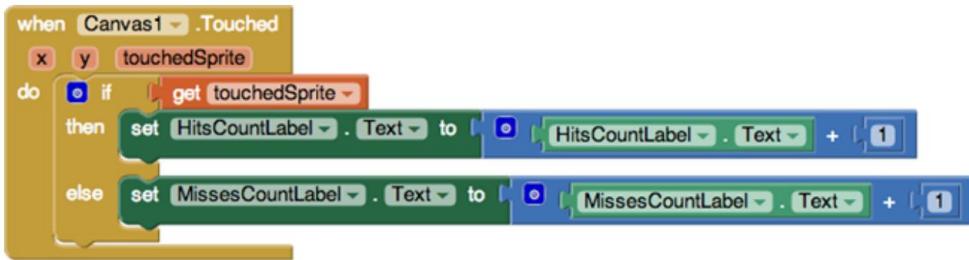


Figure 3-8. Incrémentation du nombre de hits (HitsCountLabel) ou d'échecs (MissesCountLabel) lorsque Canvas1 est touché

Vous pouvez traduire les blocs de la figure 3-8 de la manière suivante : chaque fois que vous appuyez sur le canevas, vérifiez si un sprite a été tapé. Parce qu'il n'y a qu'un seul sprite dans notre programme, ce doit être Mole1. Si Mole1 est sélectionné, ajoutez un au nombre dans HitsCountLabel.Text ; sinon, ajoutez-en un à MissesCountLabel.Text. (La valeur de touchedSprite est fausse si aucun sprite n'a été touché.)

Voici comment créer les blocs :

1. Cliquez sur le tiroir Canvas1 et faites glisser Canvas1.Touched.
2. Cliquez sur le tiroir Contrôle et faites glisser le bloc if-then . Cliquez sur son icône bleue et ajoutez une branche else . Ensuite, placez-le dans Canvas1.Touched.
3. Passez la souris sur le paramètre d'événement touchedSprite sur Canvas1.Touched, puis faites glisser le bloc get touchedSprite et placez-le dans le socket de test de if-then-autre.
4. Parce que nous voulons que HitsCountLabel.Text soit incrémenté si le test réussit (si le grain de beauté a été touché), procédez comme suit :
 - Depuis le tiroir HitsCountLabel, faites glisser l' ensemble HitsCountLabel.Text bloquer , en le plaçant à droite de « alors ».
 - Cliquez sur le tiroir Math et faites glisser un signe plus (+), en le plaçant dans le champ « à » prise.
 - Cliquez sur le tiroir HitsCountLabel et faites glisser le bloc HitsCountLabel.Text à gauche du signe plus.
 - Cliquez sur le tiroir Math et faites glisser un bloc 0 à droite du signe plus.
 Cliquez sur 0 et remplacez-le par 1.
5. Répétez l'étape 4 pour MissesCountLabel dans la section else du bloc ifelse .

54 Chapitre 3 : MoleMash



Testez votre application Vous pouvez tester ce nouveau code sur votre appareil en appuyant sur la toile, à la fois sur et depuis la taupe, et en regardant le score changer.

RÉSUMÉ PROCÉDURAL

La possibilité de nommer et d'appeler ultérieurement un ensemble d'instructions comme MoveMole est l'un des outils clés de l'informatique et est appelée abstraction procédurale. On l'appelle « abstraction » parce que l'appelant de la procédure (qui, dans les projets du monde réel, est susceptible d'être différent de l'auteur de la procédure) a seulement besoin de savoir ce que fait la procédure (déplace la taupe), et non comment elle le fait (en effectuant deux appels au générateur de nombres aléatoires). Sans abstraction procédurale, les grands programmes informatiques ne seraient pas possibles, car ils contiennent trop de code pour que les individus puissent les garder en tête à la fois. Ceci est analogue à la division du travail dans le monde réel, où, par exemple, différents ingénieurs conçoivent différentes parties d'une voiture, aucun d'eux ne comprenant tous les détails, et le conducteur n'a qu'à comprendre l'interface (par exemple, en appuyant sur le bouton). pédale de frein pour arrêter la voiture), pas la mise en œuvre.

Certains avantages de l'abstraction procédurale par rapport au copier-coller de code sont :

- Il est plus facile de tester le code s'il est soigneusement séparé du reste du programme.
- S'il y a une erreur dans le code, il suffit de la corriger à un seul endroit.
- Pour modifier la mise en œuvre, par exemple en s'assurant que la taupe ne bouge pas quelque part où il est apparu récemment, il vous suffit de modifier le code à un seul endroit.

- Les procédures peuvent être rassemblées dans une bibliothèque et utilisées dans différents programmes.
(Malheureusement, cette fonctionnalité n'est actuellement pas prise en charge dans App Inventor.)
- Diviser le code en morceaux vous aide à réfléchir et à mettre en œuvre l'application (« diviser pour régner »).

- Choisir de bons noms pour les procédures permet de documenter le code, ce qui le rend plus facile à lire pour quelqu'un d'autre (ou pour vous, un mois plus tard).

Dans les chapitres suivants, vous apprendrez comment rendre les procédures encore plus puissantes : en ajoutant des arguments, en fournissant des valeurs de retour et en faisant en sorte que les procédures s'appellent elles-mêmes. Pour un aperçu, voir le chapitre 21.

RÉINITIALISER LE SCORE

Un ami qui vous voit jouer à MoleMash voudra probablement l'essayer aussi, il est donc bon d'avoir un moyen de réinitialiser le nombre de succès et d'échecs à 0. En fonction des didacticiels que vous avez déjà suivis, vous pourriez être Je ne peux pas comprendre comment procéder sans lire les instructions suivantes.

Pensez à l'essayer avant de lire à l'avance.

Ce dont nous avons besoin, c'est d'un bloc `ResetButton.Click` qui définit les valeurs de `HitsCountLabel.Text` et `MissesCountLabel.Text` à 0. Créez les blocs affichés dans Figure 3-9.

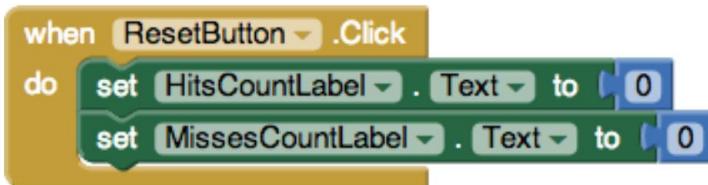


Figure 3-9. Réinitialisation du nombre de succès (`HitsCountLabel`) et d'échecs (`MissesCountLabel`) lorsque le bouton Réinitialiser est enfoncé

À ce stade, vous n'avez probablement pas besoin d'instructions étape par étape pour créer un gestionnaire d'événements de clic de bouton avec des étiquettes de texte, mais voici une astuce pour accélérer le processus : au lieu d'obtenir votre numéro dans le tiroir Math, tapez simplement 0, et le

56 Chapitre 3 : MoleMash

le bloc devrait être créé pour vous. (Ces types de raccourcis clavier existent également pour d'autres blocs.)



Testez votre application Essayez de frapper et de manquer la taupe, puis d'appuyer sur le bouton Réinitialiser.

VIBRANT LORSQUE LE TAÏE EST TOUCHÉ

Nous avons dit plus tôt que nous souhaitions que l'appareil vibre lorsque l'utilisateur tape sur la taupe, ce que nous pouvons faire avec le bloc Sound1.Vibrate , comme le montre la figure 3-10.



Figure 3-10. Faire vibrer l'appareil brièvement (pendant 100 millisecondes) lorsque le grain de beauté est touché



Testez votre application Découvrez comment fonctionne la vibration lorsque vous appuyez réellement sur le grain de beauté. Si la vibration est trop longue ou trop courte à votre goût, modifiez le nombre de millisecondes dans le bloc Sound1.Vibrate.

L'application complète : MoleMash

La figure 3-11 illustre les blocs de l'application MoleMash complète.

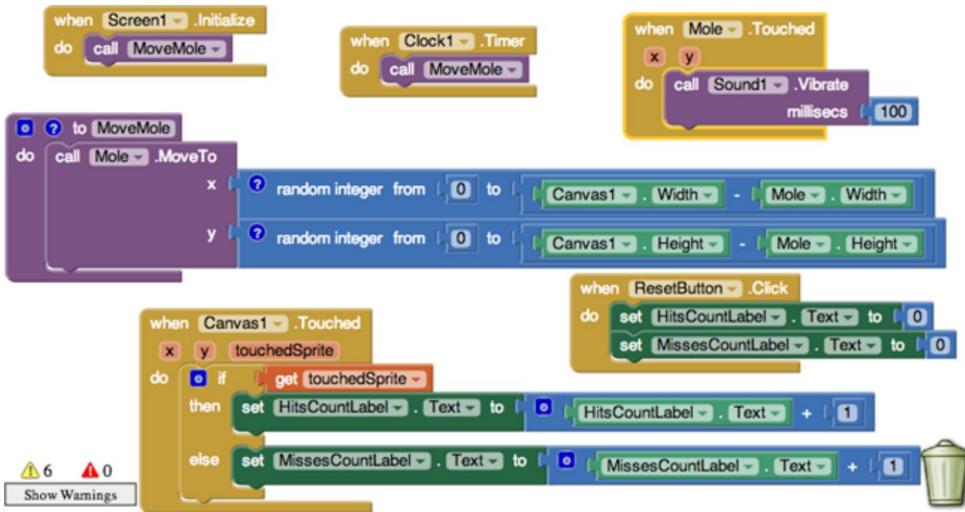


Figure 3-11. L'application MoleMash complète

Variantes

Voici quelques idées d'ajouts à MoleMash :

- Ajoutez des boutons pour permettre à l'utilisateur d'accélérer ou de ralentir la taupe.
- Ajoutez une étiquette pour suivre et afficher le nombre de fois où la taupe a été appuée (déplacé).
- Ajoutez un deuxième ImageSprite avec une image de quelque chose que l'utilisateur ne devrait pas frapper, comme une fleur. Si l'utilisateur le touche, pénalisez-le en réduisant son score ou en mettant fin à la partie.
- Au lieu d'utiliser l'image d'un grain de beauté, laissez l'utilisateur sélectionner une image avec le Composant ImagePicker .

Résumé

Dans ce chapitre, nous avons abordé un certain nombre de techniques utiles pour les applications en général et les jeux en particulier :

- Le composant Canvas utilise un système de coordonnées xy, où x représente la direction horizontale (de 0 à gauche à Canvas.Width-1 à droite) et y la direction verticale (de 0 en haut à Canvas.Height-1 en bas). La hauteur et la largeur d'un ImageSprite peuvent être soustraites de la hauteur et de la largeur d'un Canvas pour garantir que le sprite s'adapte entièrement au Canvas.

58 Chapitre 3 : MoleMash

- Vous pouvez profiter de l'écran tactile de l'appareil via le Canvas et Méthodes Touched des composants ImageSprite .
- Vous pouvez créer des applications en temps réel qui réagissent non seulement aux entrées de l'utilisateur mais également en réponse au minuteur interne de l'appareil. Plus précisément, le bloc Clock.Timer s'exécute à la fréquence spécifiée dans la propriété Clock.Interval et peut être utilisé pour déplacer des composants ImageSprite (ou autres).
- Vous pouvez utiliser des étiquettes pour afficher les scores, qui augmentent (ou diminuent) en réponse à la les actions du joueur.
- Vous pouvez fournir un retour tactile aux utilisateurs via la méthode Sound.Vibrate , qui fait vibrer l'appareil pendant le nombre de millisecondes spécifié.
- Au lieu d'utiliser simplement les méthodes intégrées, vous pouvez créer des procédures pour nommer un ensemble de blocs qui peuvent être appelés de la même manière que ceux intégrés. C'est ce qu'on appelle l'abstraction procédurale et constitue un concept clé en informatique, permettant la réutilisation du code et rendant possibles des applications complexes.
- Vous pouvez générer un comportement imprévisible avec le bloc entier aléatoire (dans le Tiroir mathématique), rendant un jeu différent à chaque fois qu'il est joué.

Vous apprendrez davantage de techniques de jeu, notamment la détection de collisions entre déplacer les composants ImageSprite , au chapitre 5.

Pas d'envoi de SMS en conduisant



Ce chapitre vous guide à travers la création de No Texting While Driving, une application de « répondeur texte » qui répond automatiquement aux messages texte que vous recevez pendant que vous conduisez (ou au bureau, etc.), prononce les messages texte à haute voix et envoie même des informations de localisation dans le cadre du réponse textuelle automatisée. L'application montre comment vous pouvez contrôler

certaines des fonctionnalités intéressantes d'un téléphone Android, notamment l'envoi de SMS, la synthèse vocale, les données persistantes et la détection de localisation GPS.

En janvier 2010, le Conseil national de sécurité (NSC) des États-Unis a annoncé les résultats d'une étude selon laquelle au moins 28 pour cent de tous les accidents de la route, soit près de 1,6 million d'accidents chaque année, sont causés par des conducteurs utilisant des téléphones portables, et au moins 200 000 de ces accidents se sont produits alors que les conducteurs envoyait des SMS.¹ En conséquence, de nombreux États ont complètement interdit aux conducteurs d'utiliser leur téléphone portable.

¹ <http://bit.ly/1qiH7aZ>

Daniel Finnegan, étudiant à l'Université de San Francisco qui suit un cours de programmation App Inventor, a eu l'idée d'une application pour lutter contre l'épidémie de conduite automobile et d'envoi de SMS. L'application qu'il a créée, illustrée à la figure 4-1, répond automatiquement (et en mains libres) à tout texte avec un message tel que « Je conduis en ce moment, je vous contacterai sous peu ».

L'application a ensuite été étendue pour pouvoir prononcer les textes entrants à haute voix et ajoutez la position GPS du conducteur au texte de réponse automatique, et cela a été transformé en didacticiel pour l'application Site de l'inventeur.

Quelques semaines après la publication de l'application sur le site App Inventor, State Farm Insurance a créé une application Android appelée On the Move, qui avait des fonctionnalités similaires à No Texting While Driving.

Nous ne savons pas si l'application de Daniel ou le didacticiel du site App Inventor a influencé On the Move, mais il est intéressant d'envisager la possibilité qu'une application créée dans un cours de programmation débutant (par un étudiant en écriture créative, rien de moins !) puisse avoir inspiré ce logiciel produit en série, ou du moins a contribué à l'écosystème qui l'a donné naissance. Cela démontre certainement comment App Inventor a abaissé la barrière à l'entrée afin que toute personne ayant une bonne idée puisse transformer rapidement et à moindre coût son idée en une application tangible et interactive. Clive Thompson du magazine Wired a repris la nouveauté et a écrit ceci :

Après tout, les logiciels affectent presque tout ce que nous faisons. Choisissez n'importe quel problème majeur – le réchauffement climatique, les soins de santé ou, dans le cas de Finnegan, la sécurité routière – et un logiciel intelligent fait partie de la solution. Pourtant, seule une infime partie des gens envisagent d'apprendre à écrire du code, ce qui signifie que nous n'exploitons pas la créativité d'une grande partie de la société.² App Inventor vise à exploiter la créativité mentionnée par Thompson, à ouvrir le monde de la création de logiciels à tout le monde.

Ce que vous apprenez

Il s'agit d'une application plus complexe que celles des chapitres précédents, vous allez donc la créer une fonctionnalité à la fois, en commençant par le message de réponse automatique. Vous en apprendrez davantage sur :



Figure 4-1. L'application Pas d'envoi de SMS en conduisant

² Clive Thompson, « Clive Thompson sur le codage pour les masses », <http://wrd.cm/1uT25O5>

- Le composant Texting pour l'envoi de textes et le traitement des textes reçus.
- Un formulaire de saisie pour soumettre le message de réponse personnalisé.
- Le composant de base de données TinyDB pour enregistrer le message personnalisé même après la fermeture de l'application.
- L' événement Screen.Initialize pour charger la réponse personnalisée lorsque l'application lance.
- Le composant TextToSpeech pour prononcer des textes à haute voix.
- Le composant LocationSensor pour signaler l'emplacement actuel du conducteur.

Commencer

Ouvrez votre navigateur sur le site Web App Inventor et démarrez un nouveau projet. Nommez-le « NoTextingWhileDriving » (rappelez-vous que les noms de projets ne peuvent pas contenir d'espaces) et définissez le titre de l'écran sur « No Texting While Driving ». Ensuite, cliquez sur Connecter et configuez les tests en direct sur votre appareil ou sur l'émulateur.

Conception des composants

L'interface utilisateur de l'application est relativement simple : elle comporte une étiquette indiquant à l'utilisateur comment fonctionne l'application, une étiquette qui affiche le texte qui doit être automatiquement envoyé en réponse aux SMS entrants, une zone de texte pour modifier la réponse et un bouton pour soumettre la modification. Vous devrez également faire glisser un composant Texting , un composant TinyDB , un composant TextToSpeech et un composant LocationSensor , qui apparaîtront tous dans la zone « Composants non visibles ». Vous pouvez voir à quoi cela devrait ressembler dans l'instantané du Concepteur de composants dans la figure 4-2.

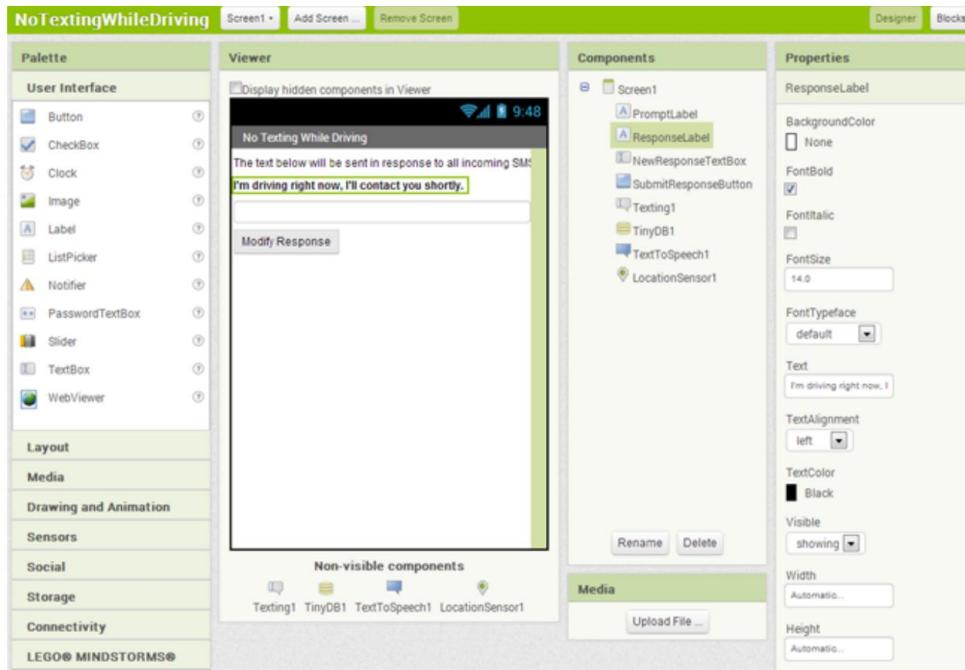


Figure 4-2. L'application Pas d'envoi de SMS pendant la conduite dans le Concepteur de composants

Vous pouvez créer l'interface utilisateur illustrée à la figure 4-2 en faisant glisser le composants répertoriés dans le tableau 4-1.

Tableau 4-1. Tous les composants de l'application No Texting

Type de composant	Groupe de palettes	Comment vous l'appellerez	But
Étiquette	Étiquette d'invite de l'interface utilisateur		Faites savoir à l'utilisateur comment fonctionne l'application.
Étiquette	Étiquette de réponse de l'interface utilisateur		La réponse qui sera renvoyée au expéditeur.
Zone de texte	Interface utilisateur newResponseTextBox	L'utilisateur entrera ici la réponse personnalisée.	
Bouton	Interface utilisateur SubmitResponseButton	L'utilisateur clique dessus pour soumettre la réponse.	
Envoyer des SMS	Sociale	Envoyer des SMS1	Traitez les textes.
MinusculeDB	Stockage	MinusculeDB1	Stockez la réponse dans la base de données.
Médias TextToSpeech		TexteVersSpeech1	Prononcez le texte à voix haute.
Capteurs LocationSensor		LocalisationCapteur1	Déetectez où se trouve l'appareil.

Définissez les propriétés des composants de la manière suivante :

- Définissez le texte de PromptLabel sur « Le texte ci-dessous sera envoyé en réponse à tous les SMS reçus pendant l'exécution de cette application.
- Définissez le texte de ResponseLabel sur : "Je conduis en ce moment, je vous contacterais". prochainement." Vérifiez sa propriété d'audace .
- Définissez le texte de NewResponseTextbox sur « ». (Cela laisse la zone de texte vide pour le entrée de l'utilisateur.)
- Définissez l' indice de NewResponseTextbox sur « Entrer le nouveau texte de réponse ».
- Définissez le texte de SubmitResponseButton sur « Modifier la réponse ».

Ajout de comportements aux composants

Vous commencerez par programmer le comportement de réponse automatique dans lequel une réponse textuelle est envoyée à tout texte entrant. Vous ajouterez ensuite des blocs afin que l'utilisateur puisse spécifier une réponse personnalisée et enregistrer cette réponse de manière persistante. Enfin, vous ajouterez des blocs qui lieront à haute voix les textes entrants et ajouterez des informations de localisation aux textes de réponse automatique.

RÉPONSE AUTOMATIQUE À UN TEXTE

Pour le comportement de réponse automatique, vous utiliserez le composant Texting d'App Inventor . Vous pouvez considérer ce composant comme une petite personne à l'intérieur de votre téléphone qui sait lire et écrire des textes. Pour la lecture de textes, le composant fournit un bloc d'événement Texting.MessageReceived . Vous pouvez faire glisser ce bloc vers l'extérieur et y placer des blocs pour montrer ce qui doit se passer lorsqu'un texte est reçu. Dans le cas de cette application, nous souhaitons renvoyer automatiquement un texte en réponse.

Vous pouvez envoyer un texte avec trois blocs. Tout d'abord, vous définissez le numéro de téléphone auquel le texte doit être envoyé, qui est une propriété du composant Texting1 . Ensuite, vous définissez le message à envoyer, également une propriété de Texting1. Enfin, vous envoyez réellement le texte avec le bloc Texting1.SendMessage . Le tableau 4-2 répertorie tous les blocs dont vous aurez besoin pour ce comportement de réponse automatique, et la figure 4-3 montre à quoi ils devraient ressembler dans l'éditeur de blocs.

Tableau 4-2. Les blocs pour envoyer une réponse automatique

Type de bloc	Tiroir	But
SMS1.MessageReceived	Envoyer des SMS	Le gestionnaire d'événements qui est déclenché lorsque le téléphone reçoit un SMS.
définir Texting1.PhoneNumber sur SMS		Définissez la propriété PhoneNumber avant l'envoi.
numéro de valeur	Faire glisser depuis le bloc	Le numéro de téléphone de la personne qui a envoyé le texte.

Type de bloc	Tiroir	But
définissez Texting1.Message sur	Envoyer des SMS	Définissez la propriété Message avant l'envoi.
ResponseLabel.Text	Étiquette de réponse	Le message que l'utilisateur a saisi.
SMS1.SendMessage	Envoyer des SMS	Envoyez le message.

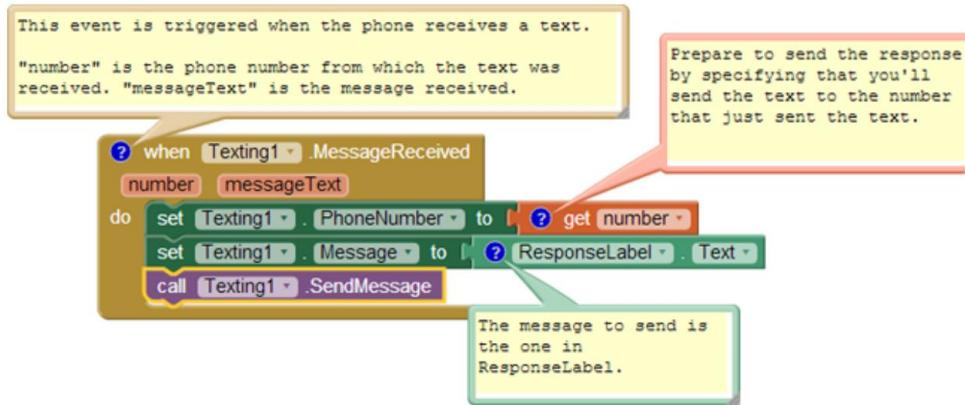


Figure 4-3. Répondre à un SMS entrant

Comment fonctionnent les blocs

Lorsque le téléphone reçoit un message texte, l'événement Texting1.MessageReceived est déclenché. Le numéro de téléphone de l'expéditeur est dans le numéro d'argument, et le message reçu est dans l'argument messageText.

Comme le texte de réponse automatique doit être renvoyé au expéditeur, SMS.PhoneNumber est défini sur un numéro. Texting.Message est défini à ResponseLabel.Text, qui correspond à ce que vous avez tapé dans le Designer : « Je conduis maintenant, je vous contacterai sous peu. Lorsque ceux-ci sont définis, l'application appelle Texting.SendMessage pour envoyer réellement la réponse.



Testez votre application Vous aurez besoin de deux téléphones pour tester ce comportement, un pour exécuter l'application et un pour envoyer le texte initial. Si tu ne le fais pas avec un deuxième téléphone à portée de main, vous pouvez utiliser Google Voice ou un service similaire sur votre ordinateur et envoyez des SMS à partir de celui-ci service au téléphone exécutant l'application. Après avoir configuré les choses, envoyer un SMS au téléphone exécutant l'application. Est-ce que le premier téléphone recevoir le texte de réponse ?

SAISIR UNE RÉPONSE PERSONNALISÉE

Ensuite, ajoutons des blocs pour que l'utilisateur puisse saisir sa propre réponse personnalisée. Dans le Concepteur de composants, vous avez ajouté un composant TextBox nommé NewResponseTextbox ; c'est ici que l'utilisateur saisira la réponse personnalisée. Lorsque l'utilisateur clique sur le SubmitResponseButton, vous devez copier l'entrée (NewResponseTextbox) dans le ResponseLabel, qui est utilisé pour répondre aux textes. Le tableau 4-3 répertorie les blocs dont vous aurez besoin pour transférer une réponse nouvellement saisie dans ResponseLabel.

Tableau 4-3. Blocs d'affichage de la réponse personnalisée

Type de bloc	Tiroir	But
SubmitResponseButton.Click	Bouton SoumettreRéponse	L'utilisateur clique sur ce bouton pour soumettre un nouveau message de réponse.
définir ResponseLabel.Text sur	ResponseLabel Déplace (définit)	la nouvelle valeur entrée vers cette étiquette.
NewResponseTextbox.Text	NewResponseTextbox	L'utilisateur a saisi la nouvelle réponse ici.
définir NewResponseTextbox.Text sur NewResponseTextbox		Videz la zone de texte après le transfert information
texte ("")	Texte	Le texte vide.

Comment fonctionnent les blocs

Pensez à la façon dont vous interagissez avec un formulaire de saisie typique : vous tapez d'abord quelque chose dans un texte, puis cliquez sur un bouton de soumission pour signaler au système de le traiter. Le formulaire de saisie pour cette application, ce n'est pas différent. La figure 4-4 montre comment les blocs sont programmés pour que lorsque l'utilisateur clique sur le SubmitResponseButton, le SubmitResponseButton.Click

l'événement est déclenché.

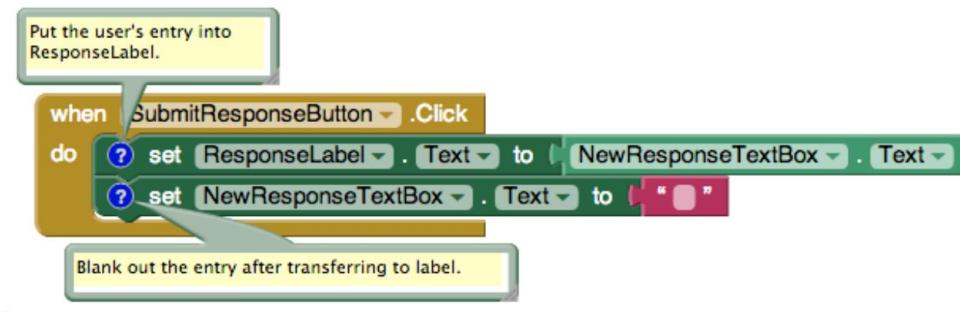


Figure 4-4. Définition de la réponse à l'entrée de l'utilisateur

Dans ce cas, le gestionnaire d'événements copie (ou, en termes de programmation, définit) ce que l'utilisateur a entré dans NewResponseTextbox dans ResponseLabel. Rappelez-vous que ResponseLabel contient le message qui sera envoyé dans la réponse automatique, vous voulez donc être sûr d'y placer le message personnalisé nouvellement saisi.



Testez votre application Saisissez une réponse personnalisée et soumettez-la, puis utilisez le deuxième téléphone pour envoyer un autre SMS au téléphone exécutant l'application. La réponse personnalisée a-t-elle été envoyée ?

STOCKAGE PERMANENT DE LA RÉPONSE PERSONNALISÉE

Votre utilisateur peut désormais personnaliser la réponse automatique, mais il y a un problème : si l'utilisateur saisit une réponse personnalisée, puis ferme l'application et la relance, la réponse personnalisée n'apparaîtra pas (à la place, la réponse par défaut le fera). Ce comportement n'est pas celui auquel vos utilisateurs s'attendent ; ils voudront voir la réponse personnalisée qu'ils ont saisie lorsqu'ils redémarreront l'application. Pour que cela se produise, vous devez stocker cette réponse personnalisée de manière persistante.

Placer des données dans la propriété ResponseLabel.Text revient techniquement à les stocker, mais le problème est que les données stockées dans les propriétés du composant sont des données transitoires. Les données transitoires sont comme votre mémoire à court terme ; le téléphone « l'oublie » dès qu'une application se ferme. Si vous souhaitez que votre application se souvienne de quelque chose de manière persistante, vous devez le transférer de la mémoire à court terme (une propriété ou une variable d'un composant) vers la mémoire à long terme (une base de données ou un fichier).

Pour stocker des données de manière persistante dans App Inventor, vous utilisez le composant TinyDB , qui stocke les données dans un fichier sur l'appareil Android. TinyDB fournit deux fonctions : StoreValue et GetValue. Avec le premier, l'application peut stocker des informations dans la base de données de l'appareil, tandis qu'avec le second, l'application peut récupérer des informations déjà stockées.

Pour de nombreuses applications, vous utiliserez le schéma suivant :

1. Stockez les données dans la base de données chaque fois que l'utilisateur soumet une nouvelle valeur.

2. Lorsque l'application démarre, chargez les données de la base de données dans une variable ou propriété.

Vous commencerez par modifier le gestionnaire d'événements SubmitResponseButton.Click afin qu'il stocke les données de manière persistante, en utilisant les blocs répertoriés dans le tableau 4-4.

Tableau 4-4. Blocs pour stocker la réponse personnalisée avec TinyDB

Type de bloc	Tiroir	But
TinyDB1.StoreValue		Stockez le message personnalisé dans la base de données du téléphone.

Type de bloc	Tiroir	But
text («responseMessage») Text		Utilisez-le comme balise pour les données.
ResponseLabel.Text	ResponseLabel	Le message de réponse est maintenant ici.

Comment fonctionnent les blocs

Cette application utilise TinyDB pour prendre le texte qu'elle vient de mettre dans ResponseLabel et le stocker dans la base de données. Comme le montre la figure 4-5, lorsque vous stockez quelque chose dans la base de données, vous fournissez une balise avec celui-ci ; dans ce cas, la balise est « responseMessage ». Considérez la balise comme le nom des données dans la base de données ; il identifie de manière unique les données que vous stockez. Comme vous le verrez dans la section suivante, vous utiliserez la même balise («responseMessage») lorsque vous rechargerez les données depuis la base de données.



Figure 4-5. Stockage persistant de la réponse personnalisée

RÉCUPÉRATION DE LA RÉPONSE PERSONNALISÉE À L'OUVERTURE DE L'APPLICATION

La raison du stockage de la réponse personnalisée dans la base de données est qu'elle puisse être rechargée dans l'application la prochaine fois que l'utilisateur l'ouvrira. App Inventor fournit un bloc d'événement spécial qui est déclenché à l'ouverture de l'application : Screen1.Initialize (si vous avez terminé MoleMash au chapitre 3, vous l'avez déjà vu). Si vous faites glisser ce bloc d'événement et y placez des blocs, ces blocs seront exécutés immédiatement au lancement de l'application.

Pour cette application, votre gestionnaire d'événements Screen1.Initialize chargera la réponse personnalisée à partir de la base de données en utilisant la fonction TinyDB.GetValue. Les blocs dont vous aurez besoin pour cela sont indiqués dans le tableau 4-5.

Tableau 4-5. Blocs pour charger les données à l'ouverture de l'application

Type de bloc	Tiroir	But
Écran1.Initialiser	Écran1	Ceci est déclenché au démarrage de l'application.
TinyDB1.GetValue	MinusculeDB1	Obtenez le texte de réponse stocké dans la base de données.
texte (« réponseMessage »)	Texte	Branchez-le dans le socket de balise de TinyDB.GetValue, ce qui fait assurer-vous que le texte est le même que celui utilisé dans TinyDB.StoreValue plus tôt.
texte (« Je conduis en ce moment, je vais vous contacter sous peu »)	Texte	Branchez-le dans l'emplacement valueIfTagNotThere de TinyDB.GetValue. C'est le message par défaut qui devrait être utilisé si l'utilisateur n'a pas encore stocké de réponse personnalisée.
		définissez ResponseLabel.Text sur ResponseLabel Placez la valeur récupérée dans ResponseLabel.

Comment fonctionnent les blocs

La figure 4-6 montre les blocs. Pour les comprendre, il faut imaginer une ouverture utilisateur l'application pour la première fois, en saisissant une réponse personnalisée et en ouvrant l'application les fois suivantes. La première fois que l'utilisateur ouvre l'application, il n'y aura aucune personnalisation réponse dans la base de données à charger, vous souhaitez donc laisser la réponse par défaut dans le Étiquette de réponse. Aux lancements successifs, vous souhaitez charger les fichiers précédemment stockés réponse personnalisée de la base de données et placez-la dans ResponseLabel.



Figure 4-6. Chargement de la réponse personnalisée à partir de la base de données lors de l'initialisation de l'application

Lorsque l'application démarre, l'événement Screen1.Initialize est déclenché. L'application appelle le TinyDB1.GetValue avec une balise de réponseMessage, la même balise que vous avez utilisée lorsque vous a stocké l'entrée de réponse personnalisée de l'utilisateur plus tôt. S'il y a des données dans TinyDB avec une balise de ResponseMessage, il est renvoyé et placé dans ResponseLabel.

Cependant, il n'y aura pas de données au premier lancement de l'application ; ce sera le cas jusqu'à ce que l'utilisateur tape une réponse personnalisée. Pour gérer de tels cas, TinyDB1.GetValue a un deuxième paramètre, valueIfTagNotThere. Si aucune donnée n'est trouvée, la valeur dans valueIfTagNotThere est utilisée à la place. Dans ce cas, « Je conduis en ce moment, je vous contacte sous peu », la valeur par défaut, est placée dans ResponseLabel.



Testez votre application Pour tester ce comportement, vous devez redémarrer votre application pour voir si les données sont réellement stockées de manière persistante et récupérées correctement. Lors des tests en direct, vous pouvez redémarrer l'application en modifiant certaines propriétés de composants dans le concepteur, telles que la taille de police d'une étiquette. Cela entraînera le rechargement de l'application et le déclenchement de Screen.Initialize. Bien sûr, vous pouvez également tester l'application en la créant et en installant le fichier .apk sur votre téléphone. Une fois l'application sur votre téléphone, lancez-la, saisissez un message pour la réponse personnalisée, fermez l'application, puis rouvrez-la. Si le message que vous avez saisi est toujours là, tout fonctionne correctement.

PARLER À HAUTE VOIX DES TEXTES ENTRANTS

Dans cette section, vous allez modifier l'application de sorte que lorsque vous recevez un SMS, le numéro de téléphone de l'expéditeur, ainsi que le message, soient prononcés à haute voix. L'idée ici est que lorsque vous conduisez et entendez un texte arriver, vous pourriez être tenté de vérifier le texte même si vous savez que l'application envoie une réponse automatique. Avec la synthèse vocale, vous pouvez entendre les textes entrants et garder les mains sur le volant.

Les appareils Android offrent des fonctionnalités de synthèse vocale et App Inventor fournit un composant, TextToSpeech, qui prononcera tout texte que vous lui donnerez. Notez que le « Texte » dans TextToSpeech fait référence à une séquence de lettres, de chiffres et de ponctuation, et non à un texte SMS.

Le composant TextToSpeech est très simple à utiliser. Vous venez d'appeler sa fonction Speak et branchez le texte que vous souhaitez prononcer dans sa fente de message. Par exemple, les blocs illustrés à la figure 4-7 prononceraient les mots « Hello World ».

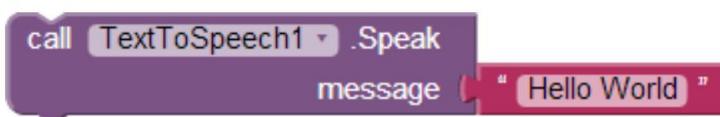


Figure 4-7. Blocages pour prononcer « Hello World » à haute voix

Pour l'application No Texting While Driving, vous devrez fournir un formulaire plus compliqué. message à prononcer, qui comprend à la fois le texte reçu et le numéro de téléphone de la personne qui l'a envoyé. Au lieu de connecter un objet texte statique tel que le bloc de texte « Hello World », vous connecterez un bloc de jointure . Une fonction couramment utilisée, join, vous permet de combiner des morceaux de texte séparés (ou des nombres et autres caractères) en un seul objet texte.

Vous devrez appeler TextToSpeech.Speak dans le Gestionnaire d'événements Texting.MessageReceived que vous avez programmé précédemment. Les blocs que vous avez programmés précédemment gèrent cet événement en définissant le numéro de téléphone et le message.

propriétés du composant Texting de manière appropriée, puis envoyer la réponse texte. Vous allez étendre ce gestionnaire d'événements en ajoutant les blocs répertoriés dans le tableau 4-6.

Tableau 4-6. Blocs pour prononcer à haute voix le texte entrant

Type de bloc	Tiroir	But
TextToSpeech1.Parler TextToSpeech1		Dites à haute voix le message reçu.
rejoindre	Texte	Concaténer (unir) les mots qui seront prononcés.
texte (« Texte SMS reçu de ») Texte		Les premiers mots prononcés.
obtenir un numéro	Faire glisser depuis quand bloquer Le numéro à partir duquel le texte original a été reçu.	
texte (« .Le message est »)	Texte	Mettez un point après le numéro de téléphone, puis dites : « Le message est.
obtenir le texte du message	Faites glisser depuis le bloc Le message d'origine reçu.	

Comment fonctionnent les blocs

Une fois la réponse envoyée, la fonction TextToSpeech1.Speak est appelée, comme indiqué en bas de la figure 4-8. Vous pouvez brancher n'importe quel texte dans la prise de message du Fonction TextToSpeech1.Speak . Dans ce cas, join est utilisé pour construire les mots à parlé - il concatène (ou joint) le texte « Texte SMS reçu de » et le numéro de téléphone à partir duquel le message a été reçu (obtenir le numéro), plus le texte « .Le message est », et enfin le message reçu (get messageText). Alors, si le texte « bonjour » a été envoyé à partir du numéro « 111-2222 », le téléphone disait : « SMS reçu de 111-2222. Le message est bonjour.

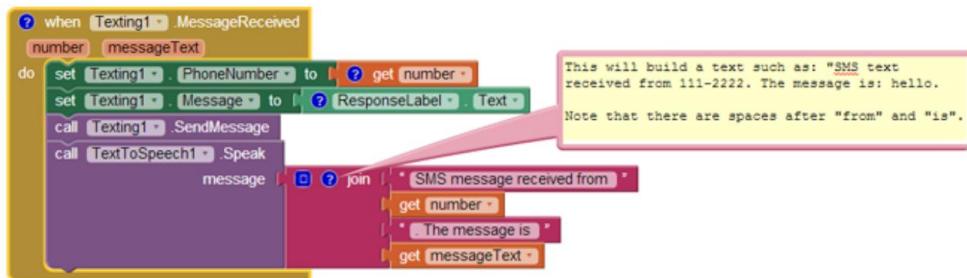


Figure 4-8. Prononcer à haute voix le texte entrant



Testez votre application Vous aurez besoin d'un deuxième téléphone pour tester votre application.

Depuis le deuxième téléphone, envoyez un SMS au téléphone exécutant le application. Le téléphone qui exécute l'application prononce-t-il le texte à haute voix ? Envoie-t-il toujours une réponse automatisée ?

AJOUT D'INFORMATIONS DE LOCALISATION À LA RÉPONSE

Les applications d'enregistrement aident les gens à se localiser mutuellement. Il y a une grande confidentialité préoccupations concernant de telles applications, l'une des raisons étant que le suivi de la localisation attise l'intérêt des gens. peur d'un appareil « Big Brother » qu'un gouvernement totalitaire pourrait mettre en place pour traquer où se trouvent ses citoyens. Mais les applications qui utilisent les informations de localisation peuvent être très utiles.

Pensez à un enfant perdu ou à des randonneurs qui ont parcouru un sentier dans les bois.

Dans l'application No Texting While Driving, vous pouvez utiliser le suivi de localisation pour transmettre un peu plus d'informations dans la réponse automatique aux SMS entrants. Au lieu de simplement « je conduis », le message de réponse peut être quelque chose comme : « Je conduis et je suis actuellement au 3413 Avenue des Cerise. Pour quelqu'un qui attend l'arrivée d'un ami ou d'un membre de la famille, ce des informations supplémentaires peuvent être utiles.

App Inventor fournit le composant LocationSensor pour l'interface avec le GPS du téléphone (ou Global Positioning System). Outre la latitude et la longitude Informations, le LocationSensor peut également exploiter Google Maps pour fournir au conducteur les informations adresse postale actuelle.

Il est important de noter que LocationSensor n'a pas toujours de lecture. Pour ça Pour cette raison, vous devez veiller à utiliser correctement le composant. Plus précisément, votre application doit répondre au gestionnaire d'événements LocationSensor.LocationChanged . UN L'événement LocationChanged se produit lorsque le capteur de localisation du téléphone obtient pour la première fois une lecture, et lorsque le téléphone est déplacé pour générer une nouvelle lecture. En utilisant les blocs répertoriés dans Tableau 4-7, notre schéma, illustré à la figure 4-9, répondra à l'événement LocationChanged en plaçant l'adresse actuelle dans une variable que nous nommerons lastKnownLocation. Plus tard, nous le ferons modifier le message de réponse pour incorporer l'adresse que nous obtenons de cette variable.

Tableau 4-7. Blocs pour configurer le capteur de localisation

Type de bloc	Tiroir	But
initialiser la variable globale (``dernier emplacement connu``)	Variables	Créer une variable pour contenir la dernière lecture adresse.
texte (« inconnu »)	Texte	Définissez la valeur par défaut au cas où le téléphone le capteur ne fonctionne pas.
LocationSensor1.LocationChanged	LocalisationCapteur1	Ceci est déclenché au premier emplacement lecture et chaque changement de lieu.
définir global lastKnownLocation sur	Faire glisser depuis l'initialisation bloc global.	Définissez cette variable pour qu'elle soit utilisée plus tard.
LocationSensor1.CurrentAddress	LocalisationCapteur1	Il s'agit d'une adresse postale telle que « 2222 Willard Street, Atlanta, Géorgie.

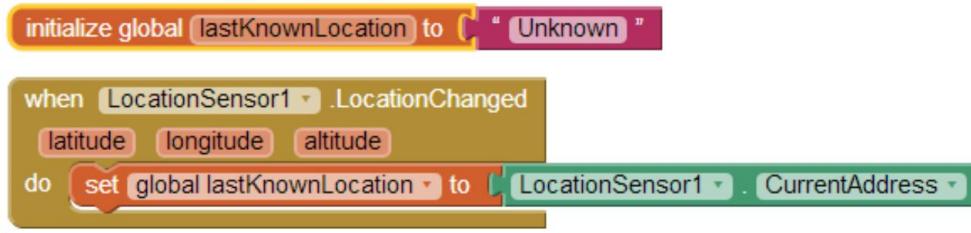


Figure 4-9. Enregistrer la position du téléphone dans une variable à chaque fois que la position GPS est senti

Comment fonctionnent les blocs

L'événement LocationSensor1.LocationChanged est déclenché la première fois que le capteur obtient une lecture de localisation, puis à chaque fois que l'appareil est déplacé pour qu'une nouvelle lecture est générée. La fonction LocationSensor1.CurrentAddress est appelée pour obtenir le adresse postale actuelle de l'appareil et stockez-la dans la variable lastKnownLocation.

Notez qu'avec ces blocs, vous n'avez terminé que la moitié du travail. L'application a encore besoin pour incorporer les informations de localisation dans le texte de réponse automatique qui sera envoyé retour à l'expéditeur. Vous le ferez ensuite.

ENVOI DE LA LOCALISATION DANS LE CADRE DE LA RÉPONSE

En utilisant la variable lastKnownLocation, vous pouvez modifier le Texting1.MessageReceived gestionnaire d'événements pour ajouter des informations de localisation à la réponse. Le tableau 4-8 répertorie les blocs vous en aurez besoin.

Tableau 4-8. Blocs pour afficher les informations de localisation dans la réponse automatique

Type de bloc	Tiroir	But
rejoindre	Texte	concaténer du texte ensemble
ResponseLabel.Text	MessageTextBox Il s'agit du message (personnalisé) dans la zone de texte.	
texte (« Mon dernier emplacement connu est : »)	Texte	Ceci sera prononcé après le message personnalisé (notez le espace de tête).
obtenir le capteur de localisation global lastKnownLocation		Il s'agit d'une adresse telle que « 1600 Pennsylvania Ave NW, Washington, DC 20500. »

Comment fonctionnent les blocs

Ce comportement fonctionne de concert avec l'événement LocationSensor1.LocationChanged et la variable lastKnownLocation. Comme vous pouvez le voir sur la figure 4-10, au lieu de directement en envoyant un message contenant le texte dans ResponseLabel.Text, l'application crée d'abord un

message en utilisant join. Il combine le texte de réponse dans ResponseLabel.Text avec le texte « Mon dernier emplacement connu est : » suivi de la variable lastKnownLocation.

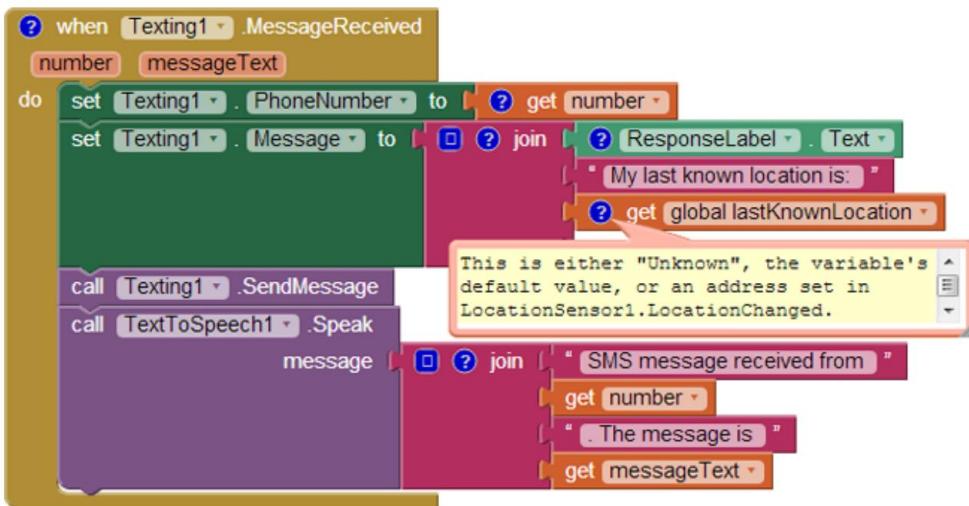


Figure 4-10. Inclure des informations de localisation dans le texte de la réponse

La valeur par défaut de lastKnownLocation est « inconnu », donc si le capteur de localisation n'a pas encore généré de lecture, la deuxième partie du message de réponse contiendra le texte « Ma dernière position connue est : inconnue. ». S'il y a eu une lecture, la deuxième partie de la réponse sera quelque chose comme : « Mon dernier emplacement connu est : 1600 Pennsylvania Ave NW, Washington, DC 20500. »



Testez votre application Depuis le deuxième téléphone, envoyez un SMS au téléphone exécutant l'application. Le deuxième téléphone reçoit-il le texte de réponse avec les informations de localisation ? Si ce n'est pas le cas, assurez-vous d'avoir activé le GPS dans les paramètres de localisation du téléphone exécutant l'application.

L'application complète : pas d'envoi de SMS en conduisant

La figure 4-11 montre la configuration du bloc final pour Pas d'envoi de SMS pendant la conduite.

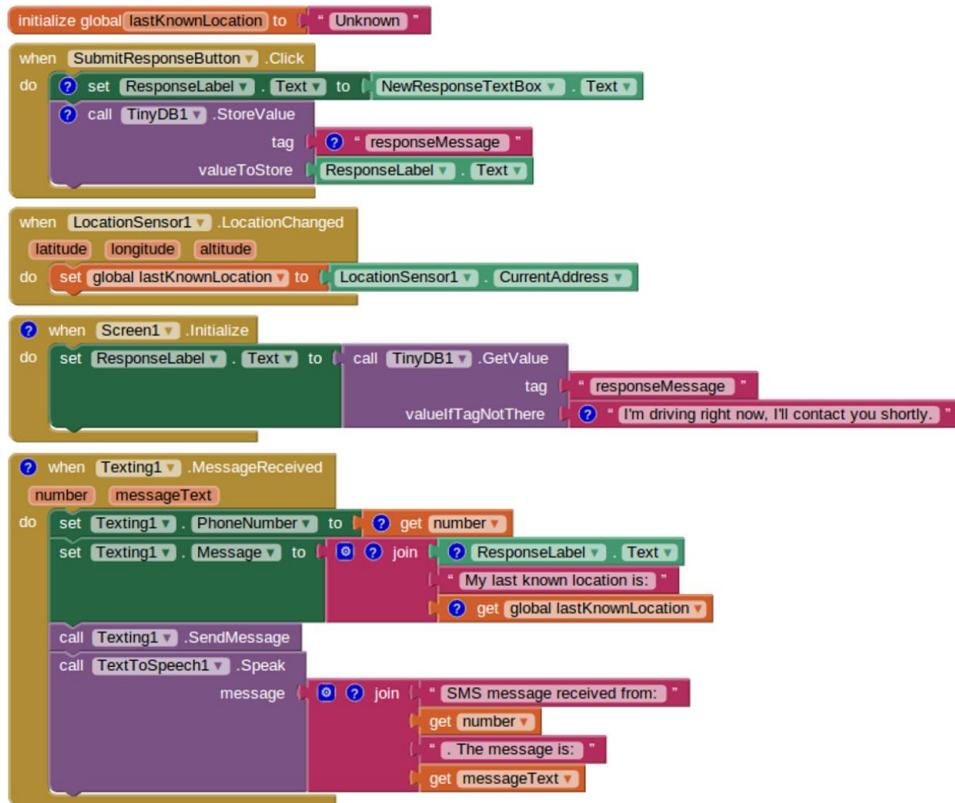


Figure 4-11. L'application complète No Texting While Driving

Variantes

Une fois que l'application fonctionne, vous souhaiterez peut-être explorer certaines variantes, telles que les suivantes :

- Écrivez une version qui permet à l'utilisateur de définir des réponses personnalisées pour des besoins particuliers. numéros de téléphone entrants. Vous devrez ajouter des blocs conditionnels (if) qui vérifient ces nombres. Pour plus d'informations sur les blocs conditionnels, voir le chapitre 18.
- Écrivez une version qui envoie des réponses personnalisées selon que l'utilisateur est ou non. dans certaines limites de latitude/longitude. Ainsi, si l'application détermine que vous êtes dans la chambre 222, elle renverra « Bob est dans la chambre 222 et ne peut pas envoyer de SMS pour le moment ». Pour plus d'informations sur le LocationSensor et la détermination des limites, voir le chapitre 23.
- Écrivez une version qui déclenche une alarme lorsqu'un texte est reçu d'un numéro dans une liste de « notifier ». Pour obtenir de l'aide sur l'utilisation des listes, consultez le chapitre 19.

Résumé

Voici quelques-uns des concepts que nous avons abordés dans ce didacticiel :

- Vous pouvez utiliser le composant Texting pour envoyer des messages texte et traiter ceux reçus.
Avant d'appeler `Texting.SendMessage`, vous devez définir les propriétés `PhoneNumber` et `Message` du composant Texting . Pour répondre à un texte entrant, programmez le gestionnaire `Texting.MessageReceived` .
- Le composant TinyDB est utilisé pour stocker des informations de manière persistante :
base de données du téléphone, afin que les données puissent être rechargées à chaque fois que l'application est ouverte.
Pour plus d'informations sur TinyDB, consultez le chapitre 22.
- Le composant TextToSpeech prend n'importe quel objet texte et le prononce à haute voix.
- Vous pouvez utiliser join pour rassembler (ou concaténer) des éléments de texte séparés en un seul objet texte.
- Le composant LocationSensor peut signaler la latitude, la longitude et l'adresse actuelle du téléphone.
Pour vous assurer qu'il dispose d'une lecture, vous devez accéder à ses données dans le gestionnaire d'événements `LocationSensor.LocationChanged` , qui est déclenché la première fois qu'une lecture est effectuée et à chaque modification ultérieure. Pour plus d'informations sur le LocationSensor, voir le chapitre 23.

Si vous souhaitez explorer davantage les applications de traitement de SMS, consultez le Application Broadcast Hub au chapitre 11.

Chasse aux coccinelles

Figure 5-1.



Les jeux font partie des applications pour appareils mobiles les plus passionnantes, à la fois pour jouer et pour créer. Le récent succès Angry Birds a été téléchargé 50 millions de fois au cours de sa première année et est joué plus d'un million d'heures chaque jour, selon Rovio, son développeur. (On parle même d'en faire un long métrage !) Bien que nous ne puissions pas garantir ce genre de succès, nous pouvons vous aider à créer vos propres jeux avec App Inventor, y compris celui-ci impliquant une coccinelle mangeant des pucerons tout en évitant une grenouille.

Ce que vous construirez

Dans ce jeu de « mâcheur à la première personne », l'utilisateur sera représenté par une coccinelle dont le mouvement sera contrôlé par l'inclinaison de l'appareil. Cela amène l'utilisateur dans le jeu d'une manière différente de MoleMash (chapitre 3), dans lequel l'utilisateur se trouvait à l'extérieur de l'appareil et y pénétrait.

L'application Ladybug Chase est illustrée à la figure 5-1. L'utilisateur peut :

- Contrôlez une coccinelle en inclinant l'appareil.
- Affichez une barre de niveau d'énergie sur l'écran, qui diminue avec le temps, conduisant à la famine de la coccinelle.
- Demandez à la coccinelle de chasser et de manger les pucerons pour gagner de l'énergie et éviter la famine.
- Aidez la coccinelle à éviter une grenouille qui veut manger il.



Figure 5-2. Le jeu Ladybug Chase dans le Designer

Ce que vous apprendrez

Vous devez parcourir l'application MoleMash au chapitre 3 avant de vous plonger dans ce chapitre, car cela suppose que vous connaissez la création de procédures, la génération de nombres aléatoires, le bloc if-then-else et les composants ImageSprite, Canvas, Sound et Clock .

En plus de revoir le matériel de MoleMash et d'autres chapitres précédents, ce chapitre présente :

- Utilisation de plusieurs composants ImageSprite et détection des collisions entre eux.
- Détection des inclinaisons de l'appareil avec un composant OrientationSensor et utilisation de celui-ci pour contrôler un ImageSprite.
- Modification de l'image affichée pour un ImageSprite.
- Dessiner des lignes sur un composant Canvas .
- Contrôler plusieurs événements avec un composant Clock .
- Utiliser des variables pour suivre des chiffres (le niveau d'énergie de la coccinelle).
- Création et utilisation de procédures paramétrées.
- Utilisation du bloc et .

Conception des composants

Cette application aura un canevas qui fournira un terrain de jeu pour trois composants ImageSprite : un pour la coccinelle, un pour le puceron et un pour la grenouille, qui nécessitera également un composant sonore pour son « rabbit ». L' OrientationSensor sera utilisé pour mesurer l'inclinaison de l'appareil pour déplacer la coccinelle, et une horloge sera utilisée pour changer la direction du puceron.

Il y aura une deuxième toile qui affichera le niveau d'énergie de la coccinelle. Un bouton Réinitialiser redémarrera le jeu si la coccinelle meurt de faim ou est mangée. Le tableau 5-1 fournit une liste complète des composants de cette application.

Tableau 5-1. Tous les composants du jeu Ladybug Chase

Type de composant	Groupe de palettes	Comment tu l'appelleras	But
Toile	Dessin et Animation	ChampCanvas	Terrain de jeu.

Type de composant	Groupe de palettes	Comment tu l'appelleras	But
ImageSprite	Dessin et Animation	Coccinelle	Lecteur contrôlé par l'utilisateur.
Capteurs d'orientation		Capteur d'orientation1	Déetectez l'inclinaison du téléphone pour contrôler le coccinelle.
Horloge	Interface utilisateur	Horloge1	Détermine quand modifier le Titres d'ImageSprites.
ImageSprite	Dessin et Animation	Puceron	La proie de la coccinelle.
ImageSprite	Dessin et Animation	Grenouille	Le prédateur de la coccinelle.
Toile	Dessin et Animation	ÉnergieCanvas	Affichez le niveau d'énergie de la coccinelle.
Bouton	Interface utilisateur	Bouton Redémarrer	Redémarrez le jeu.
Son	Médias	Son1	« Ribbit » quand la grenouille mange la coccinelle.

COMMENCER

Téléchargez les fichiers suivants :

- <http://appinventor.org/bookFiles/LadybugChase/ladybug.png>
- <http://appinventor.org/bookFiles/LadybugChase/aphid.png>
- http://appinventor.org/bookFiles/LadybugChase/dead_ladybug.png
- <http://appinventor.org/bookFiles/LadybugChase/frog.png>
- <http://appinventor.org/bookFiles/LadybugChase/frog.wav>

Ce sont des images de la coccinelle, du puceron, de la coccinelle morte et de la grenouille, ainsi qu'un son fuyez pour la tête de grenouille. Après les avoir téléchargés sur votre ordinateur, ajoutez-les à votre app dans la section Média du Designer.

Connectez-vous au site Web App Inventor et démarrez un nouveau projet. Nomme le "LadybugChase" et définissez également le titre de l'écran sur "Ladybug Chase". Ouvrez les blocs Editeur et connectez-vous à l'appareil.

PLACEMENT DES COMPOSANTS INITIAUX

Bien que les chapitres précédents vous aient demandé de créer tous les composants en même temps, c'est pas la façon dont les développeurs travaillent généralement. Au lieu de cela, il est plus courant de créer une partie d'un programme à la fois, testez-le, puis passez à la partie suivante du programme. Dans ce section, nous allons créer la coccinelle et contrôler son mouvement.

- Dans le Designer, créez un canevas, nommez-le FieldCanvas et définissez sa largeur sur « Fill parent » et sa hauteur sur 300 pixels.
- Placez un ImageSprite sur le canevas, renommez-le Ladybug et définissez sa propriété Picture sur l'image de la coccinelle. Ne vous inquiétez pas des valeurs des propriétés X et Y , car celles-ci dépendront de l'endroit où vous avez placé l' ImageSprite sur le canevas.

Comme vous l'avez peut-être remarqué, les ImageSprites ont également des propriétés Intervalle, Titre et Vitesse , que vous utiliserez dans ce programme :

- La propriété Interval , que vous pouvez définir sur 10 (millisecondes) pour ce jeu, spécifie la fréquence à laquelle ImageSprite doit se déplacer (par opposition à être déplacé par la procédure MoveTo , que vous avez utilisée pour MoleMash).
- La propriété Heading indique la direction dans laquelle l' ImageSprite doit se déplacer, en degrés. Par exemple, 0 signifie plein à droite, 90 signifie tout droit, 180 signifie plein à gauche, et ainsi de suite. Laissez le titre tel quel pour le moment ; nous allons le modifier dans l'éditeur de blocs.
- La propriété Speed spécifie le nombre de pixels que ImageSprite doit déplacer à chaque fois que son intervalle (10 millisecondes) passe. Nous définirons également la propriété Speed dans l'éditeur de blocs.

Le mouvement de la coccinelle sera contrôlé par un OrientationSensor, qui détecte la façon dont l'appareil est incliné. Nous souhaitons utiliser le composant Clock pour vérifier l'orientation de l'appareil toutes les 10 millisecondes (100 fois par seconde) et modifier le cap (direction) de la coccinelle en conséquence. Nous allons configurer cela dans l'éditeur de blocs comme suit :

1. Ajoutez un OrientationSensor, qui apparaîtra dans les « Composants non visibles » section.
2. Ajoutez une horloge, qui apparaîtra également dans la section « Composants non visibles », et définissez son TimerInterval sur 10 millisecondes. Vérifiez ce que vous avez ajouté par rapport à la figure 5-2.

Si vous utilisez un appareil autre que l'émulateur, vous devrez désactiver l'auto-rotation de l'écran, qui change la direction de l'affichage lorsque vous tournez l'appareil. Sélectionnez Screen1 et définissez sa propriété ScreenOrientation sur Portrait.

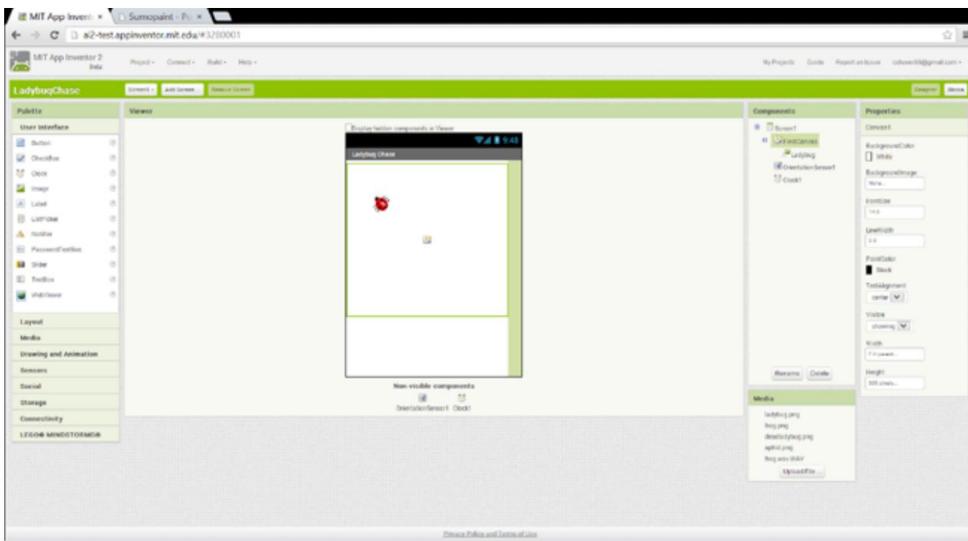


Figure 5-3. Configuration de l'interface utilisateur dans le Component Designer pour animer la coccinelle

Ajout de comportements aux composants

DÉPLACER LA COCCINELLE

En accédant à l'éditeur de blocs, créez la procédure `UpdateLadybug` et un bloc `Clock1.Timer`, comme indiqué dans la figure 5-3. Essayez de taper les noms de certains blocs (tels que « `Clock1.Timer` ») au lieu de les faire glisser hors des tiroirs. (Notez que l'opération appliquée au nombre 100 est une multiplication, indiquée par un astérisque, ce qui peut être difficile à voir sur la figure.) Vous n'avez pas besoin de créer les légendes de commentaires jaunes, bien que vous puissiez le faire en cliquant avec le bouton droit sur un bloc et en sélectionnant Ajouter un commentaire.

La procédure `UpdateLadybug` utilise deux des fonctionnalités les plus avancées d'`OrientationSensor`. propriétés utiles :

- Angle, qui indique la direction dans laquelle l'appareil est incliné (en degrés).
- Magnitude, qui indique le degré d'inclinaison, allant de 0 (pas d'inclinaison) à 1 (inclinaison maximale).

Multiplier la magnitude par 100 indique à la coccinelle qu'elle doit se déplacer entre 0 et 100 pixels dans le titre (direction) spécifié chaque fois que son `TimerInterval`, que vous avez précédemment défini sur 10 millisecondes dans le concepteur de composants, passe.

Bien que vous puissiez essayer cela sur l'appareil connecté, le mouvement de la coccinelle peut être à la fois plus lent et plus saccadé que si vous emballiez et téléchargez l'application sur l'appareil. Si après cela vous trouvez le mouvement de la coccinelle trop lent, augmentez le multiplicateur de vitesse. Si la coccinelle semble trop saccadée, diminuez-la.

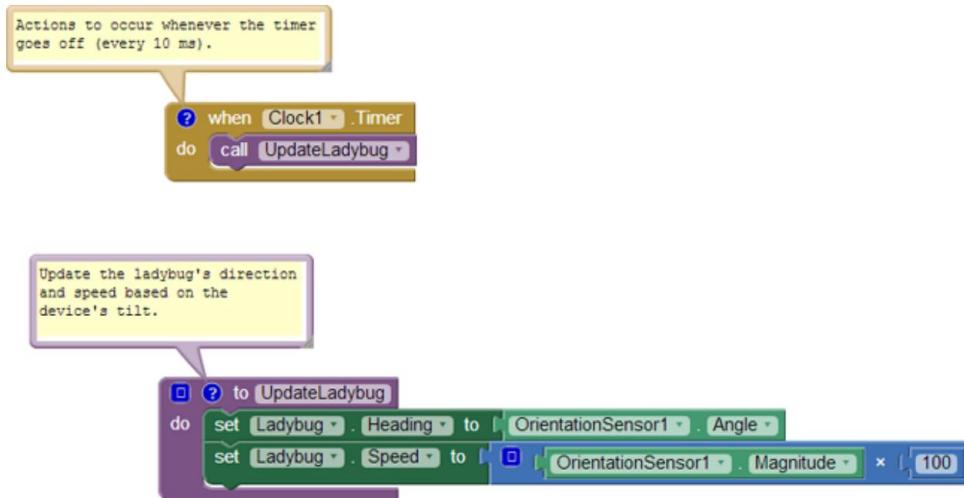


Figure 5-4. Changer le cap et la vitesse de la coccinelle toutes les 10 millisecondes

AFFICHAGE DU NIVEAU D'ÉNERGIE

Nous afficherons le niveau d'énergie de la coccinelle via une barre rouge dans un deuxième canevas. La ligne aura 1 pixel de haut et sa largeur sera du même nombre de pixels que l'énergie de la coccinelle, qui va de 200 (bien nourrie) à 0 (morte).

Ajout d'un composant

Dans le Designer, créez un nouveau canevas. Placez-le sous FieldCanvas et nommez-le EnergyCanvas. Définissez sa propriété Largeur sur « Remplissage parent » et sa Hauteur sur 1 pixel.

Créer une variable : énergie

Dans l'éditeur de blocs, vous devrez créer une énergie variable avec une valeur initiale de 200 pour suivre le niveau d'énergie de la coccinelle, comme le montre la figure 5-4. (Comme vous vous en souvenez peut-être, nous avons d'abord utilisé une variable, `dotSize`, dans l'application PaintPot au chapitre 2.) Voici comment procéder :

1. Dans l'éditeur de blocs, faites glisser un nom global d'initialisation à bloquer. Remplacez le texte « nom » par « énergie ».
2. Créez un bloc numéro 200 (soit en commençant à taper le nombre 200, soit en en faisant glisser un bloc numérique depuis le tiroir Math) et branchez-le à l'énergie globale, comme le montre la figure 5-4.

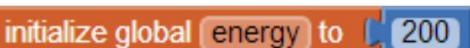


Figure 5-5. Initialisation de l'énergie variable à 200

La figure 5-5 illustre que lorsque vous définissez une variable, de nouveaux blocs set et get sont créé pour cela et auquel vous pouvez accéder en passant la souris sur le nom de la variable.



Figure 5-6. Lorsque vous passez la souris sur une variable dans le bloc global d'initialisation, vous pouvez faire glisser les blocs set et get pour la variable.

Dessiner la barre énergétique

Nous voulons communiquer le niveau d'énergie avec une barre rouge, qui a une longueur en pixels égale à la valeur d'énergie. Pour ce faire, nous pourrions créer deux ensembles de blocs similaires comme suit :

1. Tracez une ligne rouge de (0, 0) à (énergie, 0) dans FieldCanvas pour afficher le courant niveau d'énergie.
2. Tracez une ligne blanche de (0, 0) à (EnergyCanvas.Width, 0) dans FieldCanvas pour effacer le niveau d'énergie actuel avant de dessiner le nouveau niveau.

Cependant, une meilleure alternative consiste à créer une procédure capable de tracer une ligne de n'importe quelle longueur et de n'importe quelle couleur dans FieldCanvas. Pour ce faire, nous devons spécifier deux paramètres, la longueur et la couleur, lorsque notre procédure est appelée, tout comme nous devions spécifier les valeurs des paramètres dans MoleMash lorsque nous avons appelé la procédure intégrée d'entiers aléatoires . Voici les étapes de création d'une procédure DrawEnergyLine :

1. Accédez au tiroir Procédures et faites glisser un bloc de procédure vers . Choisissez la version qui contient « do » plutôt que « return » car notre procédure ne renverra pas de valeur.
2. Cliquez sur le nom de la procédure (« procédure ») et remplacez-le par « DrawEnergyLine ».
3. En haut à gauche du nouveau bloc, cliquez sur le petit carré bleu. Cela ouvre la fenêtre illustrée à gauche de la figure 5-6.
4. Depuis le côté gauche de cette fenêtre, faites glisser une entrée vers la droite, en changeant son nom de « x » à « longueur ». Cela indique que la procédure aura un paramètre nommé « longueur ».
5. Répétez l'opération pour un deuxième paramètre nommé « couleur », qui doit se placer en dessous de celui nommé « longueur ». Cela devrait ressembler au côté droit de la figure 5-6.

6. Cliquez à nouveau sur l'icône bleue pour fermer la fenêtre de saisie.

7. Remplissez le reste de la procédure comme indiqué dans la figure 5-7. Vous pouvez trouver la couleur et la longueur en passant la souris sur leurs noms dans la définition de la procédure.

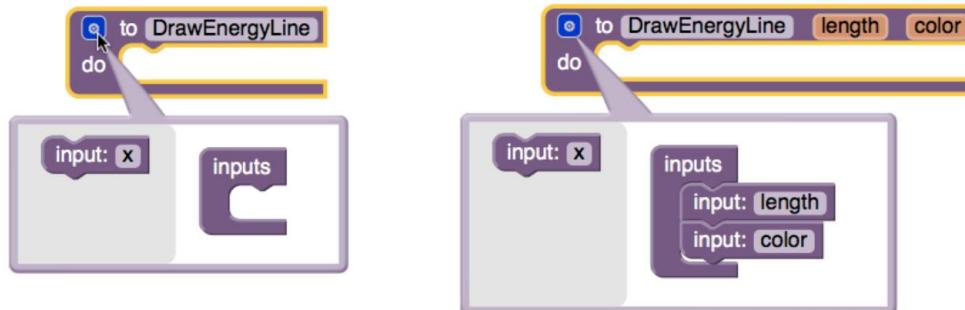


Figure 5-7. Ajout d'entrées (paramètres) à la procédure DrawEnergyLine

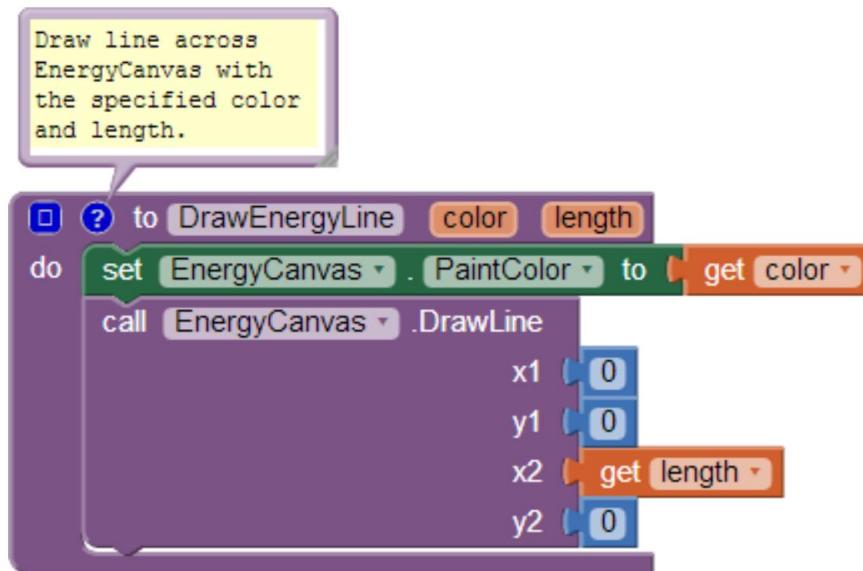


Figure 5-8. Définir la procédure DrawEnergyLine

Maintenant que vous savez créer vos propres procédures, écrivons également une procédure DisplayEnergy qui appelle DrawEnergyLine deux fois : une fois pour effacer l'ancienne ligne (en traçant une ligne blanche sur tout le canevas), et une fois pour afficher l'ancienne ligne. nouvelle ligne, comme le montre la figure 5-8.

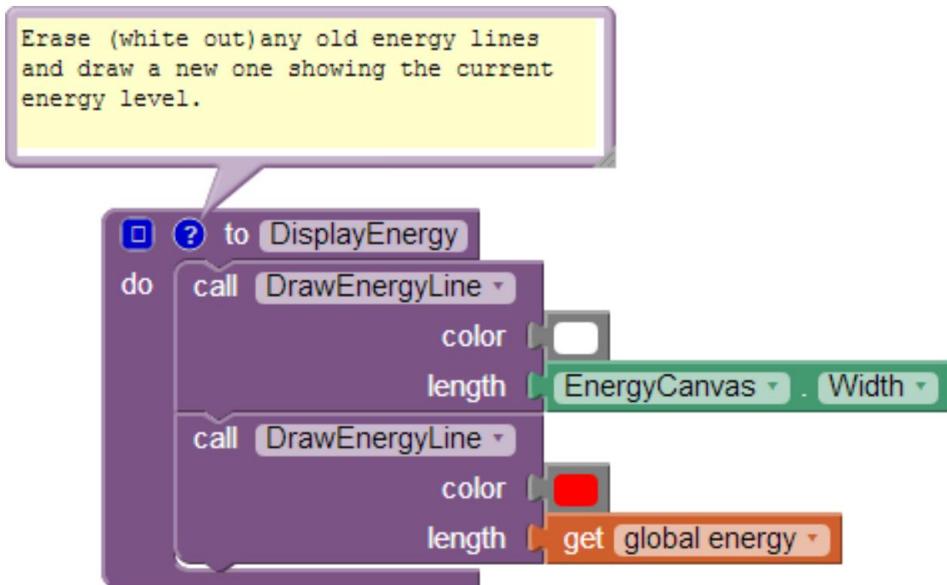


Figure 5-9. Définition de la procédure DisplayEnergy

La procédure DisplayEnergy se compose de quatre lignes qui effectuent les opérations suivantes :

1. Définissez la couleur de la peinture sur blanc.
2. Tracez une ligne tout au long d' EnergyCanvas (qui ne mesure que 1 pixel de haut).
3. Définissez la couleur de la peinture sur rouge.
4. Tracez une ligne dont la longueur en pixels est la même que la valeur énergétique.



Remarque Le processus de remplacement du code commun par des appels à une nouvelle procédure est appelé refactoring, un ensemble de techniques puissantes permettant de rendre les programmes plus maintenables et plus fiables. Dans ce cas, si nous voulions modifier la hauteur ou l'emplacement de la ligne d'énergie, nous n'aurions qu'à apporter une seule modification à DrawEnergyLine, plutôt que de modifier chaque appel.

FAMINE

Contrairement aux applications des chapitres précédents, ce jeu a une fin : c'est terminé si la coccinelle ne mange pas suffisamment de pucerons ou si elle est elle-même mangée par la grenouille. Dans l'un ou l'autre de ces cas, nous voulons que la coccinelle arrête de bouger (ce que nous pouvons faire en définissant Ladybug.Enabled sur false) et que l'image passe d'une coccinelle vivante à une coccinelle morte.

un (ce que nous pouvons faire en remplaçant `Ladybug.Picture` par le nom de l'image téléchargée appropriée). Créez la procédure `GameOver` comme indiqué dans la figure 5-9.

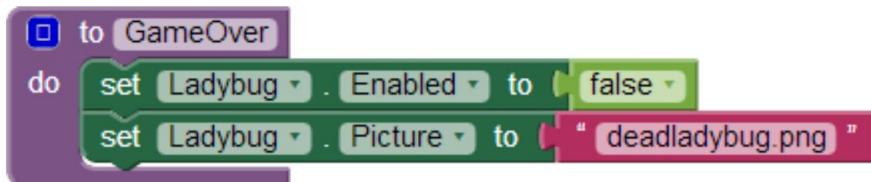


Figure 5-10. Définir la procédure `GameOver`

Ensuite, ajoutez le code souligné en rouge dans la figure 5-10 à `UpdateLadybug` (qui, comme vous pourrez rappeler, est appelé par `Clock.Timer` toutes les 10 millisecondes) à ce qui suit :

- Diminuer son niveau d'énergie.
- Afficher le nouveau niveau.
- Terminez le jeu si l'énergie est à 0.

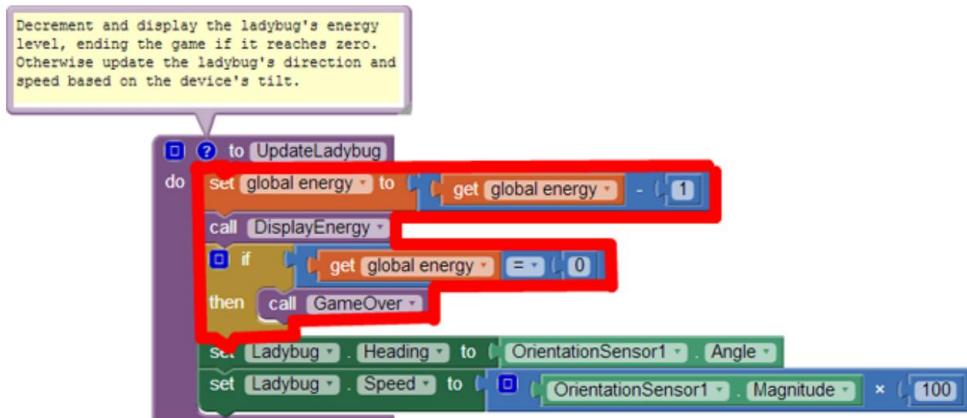


Figure 5-11. Deuxième version de la procédure `UpdateLadybug`



Testez votre application Sur votre appareil, vérifiez que le niveau d'énergie diminue avec le temps, provoquant éventuellement la disparition de la coccinelle.

AJOUTER UN PUCERON

L'étape suivante consiste à ajouter un puceron. Plus précisément, un puceron devrait s'adapter autour de `FieldCanvas`. Si la coccinelle rencontre le puceron (et le « mange » ainsi), le niveau d'énergie de la coccinelle devrait augmenter et le puceron devrait disparaître, pour être remplacé un peu par un autre.

plus tard. (Du point de vue de l'utilisateur, ce sera un puceron différent, mais ce sera en réalité le même composant ImageSprite .)

Ajout d'un ImageSprite

La première étape que vous devez entreprendre pour ajouter un puceron est de revenir dans le Designer et de créer un autre ImageSprite, en veillant à ne pas le placer sur la coccinelle. Il doit être renommé Puceron et ses propriétés définies comme suit :

- Définissez sa propriété Picture sur le fichier image de puceron que vous avez téléchargé.
- Définissez sa propriété Intervalle sur 10, ainsi, comme la coccinelle, elle se déplace tous les 10. millisecondes.
- Réglez sa Vitesse sur 2, pour qu'elle ne bouge pas trop vite pour que la coccinelle ne l'attrape.

Ne vous inquiétez pas de ses propriétés x et y (tant qu'elle n'est pas au-dessus de la coccinelle) ou sa propriété Heading , qui sera définie dans l'éditeur de blocs.

Contrôler le puceron

En expérimentant, nous avons constaté que cela fonctionnait mieux pour le puceron en changeant de direction environ une fois toutes les 50 millisecondes (5 « ticks » de Clock1). Une approche pour activer ce comportement consisterait à créer une deuxième horloge avec un TimerInterval de 50 millisecondes. Cependant, nous aimerais que vous essayiez une technique différente afin que vous puissiez en apprendre davantage sur le bloc de fractions aléatoires , qui renvoie un nombre aléatoire supérieur ou égal à 0 et inférieur à 1 à chaque fois qu'il est appelé. Créez la procédure UpdateAphid illustrée à la figure 5-11 et ajoutez-y un appel dans Clock1.Timer.

Comment fonctionnent les blocs

Chaque fois que le minuteur expire (100 fois par seconde), UpdateLadybug (comme avant) et UpdateAphid sont appellés. La première chose qui se produit dans UpdateAphid est qu'une fraction aléatoire entre 0 et 1 est générée, par exemple 0,15. Si ce nombre est inférieur à 0,20 (ce qui arrivera 20% du temps), le puceron changera de direction selon un nombre aléatoire de degrés compris entre 0 et 360. Si ce nombre n'est pas inférieur à 0,20 (ce qui sera le cas, le restant 80% du temps), le puceron restera le cours.

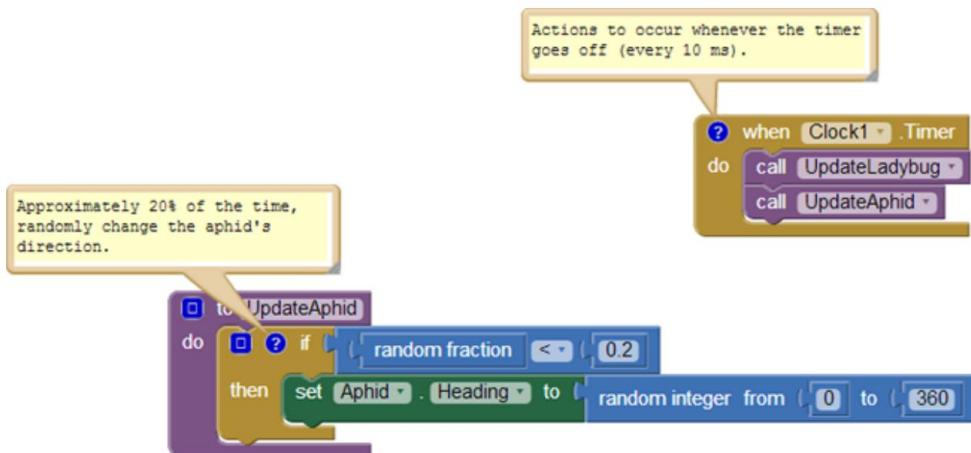


Figure 5-12. Ajout de la procédure UpdateAphid

PROGRAMMER LA COCCINELLE POUR MANGER LE PUCERON

L'étape suivante consiste à configurer la coccinelle pour qu'elle « mange » les pucerons lorsqu'ils entrent en collision.

Heureusement, App Inventor fournit des blocs pour détecter les collisions entre les composants d'ImageSprite , ce qui soulève la question : que doit-il se passer lorsque la coccinelle et le puceron entrent en collision ? Vous voudrez peut-être vous arrêter et réfléchir à cela avant de continuer à lire.

Pour gérer ce qui se passe lorsque la coccinelle et le puceron entrent en collision, créons un procédure, EatAphid, qui effectue les opérations suivantes :

- Augmente le niveau d'énergie de 50 pour simuler la dégustation d'une délicieuse friandise.
- Provoque la disparition du puceron (en mettant sa propriété Visible à false).
- Provoque l'arrêt du puceron (en définissant sa propriété Enabled sur false).
- Provoque le déplacement du puceron vers un emplacement aléatoire sur l'écran. (Cela suit le même modèle que le code pour déplacer la taupe dans MoleMash).

Vérifiez que vos blocs correspondent à la figure 5-12. Si vous aviez d'autres idées sur ce qui devrait se produire, comme des effets sonores, vous pouvez également les ajouter.

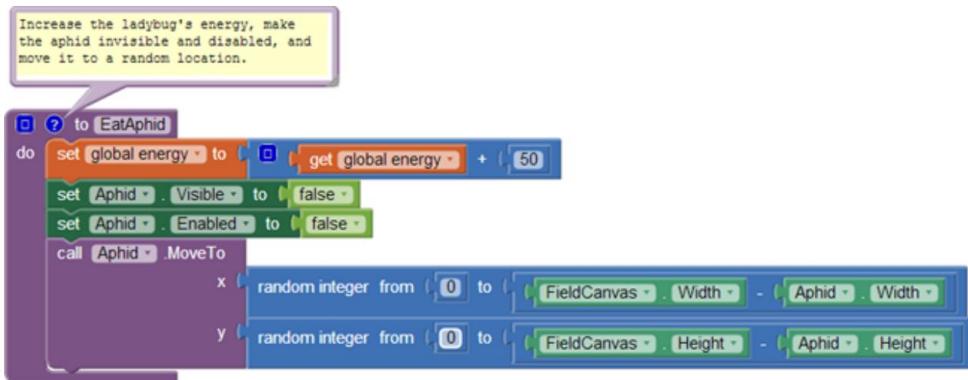


Figure 5-13. Ajout de la procédure EatAphid

Comment fonctionnent les blocs

Chaque fois que EatAphid est appelé, il ajoute 50 à l'énergie variable, évitant ainsi la famine à la coccinelle. Ensuite, les propriétés Visible et Enabled du puceron sont définies sur false, de sorte qu'il semble disparaître et s'arrêter de bouger, respectivement. Enfin, des coordonnées x et y aléatoires sont générées pour un appel à Aphid.MoveTo afin que, lorsque le puceron réapparaît, il se trouve dans un nouvel emplacement (sinon, il serait mangé dès sa réémergence).

DÉTECTION D'UNE COLLISION COCCINELLE-PUCERON

La figure 5-13 montre le code permettant de détecter les collisions entre la coccinelle et le puceron.

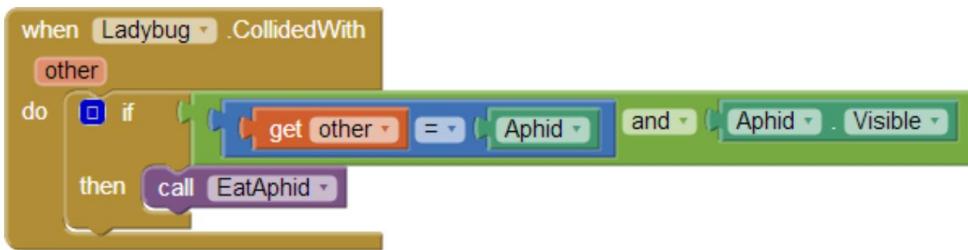


Figure 5-14. Déetecter et agir sur les collisions entre coccinelle et puceron

Comment fonctionnent les blocs

Lorsque la coccinelle entre en collision avec un autre ImageSprite, Ladybug.CollidedWith est appelé, avec le paramètre « autre » lié à ce avec quoi la coccinelle est entrée en collision. Pour l'instant, la seule chose avec laquelle il peut entrer en collision est le puceron, mais nous ajouterons une grenouille plus tard. Nous utiliserons une programmation défensive et vérifierons explicitement que la collision a eu lieu avec le puceron avant d'appeler EatAphid. Il y a aussi un contrôle pour confirmer que le puceron est visible.

Autrement, après avoir mangé un puceron mais avant sa réapparition, il pourrait entrer en collision avec le

encore une coccinelle. Sans ce contrôle, le puceron invisible serait à nouveau mangé, provoquant un nouveau saut d'énergie sans que l'utilisateur ne comprenne pourquoi.



Remarque La programmation défensive consiste à écrire du code de manière à ce qu'il soit toujours susceptible de fonctionner même si le programme est modifié. Dans la Figure 5-13, le test « autre = Puceron » n'est pas strictement nécessaire car la seule chose avec laquelle la coccinelle peut actuellement entrer en collision est le puceron, mais avoir la vérification empêchera notre programme de mal fonctionner si nous ajoutons un autre ImageSprite et oubliions de modifier Ladybug.CollidedWith. Les programmeurs passent généralement plus de temps à corriger les bogues (la variété des logiciels, pas ceux qui mangent des pucerons) qu'à écrire du nouveau code, il vaut donc la peine de prendre un peu de temps pour écrire du code de manière à éviter les problèmes en premier lieu.

LE RETOUR DU PUCERON

Pour que le puceron finisse par réapparaître, vous devez modifier UpdateAphid comme indiqué dans la figure 5-14 afin qu'il change la direction du puceron uniquement s'il est visible. (Le changer s'il est invisible est une perte de temps.) Si le puceron n'est pas visible (par exemple, s'il a été mangé récemment), il y a 1 chance sur 20 (5 %) qu'il soit réactivé. en d'autres termes, rendu éligible pour être mangé à nouveau.

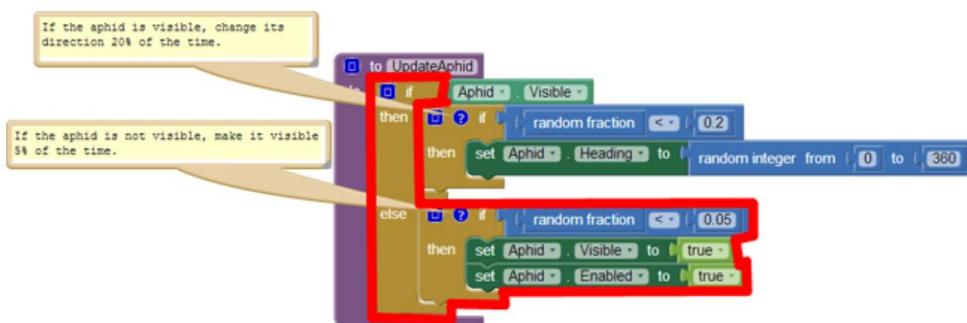


Figure 5-15. Modification de UpdateAphid pour redonner vie aux pucerons invisibles

Comment fonctionnent les blocs

UpdateAphid devient assez complexe, examinons donc attentivement son comportement :

- Si le puceron est visible (ce qui sera le cas sauf s'il vient d'être mangé), UpdateAphid se comporte comme nous l'avons écrit pour la première fois. Plus précisément, il y a 20 % de chances que sa direction change.

- Si le puceron n'est pas visible (c'est-à-dire s'il a été mangé récemment), alors la partie else du bloc if else s'exécutera. Un nombre aléatoire est alors généré. S'il est inférieur à 0,05 (ce qui sera 5 % du temps), le puceron redevient visible et est activé, ce qui le rend éligible pour être à nouveau mangé.

Étant donné que UpdateAphid est appelé par Clock1.Timer, qui se produit toutes les 10 millisecondes, et qu'il y a une chance sur 20 (5 %) que le puceron redevienne visible, le puceron prendra en moyenne 200 millisecondes (1/5 de seconde) pour réapparaître.

AJOUT D'UN BOUTON DE REDÉMARRAGE

Comme vous l'avez peut-être remarqué en testant l'application avec votre nouvelle fonctionnalité mangeuse de pucerons, le jeu a vraiment besoin d'un bouton Redémarrer. (C'est une autre raison pour laquelle il est utile de concevoir et de créer votre application en petits morceaux, puis de la tester : vous découvrez souvent des éléments que vous avez négligés, et il est plus facile de les ajouter au fur et à mesure de votre progression que d'y revenir et de les modifier après le processus. l'application est terminée.) Dans le Concepteur de composants, ajoutez un composant Button sous EnergyCanvas, renommez-le « RestartButton » et définissez sa propriété Text sur « Restart ».

Dans l'éditeur de blocs, créez le code illustré à la figure 5-15 pour effectuer les opérations suivantes lorsque le RestartButton est cliqué :

1. Remettez le niveau d'énergie à 200.
2. Réactivez le puceron et rendez-le visible.
3. Réactivez la coccinelle et changez son image en coccinelle vivante (sauf si tu veux des coccinelles zombies !).

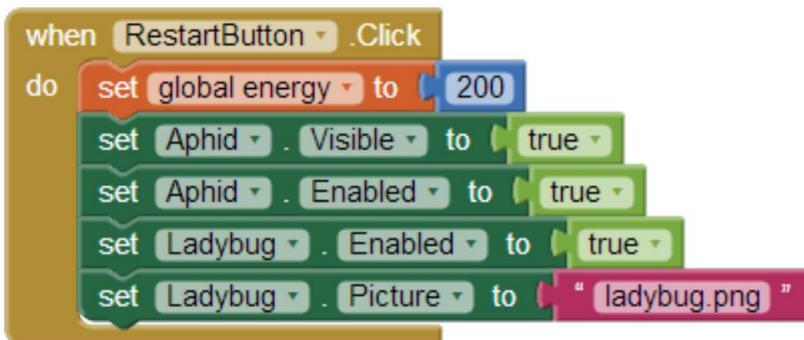


Figure 5-16. Redémarrer le jeu lorsque RestartButton est enfoncé

AJOUT DE LA GRENOUILLE

À l'heure actuelle, garder la coccinelle en vie n'est pas trop difficile. Nous avons besoin d'un prédateur. Plus précisément, nous ajouterons une grenouille qui se dirige directement vers la coccinelle. S'ils entrent en collision, la coccinelle devient le dîner et le jeu se termine.

Faire en sorte que la grenouille chasse la coccinelle

La première étape pour configurer la grenouille pour chasser la coccinelle consiste à revenir au concepteur de composants et à ajouter un troisième ImageSprite, Frog, à FieldCanvas. Définissez sa propriété Image sur l'image appropriée, son Intervalle sur 10 et sa Vitesse sur 1, car elle devrait se déplacer plus lentement que les autres créatures.

La figure 5-16 montre UpdateFrog, une nouvelle procédure que vous devez créer et appeler depuis Horloge1.Timer.

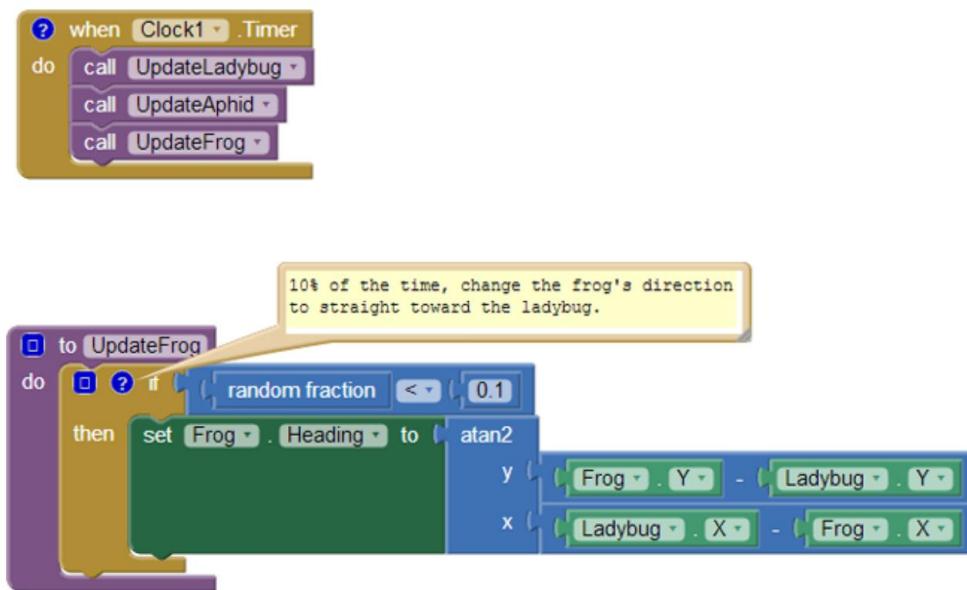


Figure 5-17. Faire avancer la grenouille vers la coccinelle

Comment fonctionnent les blocs

À présent, vous devriez être familier avec l'utilisation du bloc de fractions aléatoires pour qu'un événement se produise avec une certaine probabilité. Dans ce cas, il y a 10 % de chances que la direction de la grenouille soit modifiée pour se diriger droit vers la coccinelle. Cela nécessite de la trigonométrie, mais ne paniquez pas, vous n'avez pas besoin de le découvrir vous-même. App Inventor gère une tonne de fonctions mathématiques pour vous, même des choses comme la trigonométrie. Dans ce cas, vous souhaitez utiliser le bloc atan2 (arctangent), qui renvoie l'angle correspondant à un ensemble donné de valeurs x et y. (Pour ceux d'entre vous qui connaissent la trigonométrie, la raison

l'argument y de atan2 a le signe opposé à ce à quoi vous vous attendez (l'ordre opposé des arguments à soustraire) : la coordonnée y augmente vers le bas sur un canevas Android, à l'opposé de ce qui se produirait dans une coordonnée xy standard. système.)

Préparer la grenouille à manger la coccinelle

Nous devons maintenant modifier le code de collision pour que si la coccinelle entre en collision avec la grenouille, le niveau d'énergie et la barre passent à 0 et le jeu se termine, comme le montre la figure 5-17.

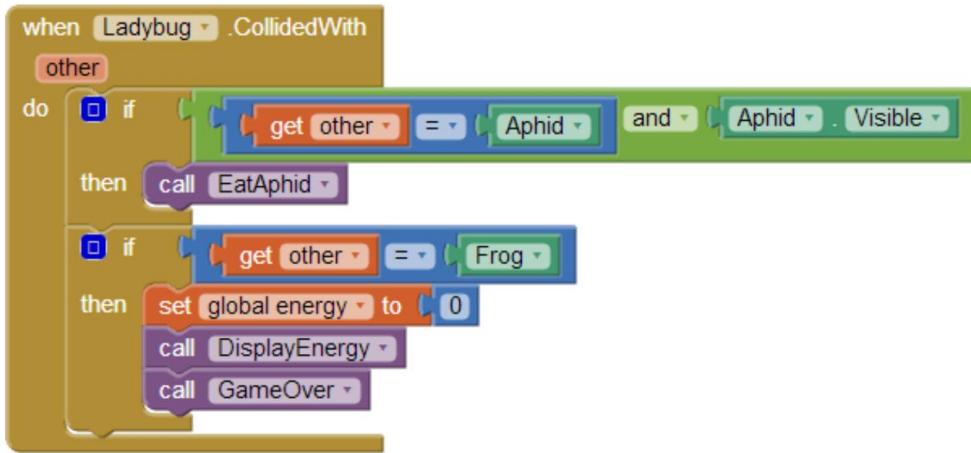


Figure 5-18. Faire manger la coccinelle à la grenouille

Comment fonctionnent les blocs

En plus du premier if, qui vérifie si la coccinelle est entrée en collision avec le puceron, il existe désormais un deuxième if, qui vérifie si la coccinelle est entrée en collision avec la grenouille. Si la coccinelle et la grenouille entrent en collision, trois choses se produisent :

1. L'énergie variable descend à 0, car la coccinelle a perdu sa force vitale.
2. DisplayEnergy est appelé pour effacer la ligne d'énergie précédente (et dessiner la nouvelle, vide).
3. La procédure que vous avez écrite plus tôt, GameOver, est appelée pour empêcher la coccinelle de bouger et changer son image en celle d'une coccinelle morte.

LE RETOUR DE LA COCCINELLE

RestartButton.Click a déjà du code pour remplacer l'image de la coccinelle morte par celle de la coccinelle vivante. Maintenant, vous devez ajouter du code pour déplacer la coccinelle vivante vers un emplacement aléatoire. (Pensez à ce qui se passerait si vous ne bougeiez pas le

coccinelle au début d'un nouveau jeu. Où serait-il par rapport à la grenouille ?)

La figure 5-18 montre les blocs permettant de déplacer la coccinelle au redémarrage du jeu.

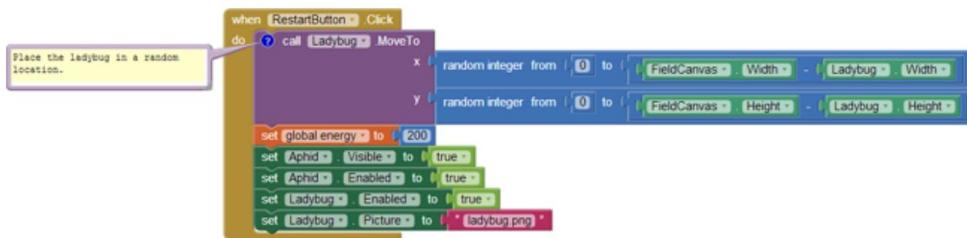


Figure 5-19. La version finale de RestartButton.Click

Comment fonctionnent les blocs

La seule différence entre cette version de RestartButton.Click et la version précédente est le bloc Ladybug.MoveTo et ses arguments. La fonction intégrée Random Integer est appelée deux fois : une fois pour générer une coordonnée x légale et une fois pour générer une coordonnée y légale. Même si rien n'empêche de placer la coccinelle au-dessus du puceron ou de la grenouille, les chances sont contre.



Testez votre application Redémarrez le jeu et assurez-vous que la coccinelle apparaît dans un nouvel emplacement aléatoire.

AJOUTER DES EFFETS SONORES

Lorsque vous avez testé le jeu, vous avez peut-être remarqué qu'il n'y a pas de très bons retours lorsque quelque chose est mangé. Pour ajouter des effets sonores et un retour tactile, procédez comme suit :

1. Dans le Concepteur de composants, ajoutez un composant Son . Définissez sa source sur fichier audio que vous avez téléchargé.
2. Accédez à l'éditeur de blocs, où vous pourrez :
 - Faites vibrer l'appareil lorsqu'un puceron est mangé en ajoutant un Bloc Sound1.Vibrate avec un argument de 100 (millisecondes) dans EatAphid.
 - Faites rire la grenouille lorsqu'elle mange la coccinelle en ajoutant un appel à Sound1.Play dans Ladybug.CollidedWith juste avant l'appel à GameOver.

L'application complète : Ladybug Chase

La figure 5-19 montre la configuration finale du bloc pour Ladybug Chase.

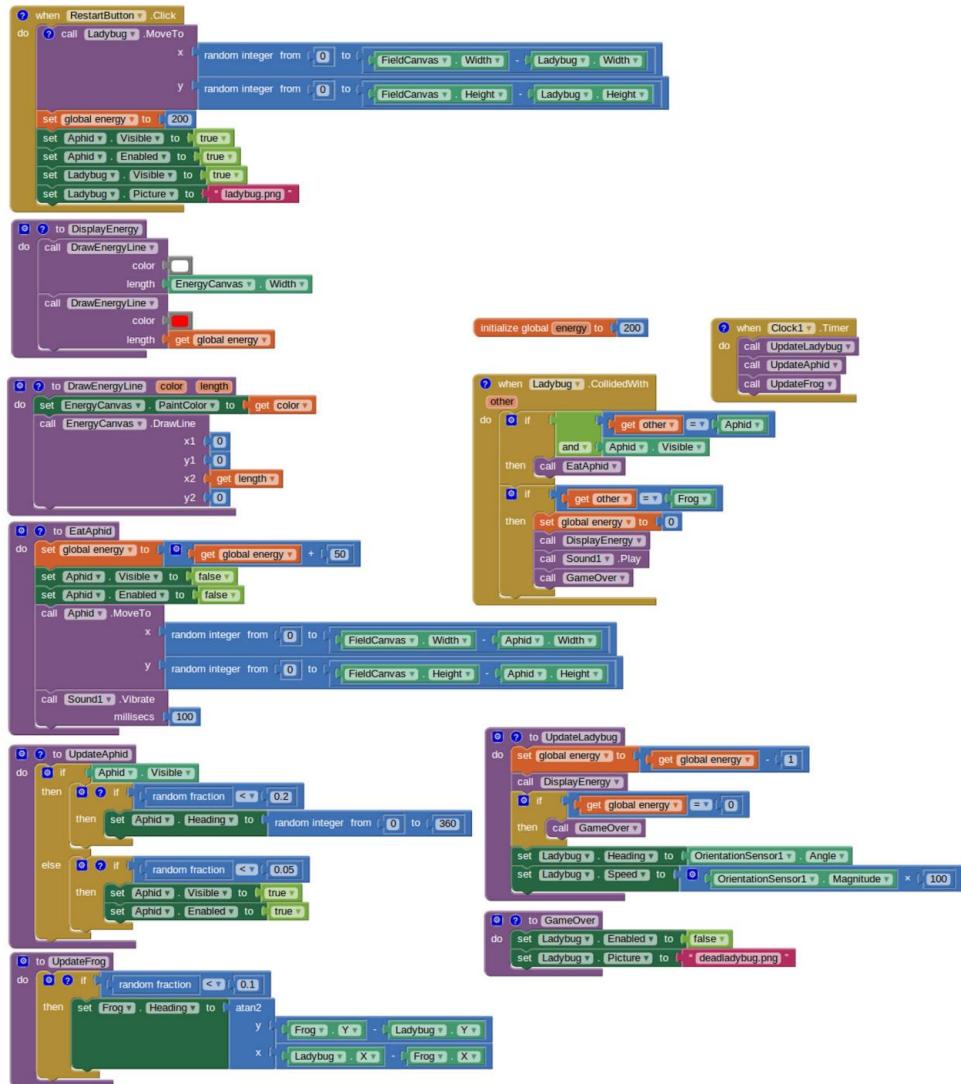


Figure 5-20. L'application complète Ladybug Chase

Variantes

Voici quelques idées pour améliorer ou personnaliser ce jeu :

- Actuellement, la grenouille et le puceron continuent de se déplacer après la fin du jeu. Empêchez cela en définissant leurs propriétés Enabled sur false dans GameOver et true dans RedémarrerButton.Cliquez.

- Afficher un score indiquant combien de temps la coccinelle est restée en vie. Vous pouvez le faire en créant une étiquette que vous incrémentez dans Clock1.Timer.
- Rendre la barre d'énergie plus visible en augmentant la hauteur d' EnergyCanvas à 2 et en traçant deux lignes, l'une au-dessus de l'autre, dans DrawEnergyLine. (C'est un autre avantage d'avoir une procédure plutôt qu'un code dupliqué pour effacer et redessiner la ligne d'énergie : il vous suffit d'effectuer une modification à un endroit pour changer la taille – ou la couleur, ou l'emplacement – de la ligne.)
- Ajoutez une ambiance avec une image d'arrière-plan et davantage d'effets sonores, tels que des sons de la nature ou un avertissement lorsque le niveau d'énergie de la coccinelle devient faible.
- Faire en sorte que le jeu devienne plus difficile au fil du temps, par exemple en augmentant la vitesse de la grenouille propriété ou en diminuant sa propriété Intervalle.
- Techniquement, la coccinelle devrait disparaître lorsqu'elle est mangée par la grenouille. Changez le jeu pour que la coccinelle devienne invisible si elle est mangée par la grenouille mais pas si elle meurt de faim.
- Remplacez les images de coccinelle, de puceron et de grenouille par d'autres à votre goût, comme un hobbit, un orc et un sorcier maléfique ou un chasseur rebelle, une capsule énergétique et un chasseur impérial.

Résumé

Avec maintenant deux jeux à votre actif (si vous avez suivi le tutoriel MoleMash), vous savez désormais comment créer vos propres jeux, ce qui est l'objectif de nombreux nouveaux programmeurs ou aspirants ! Plus précisément, vous avez appris :

- Vous pouvez avoir plusieurs composants ImageSprite (la coccinelle, le puceron et la grenouille) et détecter les collisions entre eux.
- L' OrientationSensor peut détecter l'inclinaison de l'appareil et vous pouvez utiliser cette valeur pour contrôler le mouvement d'un sprite (ou tout ce que vous pouvez imaginer).
- Un seul composant Clock peut contrôler plusieurs événements qui se produisent à la même fréquence (changements de direction de la coccinelle et de la grenouille), ou à des fréquences différentes, en utilisant le bloc de fractions aléatoires . Par exemple, si vous souhaitez qu'un événement se produise environ un quart (25 %) du temps, placez-le dans le corps d'un bloc if qui n'est exécuté que lorsque le résultat d'une fraction aléatoire est inférieur à 0,25.
- Vous pouvez avoir plusieurs composants Canvas dans une seule application, ce que nous avons fait pour avoir un terrain de jeu et afficher une variable graphiquement (plutôt que via une étiquette).

- Vous pouvez définir des procédures avec des paramètres (tels que « couleur » et « longueur » dans DrawEnergyLine) qui contrôlent le comportement, augmentant ainsi considérablement la puissance de l'abstraction procédurale.

Figure 6-1.



Dans ce chapitre, vous allez créer une application de guide touristique pour un voyage à Paris.

Créer une application cartographique entièrement fonctionnelle peut sembler très compliqué, mais App Inventor fournit deux composants de haut niveau pour vous aider : ActivityStarter , qui vous permet de lancer une autre application à partir de votre application, y compris Google Maps, et WebViewer , qui affiche n'importe quelle page Web de votre choix dans un sous-panneau de votre application. Vous explorerez ces deux composants et créerez deux versions différentes d'un guide touristique.

Ce que vous apprendrez

Ce chapitre présente les composants et concepts suivants d'App Inventor :

- Le composant Activity Starter pour lancer d'autres applications Android depuis votre application.
- Le composant WebViewer pour afficher les pages Web dans votre application.
- Comment utiliser les variables de liste pour stocker des informations sur votre application.

- Le composant ListPicker pour donner à l'utilisateur la possibilité de choisir parmi une liste de Emplacements.
- Comment créer une URL dynamiquement pour afficher différentes cartes.

Conception des composants

Créez un nouveau projet dans App Inventor et appelez-le « ParisMapTour ». L'interface utilisateur pour l'application dispose d'un composant Image avec une photo de Paris, d'un composant Label avec du texte, un composant ListPicker fourni avec un bouton associé, et dans ce première version, un composant ActivityStarter (non visible). Vous pouvez concevoir le composants à l'aide de l'instantané de la figure 6-1.



Figure 6-2. L'application Paris Map Tour fonctionnant dans l'émulateur

Vous aurez besoin des composants répertoriés dans le tableau 6-1 pour créer cette application. Faites glisser chacun composant de la palette dans la visionneuse et nommez-le comme spécifié.

Tableau 6-1. Composants pour le Paris Map Tour

Type de composant	Groupe de palettes	Comment vous l'appellerez	But
Image	Image de l'interface	utilisateur1	Afficher une image statique de Paris à l'écran.
Étiquette	Étiquette de l'interface	utilisateur1	Affichez le texte « Découvrez Paris avec votre Android ! »
Sélecteur de liste	Interface utilisateur	ListPicker1	Lorsque vous cliquez dessus, une liste de choix de destination apparaîtra apparaître.
Connectivité	ActivityStarter	ActivityStarter1	Lancez l'application Maps lorsqu'une destination est choisie.

Définition des propriétés d'ActivityStarter

ActivityStarter est un composant avec lequel vous pouvez lancer n'importe quelle application Android, y compris Google Maps ou une autre de vos propres applications. Vous allez d'abord créer ParisMapTour de manière à ce que l'application Cartes soit lancée pour afficher des cartes particulières en fonction du choix de l'utilisateur. L'utilisateur peut ensuite appuyer sur le bouton Retour pour revenir à votre application et choisir une destination différente.

ActivityStarter est un composant de niveau relativement bas dans la mesure où vous devrez définir certains propriétés avec des informations qui seraient familières à un programmeur Java Android SDK, mais complètement étrangères aux 99,99% restants du monde. Pour cette application, entrez les propriétés comme spécifié dans le tableau 6-2 et soyez prudent : elles sont sensibles à la casse, ce qui signifie qu'il est important qu'une lettre soit en majuscule ou en minuscule.

Tableau 6-2. Propriétés ActivityStarter pour le lancement de Google Maps

Propriété	Valeur
Action	android.intent.action.VIEW
ActivityClass	com.google.android.maps.MapActivity
ActivityPackage	com.google.android.apps.maps

Dans l'éditeur de blocs, vous définirez une propriété supplémentaire, DataUri, qui vous permettra de fournir une URL pour lancer une carte spécifique dans Google Maps. Cette propriété doit être définie dans l'éditeur de blocs au lieu du concepteur de composants car elle doit être dynamique : elle changera selon que l'utilisateur choisit de visiter la Tour Eiffel, le Louvre ou la Cathédrale Notre-Dame.

Nous reviendrons sur l'éditeur de blocs dans un instant, mais il y a quelques détails supplémentaires à prendre en compte avant de pouvoir passer à la programmation du comportement de vos composants :

1. Téléchargez le fichier metro.jpg à charger dans votre projet. Ensuite, définissez-le comme propriété Picture de Image1.
2. Le composant ListPicker est livré avec un bouton ; lorsque l'utilisateur clique dessus, les choix sont répertoriés. Définissez le texte de ce bouton en modifiant la propriété Text de ListPicker1 en « Choisir Paris Destination ».

Ajout de comportements aux composants

Dans l'éditeur de blocs, vous devrez définir une liste de destinations et deux comportements :

- Lorsque l'application démarre, l'application charge les destinations dans ListPicker. composant afin que l'utilisateur puisse en choisir un.
- Lorsque l'utilisateur choisit une destination dans ListPicker, l'application Maps est lancé et affiche une carte de cette destination. Dans cette première version de l'application, vous allez simplement ouvrir Maps et lui demander de lancer une recherche pour la destination choisie.

CRÉER UNE LISTE DE DESTINATIONS

Ouvrez l'éditeur de blocs et créez une variable avec la liste des destinations parisiennes en utilisant les blocs répertoriés dans le tableau 6-3.

Tableau 6-3. Blocs pour créer une variable de destinations

Type de bloc	Objectif du tiroir
initialiser les variables globales (« Destinations »)	Créer une liste des destinations.
fais une liste	Listes
texte (« Tour Eiffel »)	Texte
texte (« Musée du Louvre »)	Texte
texte (« Cathédrale Notre Dame »)	Texte

Lorsque vous faites glisser le bloc Créer une liste dans votre application, il n'en aura que deux disponibles prises. Vous pouvez en ajouter un autre en cliquant sur l'icône bleu foncé et en ajoutant un troisième article.

Après avoir fait cela, créez simplement les blocs de texte pour chaque destination et lieu. placez-les dans les trois emplacements pour créer une liste, comme le montre la figure 6-2.

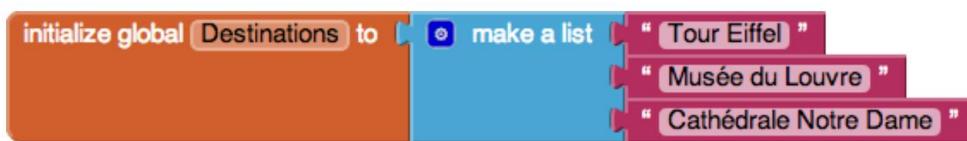


Figure 6-3. Une liste de trois éléments

LAISSER L'UTILISATEUR CHOISIR UNE DESTINATION

La liste que vous venez de définir n'apparaît pas dans l'interface utilisateur (aucune variable n'y apparaît). Vous allez utiliser un composant ListPicker pour afficher la liste des éléments parmi lesquels l'utilisateur peut choisir.

Vous préchargez les choix dans ListPicker en définissant la propriété Elements sur une liste.

Pour cette application, vous souhaitez définir la propriété Elements pour ListPicker sur liste de destinations que vous venez de créer. Comme cela ne doit être réglé qu'une seule fois, vous

définissez ce comportement dans l' événement Screen1.Initialize . Vous aurez besoin des blocs qui sont répertoriés dans le tableau 6-4.

Tableau 6-4. Blocs pour lancer le ListPicker au démarrage de l'application

Type de bloc	Tiroir	But
Écran1.Initialiser	Écran1	Cet événement est déclenché au démarrage de l'application.
définir ListPicker1.Elements sur ListPicker1		Définissez cette propriété sur la liste que vous souhaitez afficher.
obtenir des destinations mondiales	Faire glisser depuis le bloc d'initialisation des variables	La liste des destinations.

Comment fonctionnent les blocs

Screen1.Initialize est déclenché au démarrage de l'application. La figure 6-3 montre que le gestionnaire d'événements définit la propriété Elements de ListPicker de manière à ce que les trois destinations apparaissent.



Figure 6-4. Initialisez le ListPicker avec les trois choix au lancement de l'application



Testez vos applications Cliquez sur Connecter et configurez des tests en direct avec votre appareil ou émulateur. Ensuite, cliquez sur le bouton intitulé « Choisir la destination Paris ». Le sélecteur de liste devrait apparaître avec les trois éléments. À ce stade, rien ne devrait se produire lorsque vous choisissez un élément.

OUVERTURE DE CARTES AVEC UNE URL DE RECHERCHE

Ensuite, vous programmerez l'application de sorte que lorsque l'utilisateur choisit l'une des destinations, ActivityStarter lance Google Maps et recherche l'emplacement sélectionné.

Tout d'abord, considérons l'URL <http://maps.google.com?q=Paris>. Lorsque vous tapez cette URL dans la barre d'adresse d'un navigateur, elle affiche un plan de Paris. Le "?" est commun à de nombreuses URL ; cela signifie qu'un paramètre arrive. Un paramètre correspond aux informations dont le site Web a besoin pour traiter la demande.

Dans ce cas, le nom du paramètre est « q », abréviation de « requête », et sa valeur est « Paris ». Il indique à Google Maps quelle carte afficher.

Dans cette application, vous allez créer une URL de manière dynamique, en ajoutant la valeur du paramètre en fonction de l'emplacement choisi par l'utilisateur. De cette façon, vous pouvez afficher différentes cartes en fonction des choix de l'utilisateur.

Lorsque l'utilisateur choisit un élément du composant ListPicker , le L'événement ListPicker.AfterPicking est déclenché. Dans le gestionnaire d'événements pour AfterPicking, vous devez définir le DataUri du composant ActivityStarter afin qu'il sache quelle carte ouvrir, puis vous devez lancer Google Maps en utilisant ActivityStarter.StartActivity. Les blocs pour cette fonctionnalité sont répertoriés dans Tableau 6-5.

Tableau 6-5. Blocs pour lancer Google Maps avec Activity Starter

Type de bloc	Tiroir	But
ListPicker1.AfterPicking	ListePicker1	Cet événement est déclenché lorsque l'utilisateur choisit parmi ListPicker.
définir ActivityStarter1.DataUri sur ActivityStarter1		Le DataUri indique à Maps sur quelle carte ouvrir lancement.
rejoindre	Texte	Construisez le DataUri à partir de deux morceaux de texte.
texte (« http://maps.google.com?q= »)	Texte	La première partie du DataUri attendu par Maps.
ListPicker1.Sélection	ListePicker1	L'élément choisi par l'utilisateur.
ActivityStarter1.StartActivity ActivityStarter1 Lancement des cartes.		

Comment fonctionnent les blocs

Lorsque l'utilisateur choisit dans ListPicker, l'élément choisi est stocké dans ListPicker.Selection et l'événement AfterPicking sont déclenchés. Comme représenté sur la Figure 6-4, la propriété DataUri est définie sur un objet texte qui combine « http://maps.google.com/?q » avec l'élément choisi. Ainsi, si l'utilisateur choisit le premier élément, « Visite Eifel », le DataUri serait défini sur « http://maps.google.com/?q= Tour Eiffel ».



Figure 6-5. Définition du DataURI pour lancer la carte sélectionnée

Parce que vous avez déjà défini les autres propriétés de ActivityStarter pour qu'il soit ouvrir Maps, le bloc ActivityStarter1.StartActivity lance le Maps app et appelle la recherche prescrite par DataUri.



Testez votre application Redémarrez l'application et cliquez à nouveau sur le bouton « Choisir Paris Destination ». Lorsque vous choisissez l'une des destinations, une carte de cette destination apparaît-elle ? Pouvez-vous revenir à votre application avec le bouton de retour de l'appareil ?

L'application complète : Map Tour avec Activity Starter

La figure 6-5 montre la configuration finale du bloc pour la version 1 de Paris Map Tour.

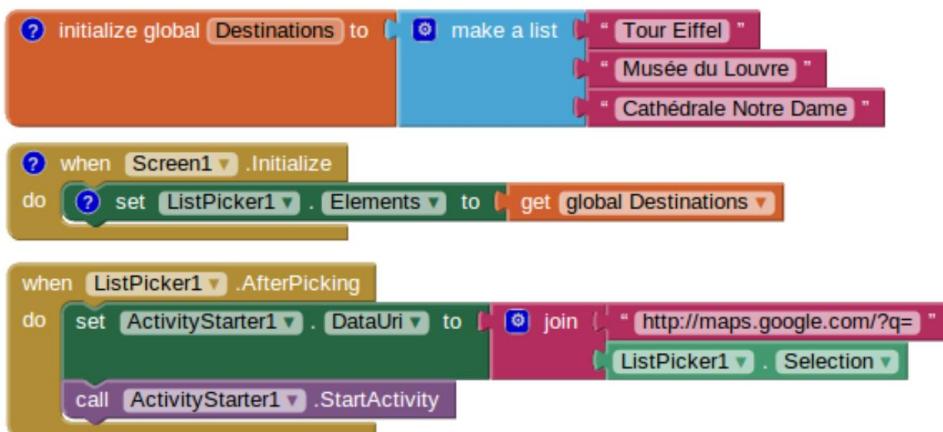


Figure 6-6. L'application Map Tour complète (version 1)

UNE VISITE VIRTUELLE AVEC LE WEB VIEWER

L'ActivityStarter est un composant important car il donne accès à toute autre application sur l'appareil. Mais il existe une autre façon de créer un guide touristique qui utilise un composant différent ; le WebViewer. WebViewer est un panneau que vous placez directement dans votre application et qui se comporte comme un navigateur. Vous pouvez ouvrir n'importe quelle page Web, y compris une carte Google Map, dans la visionneuse, et vous pouvez modifier par programme la page qui apparaît. Contrairement à un ActivityStarter, votre utilisateur ne quitte jamais votre application, vous n'avez donc pas besoin de compter sur lui pour appuyer sur le bouton Précédent pour revenir.

Dans cette deuxième version de l'application, vous utiliserez le WebViewer et vous pimenterez également l'application pour qu'elle ouvre des vues zoomées et des rues des monuments parisiens. Vous définirez une deuxième liste et utiliserez un schéma plus compliqué pour décider quelle carte afficher. Pour commencer, vous allez d'abord explorer Google Maps pour obtenir les URL de certaines cartes spécifiques. Vous utiliserez toujours les mêmes monuments parisiens pour les destinations, mais lorsque l'utilisateur en choisira un, vous utiliserez l'index (la position dans la liste) de son choix pour sélectionner et ouvrir une carte spécifique zoomée ou vue sur la rue. .

Avant de continuer, vous souhaiterez peut-être enregistrer votre projet (en utilisant Enregistrer sous) afin d'avoir une copie de la visite cartographique ActivityStarter que vous avez créée jusqu'à présent. De cette façon, si vous faites quelque chose qui provoque des problèmes dans votre application, vous pouvez toujours revenir à cette version de travail et réessayer.

Ajouter la visionneuse Web

Dans le concepteur, supprimez le composant ActivityStarter . Ensuite, depuis le tiroir Interface utilisateur, faites glisser un composant WebViewer et placez-le sous les autres composants.

Décochez la propriété Screen1.Scrollable pour que WebViewer affiche correctement les pages.

TROUVER L'URL DE CARTES SPÉCIFIQUES

L'étape suivante consiste à ouvrir Google Maps sur votre ordinateur pour trouver les cartes spécifiques que vous souhaitez lancer pour chaque destination :

1. Sur votre ordinateur, accédez à <http://maps.google.com>.
2. Recherchez un point de repère (par exemple, la Tour Eiffel).
3. Zoomez jusqu'au niveau souhaité.
4. Choisissez le type de vue souhaité (par exemple, Street View).
5. Saisissez l'URL. Dans la version classique de Maps, vous cliquez sur le bouton Lien à côté de en haut à droite de la fenêtre Cartes et copiez l'URL de la carte. Dans la nouvelle version de Google Maps, vous pouvez simplement récupérer l'URL dans la barre d'adresse.

Utilisez ce schéma pour créer des cartes sympas des monuments de Paris et extraire les URL. Le Tableau 6-6 fournit quelques exemples si vous préférez les utiliser (les URL ont été raccourcies avec le service bit.ly).

Tableau 6-6. URL de visites virtuelles pour Google Maps

Point de repère	URL des cartes
Tour Eiffel	http://bit.ly/1qiEy8B
Musée du Louvre	http://bit.ly/1qiEVQA
Cathédrale Notre Dame (vue sur la rue)	http://bit.ly/1qiF1YD

Pour afficher l'une de ces cartes dans un navigateur, collez les URL du tableau 6-6 dans la barre d'adresse.

DÉFINIR LA LISTE DES URLs

Vous aurez besoin d'une liste nommée URL, contenant une URL pour chacune des destinations. Créer cette liste comme le montre la figure 6-6 afin que les éléments correspondent aux éléments du liste de destinations (c'est-à-dire que la première URL doit correspondre à la première destination, l'Eifel La tour).

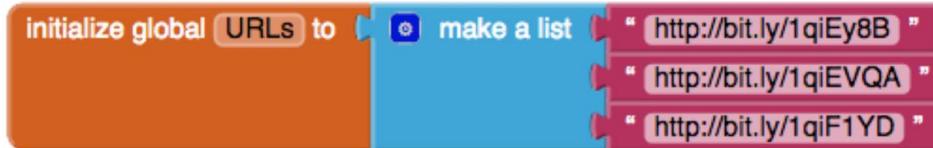


Figure 6-7. Copiez et collez les URL dans les blocs de texte de la liste des URL

MODIFICATION DU COMPORTEMENT DE LISTPICKER.AFTERPICKING

Dans la première version de cette application, le comportement ListPicker.AfterPicking définit le DataUri à une combinaison de « `<http://maps.google.com/?q=` » et de la destination vers laquelle l'utilisateur choisi dans la liste (par exemple, « Tour Eifel »). Dans cette deuxième version, l' AfterPicking le comportement doit être plus sophistiqué, car l'utilisateur choisit dans une liste (destinations), mais l'application choisit l' URL dans la liste des URL. Plus précisément, lorsque l'utilisateur choisit un élément dans ListPicker, vous devez connaître l'index de le choix afin que vous puissiez l'utiliser pour sélectionner l' URL correcte dans la liste. Nous vous expliquerons plus sur ce qu'est un index dans un instant, mais cela aide à configurer les blocs en premier pour mieux illustrer le concept. De nombreux blocs sont requis pour cette fonctionnalité, tous qui sont répertoriés dans le tableau 6-7.

Tableau 6-7. Blocs pour choisir un élément de liste en fonction de la sélection de l'utilisateur

Type de bloc	Tiroir	But
ListPicker1.AfterPicking ListPicker1		Cet événement est déclenché lorsque l'utilisateur choisit un élément.
ListPicker1.SelectionIndex ListPicker1		L'index (position) de l'élément choisi.
sélectionner un élément de la liste	Listes	Sélectionnez un élément dans la liste des URL.
obtenir des URL globales	Faites-le glisser depuis la variable initialisation	La liste des URL.
WebViewer.GoToURL	Visionneuse Web	Chargez l'URL dans la visionneuse pour afficher la carte.

Comment fonctionnent les blocs

Lorsque l'utilisateur choisit un élément dans ListPicker, l'événement AfterPicking est déclenché, comme le montre la figure 6-7. L'élément choisi, par exemple « Tour Eifel », se trouve dans ListPicker.Selection. Vous avez utilisé cette propriété dans la première version de cette application. Cependant, ListPicker possède également une propriété SelectionIndex, qui correspond à la position de la destination choisie dans la liste. Ainsi, si « Tour Eifel » est choisi, le SelectionIndex sera 1 ; si « Musée du Louvre » est choisi, ce sera 2 ; et si « Cathédrale Notre Dame de Paris » est choisie, ce sera 3.



Figure 6-8. Ouvrez l'URL sélectionnée dans le WebViewer

Vous utilisez ListPicker.SelectionIndex pour sélectionner un élément dans la liste des URL. Cela marche car les éléments des deux listes, destinations et URL, sont synchronisés : la première destination correspond à la première URL, la deuxième à la deuxième et la troisième à la troisième. Ainsi, même si l'utilisateur choisit un élément dans une liste, vous pouvez utiliser son choix (enfin, l'index de son choix) pour sélectionner la bonne URL à afficher.



Testez votre application Sur l'appareil, cliquez sur le bouton « Choisir Paris Destination ». La liste devrait apparaître avec les trois éléments. Choisissez l'un des éléments et voyez quelle carte apparaît.

L'application complète : Map Tour (Web Viewer)

La figure 6-8 montre la configuration finale du bloc pour cette deuxième version de Paris Map Tour.

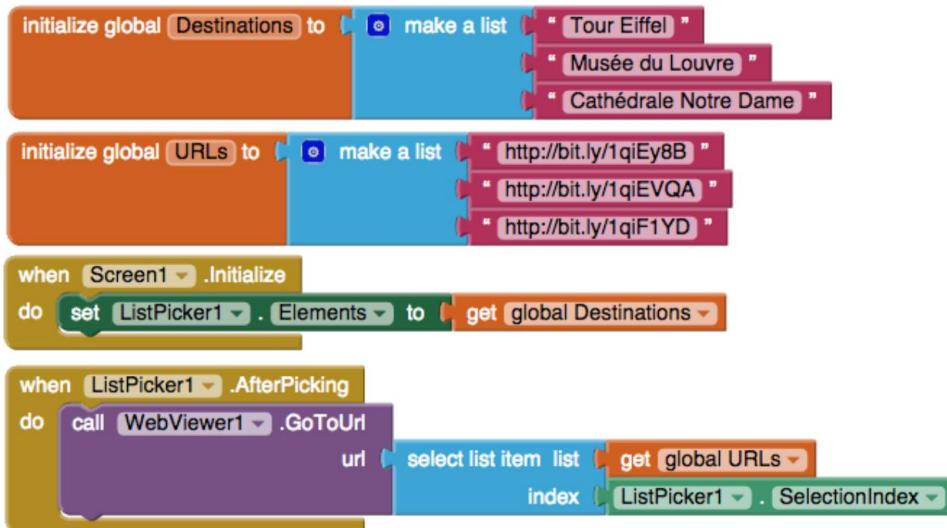


Figure 6-9. L'application Map Tour complète (version WebViewer)

Variantes

Voici quelques variantes suggérées à essayer :

- Créez une visite virtuelle de votre lieu de travail ou de votre école, ou pour vos prochaines vacances destination.
- Explorez ActivityStarter et utilisez-le pour envoyer un e-mail ou lancer une application telle que YouTube (voir <http://bit.ly/1qiFx8Z> pour aider).
- Difficile : créez une application de visite virtuelle personnalisable qui permet à un utilisateur de créer un guide pour un emplacement de son choix en saisissant le nom de chaque destination ainsi que l'URL d'une carte correspondante. Vous devrez stocker les données dans une base de données TinyWebDB et créer une application de visite virtuelle qui fonctionne avec les données saisies. Pour un exemple de création d'une base de données TinyWebDB , consultez le MakeQuiz/TakeQuiz application.

Résumé

Voici quelques-unes des idées que nous avons abordées dans ce chapitre :

- Vous pouvez utiliser des variables de liste pour conserver des données telles que des destinations cartographiques et des URL.
- Le composant ListPicker permet à l'utilisateur de choisir parmi une liste d'éléments. La propriété Elements de ListPicker contient la liste, la propriété Selection contient la

élément sélectionné, le SelectionIndex contient la position de l'élément sélectionné et l'événement AfterPicking est déclenché lorsque l'utilisateur choisit un élément dans la liste.

- Le composant ActivityStarter permet à votre application de lancer d'autres applications. Ce chapitre a démontré son utilisation avec l'application Google Maps, mais vous pouvez également lancer un navigateur ou toute autre application Android, même une autre que vous avez créée vous-même.
- Vous pouvez utiliser ListPicker.SelectionIndex pour obtenir la position d'un élément qu'un utilisateur choisit dans une liste. Vous pouvez ensuite utiliser cet index pour sélectionner des informations dans une liste différente (dont les éléments sont synchronisés avec la première liste). Pour plus d'informations sur les variables List et le composant ListPicker , voir le chapitre 19.

Android, où est ma voiture ?

Vous vous êtes garé le plus près possible du stade, mais lorsque le concert se termine, vous n'avez aucune idée de l'endroit où se trouve votre voiture. Vos amis sont également ignorants. Heureusement, vous n'avez pas perdu votre téléphone Android, qui n'oublie jamais rien, et vous vous souvenez que vous avez la nouvelle application à la mode, Android, Où est ma voiture ? Avec cette application, vous cliquez sur un bouton lorsque vous garez votre voiture, et Android utilise son capteur de localisation pour enregistrer les coordonnées GPS et l'adresse de la voiture. Plus tard, lorsque vous rouvrirez l'application, elle vous indiquera l'itinéraire depuis l'endroit où vous vous trouvez actuellement jusqu'à l'emplacement enregistré : problème résolu !



Ce que vous apprendrez

Cette application couvre les concepts suivants :

- Détermination de l'emplacement de l'appareil Android à l'aide de LocationSensor composant.
- Enregistrement persistant des données dans une base de données directement sur l'appareil à l'aide de TinyDB.
- Utilisation du composant WebViewer pour ouvrir Google Maps depuis votre application et afficher directions d'un endroit à un autre.

Commencer

Connectez-vous au site Web App Inventor et démarrez un nouveau projet. Étant donné que les noms de projets ne peuvent pas contenir d'espaces, nommez-le « AndroidWhere ». Définissez le titre de l'écran sur "Android, où est ma voiture ?" Connectez votre appareil ou émulateur pour des tests en direct.

Conception des composants

L'interface utilisateur pour Android, Où est ma voiture ? se compose d'étiquettes pour afficher vos emplacements actuels et mémorisés, ainsi que de boutons pour enregistrer un emplacement et afficher les directions vers celui-ci. Vous aurez besoin de quelques étiquettes qui affichent uniquement du texte statique ; par exemple, GPSLabel fournira le texte « GPS : » qui apparaît dans l'interface utilisateur. D'autres étiquettes, telles que CurrentLatLabel, afficheront les données du capteur de localisation. Pour ces étiquettes, vous fournirez une valeur par défaut (0,0), qui changera à mesure que le GPS acquiert des informations de localisation.

Vous aurez également besoin de trois composants non visibles : un LocationSensor pour obtenir le l'emplacement actuel, un TinyDB pour stocker les emplacements de manière persistante et un WebView pour afficher les directions Google Maps entre les emplacements actuels et stockés.

Vous pouvez créer les composants à partir de l'instantané du Component Designer de la figure 7-1.

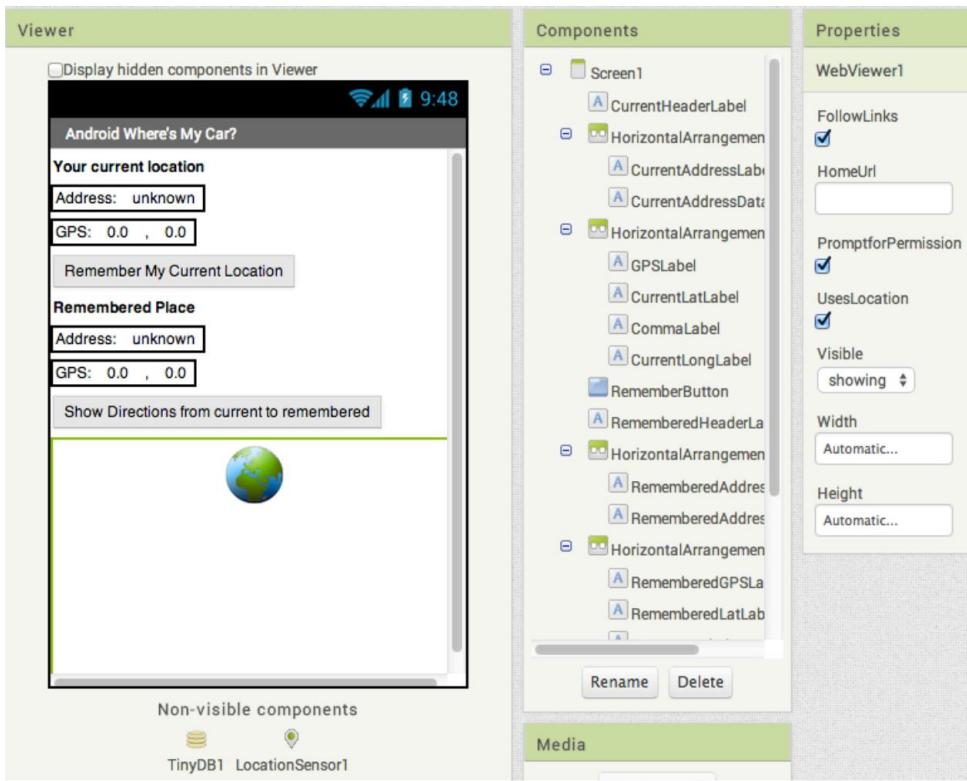


Figure 7-1. L'Android, où est ma voiture ? application dans le concepteur de composants

Vous aurez besoin des composants du tableau 7-1.

Tableau 7-1. Tous les composants de l'application

Type de composant	Palette groupe	Comment tu l'appelleras	But
Étiquette	Interface utilisateur	CurrentHeaderLabel	Afficher l'en-tête « Votre localisation actuelle ».
Disposition de l'arrangement horizontal		Disposition horizontale1	Organisez les informations d'adresse.
Étiquette	Interface utilisateur	CurrentAddressLabel	Affichez le texte « Adresse : ».
Étiquette	Interface utilisateur	CurrentAddressDataLabel	Afficher les données dynamiques : la adresse actuelle.
Disposition de l'arrangement horizontal		Disposition horizontale2	Organisez les informations GPS.
Étiquette	Interface utilisateur	Étiquette GPS	Affichez le texte « GPS : ».
Étiquette	Interface utilisateur	CurrentLatLabel	Afficher les données dynamiques : la latitude actuelle.
Étiquette	Interface utilisateur	VirguleLabel	Afficher ",".

Type de composant	Palette groupe	Comment tu l'appelleras	But
Étiquette	Interface utilisateur CurrentLongLabel		Afficher les données dynamiques : la longitude actuelle.
Bouton	Interface utilisateur RememberButton		Cliquez pour enregistrer le courant emplacement.
Étiquette	Interface utilisateur Arrangement horizontal2		Organiser mémorisé informations sur l'adresse.
Étiquette	Interface utilisateur RememberedAddressLabel		Afficher le texte « Lieu mémorisé ».
Étiquette	Interface utilisateur RememberedAddressDataLabel		Afficher les données dynamiques : la adresse mémorisée.
Étiquette	Interface utilisateur mémoriséeGPSLabel		Afficher le texte « GPS ».
Étiquette	Interface utilisateur RememberedLatLabel		Afficher les données dynamiques : je me souviens de la latitude.
Étiquette	Interface utilisateur Comma2Label		Afficher ",".
Étiquette	Interface utilisateur RememberedLongLabel		Afficher les données dynamiques : le mémorisé la longitude.
Bouton	Bouton Directions de l'interface utilisateur		Cliquez pour afficher la carte.
Capteur de localisation	Capteurs	LocalisationCapteur1	Détecter les informations GPS.
MinusculeDB	Stockage	MinusculeDB1	Conservez les souvenirs emplacement de manière persistante.
Visionneuse Web	Interface utilisateur WebViewer1		Afficher les directions.

Définissez les propriétés des composants de la manière suivante :

- Définissez la propriété Text pour les étiquettes avec du texte fixe comme spécifié dans le Tableau 7-1.
- Définissez la propriété Text des étiquettes pour les données GPS dynamiques sur « 0.0 ».
- Définissez la propriété Text des étiquettes pour les adresses dynamiques sur « inconnu ».
- DÉCOchez la propriété Enabled de RememberButton et de DirectionsButton.
- DÉCOchez la propriété Screen.Scrollable pour que le WebViewer tienne sur le écran.

Ajout de comportements aux composants

Vous aurez besoin des comportements suivants pour cette application :

- Lorsque le LocationSensor obtient une lecture, placez les données de localisation actuelles dans les étiquettes appropriées de l'interface utilisateur. Cela permettra à l'utilisateur de connaître le capteur a lu un emplacement et est prêt à s'en souvenir.
- Lorsque l'utilisateur clique sur le RememberButton, copiez les données de localisation actuelles dans les étiquettes de l'emplacement mémorisé. Vous devrez également stocker les données de localisation mémorisées afin qu'elles soient là si l'utilisateur ferme et relance l'application.
- Lorsque l'utilisateur clique sur le bouton Directions, lancez Google Maps dans le WebViewer afin qu'il affiche les directions vers l'emplacement mémorisé.
- Lorsque l'application est relancée, chargez l'emplacement mémorisé à partir de la base de données dans l'application.

AFFICHAGE DE L'EMPLACEMENT ACTUEL

L'événement LocationSensor.LocationChanged ne se produit pas seulement lorsque le périphérique l'emplacement change, mais aussi lorsque le capteur obtient pour la première fois une lecture. Parfois ce premier la lecture prendra quelques secondes, et parfois vous n'obtiendrez pas de lecture du tout si les lignes de visibilité vers les satellites GPS sont bloquées (et en fonction des paramètres de l'appareil). Pour plus d'informations sur le GPS et LocationSensor, voir le chapitre 23.

Lorsque vous obtenez une lecture de localisation, l'application doit placer les données dans les étiquettes appropriées. Le tableau 7-2 répertorie tous les blocs dont vous aurez besoin pour ce faire.

Tableau 7-2. Blocs permettant d'obtenir une lecture de localisation et de l'afficher dans l'interface utilisateur de l'application

Type de bloc	Tiroir	But
LocationSensor1.LocationChanged LocationSensor1		C'est le gestionnaire d'événements qui est déclenché lorsque le téléphone reçoit un nouveau GPS en lisant.
définissez CurrentAddressDataLabel.Text sur	CurrentAddressDataLabel	Placez les nouvelles données dans l'étiquette du adresse actuelle.
LocationSensor1.CurrentAddress LocationSensor1		Cette propriété vous donne une adresse postale.
définir CurrentLatLabel.Text sur	ÉtiquetteLatActuelle	Placez la latitude dans le champ approprié étiquette.
obtenir la latitude	Sortez de Événement LocationChanged	Branchez-le sur CurrentLatLabel.Text.
définir CurrentLongLabel.Text sur	CurrentLongLabel	Placez la longitude dans le champ approprié étiquette.
valeur longitude	Sortez de Événement LocationChanged	Branchez-le sur CurrentLongLabel.Text.

Type de bloc	Tiroir	But
définir RememberButton.Enabled sur	Bouton Mémoriser	Rappelez-vous la lecture pour le courant emplacement.
vrai	Logique	Branchez-le sur l'ensemble RememberButton.Enabled.

Comment fonctionnent les blocs

La figure 7-2 montre que la latitude et la longitude sont des paramètres du

Événement LocationChanged . Vous pouvez récupérer les références aux paramètres d'événement en passant la souris au dessus d'eux. CurrentAddress n'est pas un argument ; c'est plutôt une propriété du LocationSensor , vous le récupérez donc dans le tiroir de LocationSensor . Le capteur de localisation effectue un travail supplémentaire pour vous en appelant Google Maps pour obtenir une adresse postale correspondant à la localisation GPS.

Ce gestionnaire d'événements active également le RememberButton . Nous l'avons initialisé comme désactivé (non coché) dans le Concepteur de composants car l'utilisateur n'a rien à faire. rappelez-vous jusqu'à ce que le capteur obtienne une lecture, nous allons donc maintenant programmer ce comportement.

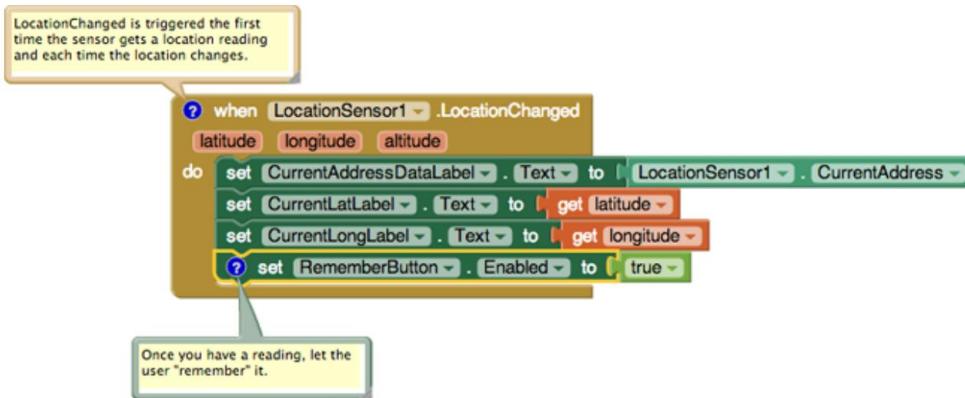


Figure 7-2. Utilisation du LocationSensor pour lire l'emplacement actuel



Testez votre application Vous voudrez probablement vous promener pour tester cela application. Vous devrez donc créer l'application et l'installer sur votre téléphone en sélectionnant Build -> App (fournissez le code QR pour .apk). Lorsque vous exécutez l'application, vous devriez voir apparaître des données GPS et le activé. **Bouton Mémoriser** pas de lecture, vérifiez vos paramètres Android pour la localisation et la sécurité et essayez aller dehors. Pour plus d'informations, voir le chapitre 23.

ENREGISTREMENT DE L'EMPLACEMENT ACTUEL

Lorsque l'utilisateur clique sur le RememberButton, les données de localisation les plus récentes doivent être placé dans les étiquettes pour afficher les données mémorisées. Le tableau 7-3 vous montre lequel blocs dont vous aurez besoin pour cette fonctionnalité.

Tableau 7-3. Blocs pour enregistrer et afficher l'emplacement actuel

Type de bloc	Tiroir	But
RememberButton.Click	Bouton Mémoriser	Déclenché lorsque l'utilisateur clique "Souviens-toi."
définir RememberedAddressDataLabel.Text à	RememberedAddressDataLabel	Placez les données d'adresse du capteur dans l'étiquette de l'adresse mémorisée.
LocationSensor1.CurrentAddress LocationSensor1		Cette propriété vous donne une rue adresse.
définissez RememberedLatLabel.Text sur	ÉtiquetteLat mémorisée	Placez la latitude détectée dans le étiquette « mémorisée ».
LocationSensor.Latitude	LocalisationCapteur1	Branchez-vous sur l'ensemble RememberedLatLabel.Text à.
définir RememberedLongLabel.Text sur RememberedLongLabel		Placez la longitude détectée dans le étiquette « mémorisée ».
LocationSensor.Longitude	LocalisationCapteur1	Branchez-le sur l'ensemble RememberedLongLabel.Text.
définir DirectionsButton.Enabled sur DirectionsButton		Cartographiez le lieu mémorisé.
vrai	Logique	Branchez-le sur l'ensemble DirectionsButton.Enabled.

Comment fonctionnent les blocs

Lorsque l'utilisateur clique sur le bouton Remember, les lectures actuelles du capteur de localisation sont insérés dans les étiquettes « mémorisées », comme le montre la figure 7-3.

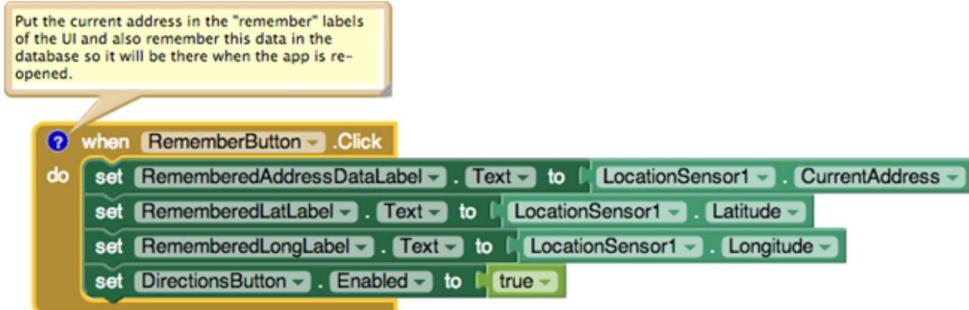


Figure 7-3. Placer les informations de localisation actuelle dans les étiquettes « mémorisées »

Vous remarquerez également que le DirectionsButton est activé. Cela pourrait devenir délicat, car si l'utilisateur clique immédiatement sur le bouton Directions , l'emplacement mémorisé sera le même que l'emplacement actuel, donc la carte qui apparaît ne fournira pas grand-chose en termes de directions. Mais ce n'est pas quelque chose que quiconque est susceptible de faire ; après que l'utilisateur se déplace (par exemple, se rend au concert), l'emplacement actuel et l'emplacement mémorisé divergent.



Testez votre application Téléchargez la nouvelle version de l'application sur votre téléphone et testez à nouveau. Lorsque vous cliquez sur les données des paramètres actuels, les données sont-elles copiées dans les paramètres mémorisés ?

AFFICHAGE DES DIRECTIONS VERS L'EMPLACEMENT Mémorisé

Lorsque l'utilisateur clique sur le bouton Directions, vous souhaitez que l'application ouvre Google Maps, puis affiche les directions depuis l'emplacement actuel de l'utilisateur jusqu'à l'emplacement mémorisé (par exemple, l'endroit où la voiture est garée).

Le composant WebViewer peut afficher n'importe quelle page Web, y compris Google Maps.

Vous allez appeler WebViewer.GoToURL pour ouvrir la carte, mais vous souhaitez ouvrir une URL qui affiche les directions depuis l'emplacement actuel vers l'emplacement mémorisé.

Une façon d'afficher des itinéraires dans Maps consiste à utiliser une URL au format suivant :

<http://maps.google.com/maps?>

saddr=37.82557,-122.47898&daddr=37.81079,-122.47710 Tapez cette URL

dans un navigateur : pouvez-vous dire vers quel monument célèbre elle vous dirige à travers ?

Pour cette application, vous devez créer l'URL et définir son adresse source (saddr) et paramètres d'adresse de destination (daddr) de manière dynamique (en blocs). Vous avez déjà rassemblé du texte dans des chapitres précédents en utilisant join ; nous le ferons ici également, en nous connectant

les données GPS pour les emplacements mémorisés et actuels. Vous mettrez l'URL que vous créez dans l'emplacement de paramètre de WebViewer.GoToURL. Le tableau 7-4 répertorie tous les blocs dont vous aurez besoin pour ça.

Tableau 7-4. Blocs pour enregistrer et afficher l'emplacement actuel

Type de bloc	Tiroir	But
DirectionsButton.Click	Bouton Directions	Déclenché lorsque l'utilisateur clique sur « Itinéraire ».
WebViewer1.GoToURL	Visionneuse Web1	Définissez l'URL de la carte que vous souhaitez apporter en haut.
rejoindre	Texte	Créez une URL à partir de plusieurs parties.
texte (« http://maps.google.com/maps? saddr="" »)	Texte	La partie fixe de l'URL, l'adresse source.
CurrentLatLabel.Text	ÉtiquetteLatActuelle	La latitude actuelle.
texte (",")	Texte	Mettez une virgule entre la latitude et valeurs de longitude.
CurrentLongLabel.Text	CurrentLongLabel	La longitude actuelle.
texte (« &daddr = »)	Texte	Le deuxième paramètre de l'URL, le adresse de destination.
RememberedLatLabel.Text	RememberedLatLabel	La latitude mémorisée.
texte (",")	Texte	Mettez une virgule entre les valeurs de latitude et la longitude.
RememberedLongLabel.Text	RememberedLongLabel	La longitude mémorisée.

Comment fonctionnent les blocs

Lorsque l'utilisateur clique sur le bouton Directions, le gestionnaire d'événements crée une URL pour une carte. et appelle WebViewer.GoToURL pour ouvrir la carte, comme le montre la figure 7-4. rejoindre est utilisé pour créez l'URL de la carte.

L'URL résultante est constituée du domaine Google Maps (<http://maps.google.com/maps>) ainsi que deux paramètres URL, saddr et daddr, qui spécifient la source et emplacements de destination pour les directions. Pour cette application, le saddr est défini sur la latitude et la longitude de l'emplacement actuel, et le papa est réglé sur la latitude et longitude de l'emplacement mémorisé pour la voiture.

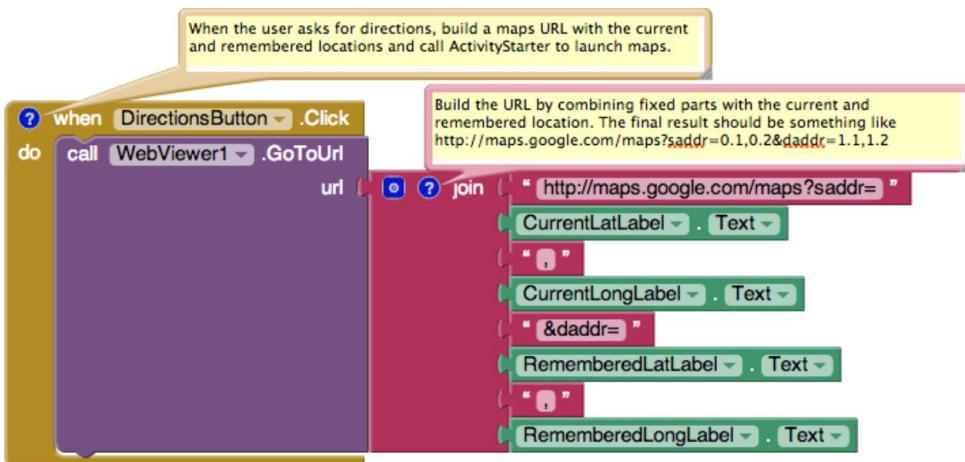


Figure 7-4. Construire l'URL à utiliser pour ouvrir la carte dans WebViewer



Testez votre application Téléchargez la nouvelle version de l'application sur votre téléphone et testez à nouveau. Lorsqu'une lecture arrive, Bouton Mémoriser cliquez sur puis promenez-vous. Lorsque vous Bouton Directions , cliquez sur la carte, la carte vous indique-t-elle comment revenir sur vos pas ? Après avoir regardé la carte, cliquez plusieurs fois sur le bouton de retour. Revenez-vous à votre application ?

STOCKAGE PERMANENT DE L'EMPLACEMENT Mémorisé

Vous disposez désormais d'une application entièrement fonctionnelle qui mémorise un emplacement de départ et dessine une carte vers cet emplacement, quel que soit l'emplacement actuel de l'utilisateur. Cependant, si l'utilisateur « se souvient » d'un emplacement et ferme ensuite l'application, les données mémorisées ne seront pas disponibles à la réouverture de l'application. Ce que vous voulez vraiment, c'est que l'utilisateur puisse enregistrer l'emplacement de la voiture, fermer l'application et se rendre à un événement, puis relancer l'application plus tard pour obtenir un itinéraire vers l'emplacement enregistré.

Si vous repensez déjà à l'application No Texting While Driving (Chapitre 4), vous êtes sur la bonne voie. Vous devez stocker les données de manière persistante dans une base de données à l'aide de TinyDB. Vous utiliserez un schéma similaire à celui que vous avez utilisé dans cette application :

1. Lorsque l'utilisateur clique sur le RememberButton, stockez les données de localisation dans la base de données.
2. Lorsque l'application démarre, chargez les données de localisation de la base de données dans une variable ou une propriété.

Vous allez commencer par modifier le gestionnaire d'événements RememberButton.Click afin qu'il stocke les données mémorisées. Pour stocker la latitude, la longitude et l'adresse, vous aurez besoin de trois appels à TinyDB.StoreValue. Le tableau 7-5 répertorie les blocs supplémentaires dont vous aurez besoin.

Tableau 7-5. Blocs pour enregistrer et afficher l'emplacement actuel

Type de bloc	Tiroir	But
TinyDB1.StoreValue (3)	MinusculeDB	Stockez les données dans la base de données de l'appareil.
texte (« adresse »)	Texte	Branchez-le dans la prise « tag » de TinyDB1.StoreValue.
LocationSensor1.CurrentAddress LocationSensor1		L'adresse à stocker de manière persistante ; branchez-le dans le Socket « valeur » de TinyDB1.StoreValue.
texte (« lat »)	Texte	Branchez-le sur la prise « tag » du deuxième TinyDB1.StoreValue.
LocationSensor1.CurrentLatitude LocationSensor1		La latitude de stocker de manière persistante ; branchez-le dans le Prise « valeur » de la seconde TinyDB1.StoreValue.
texte (« long »)	Texte	Branchez-le sur la prise « tag » du troisième TinyDB1.StoreValue.
LocationSensor1.CurrentLongitude LocationSensor1		La longitude à stocker de manière persistante ; branchez-le dans le prise « valeur » du troisième TinyDB1.StoreValue.

Comment fonctionnent les blocs

Comme le montre la figure 7-5, TinyDB1.StoreValue copie les données de localisation du Propriétés LocationSensor dans la base de données. Comme vous vous en souvenez peut-être de No Texting Pendant la conduite, la fonction StoreValue a deux arguments, la balise et la valeur. Le La balise identifie les données que vous souhaitez stocker et la valeur correspond aux données réelles que vous que vous souhaitez enregistrer : dans ce cas, les données LocationSensor .

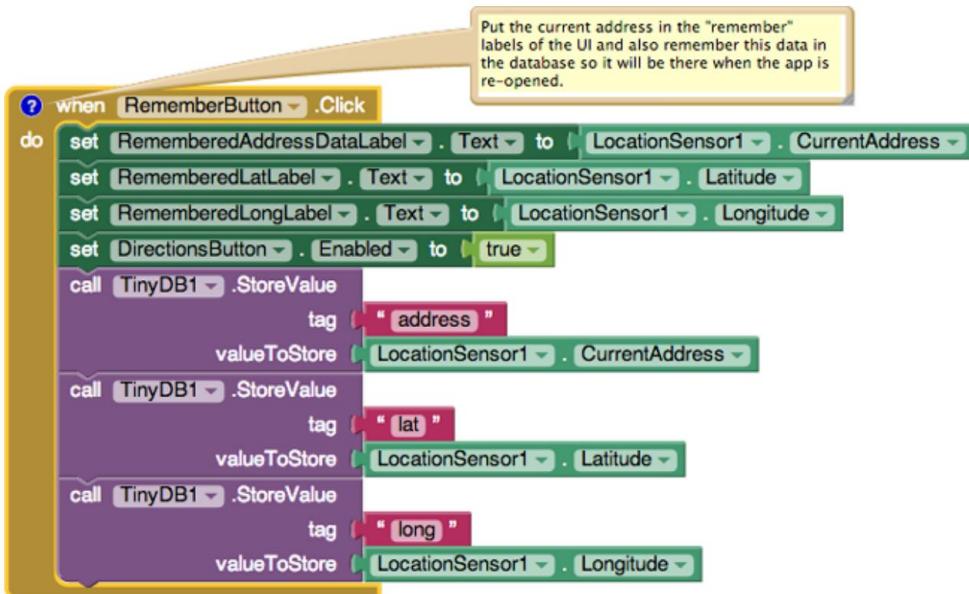


Figure 7-5. Stockage des données de localisation mémorisées dans une base de données

RÉCUPÉRATION DE L'EMPLACEMENT MÉMÉNÉRÉ AU LANCEMENT DE L'APPLICATION

Vous stockez les données dans une base de données afin de pouvoir les rappeler ultérieurement. Dans cette application, si un utilisateur stocke un emplacement puis ferme l'application, vous souhaitez récupérer ces informations de la base de données et les afficher lorsque l'utilisateur relance l'application.

Comme indiqué dans les chapitres précédents, l'événement Screen.Initialize est déclenché au lancement de votre application. Récupérer des données d'une base de données est une chose très courante au démarrage, et c'est exactement ce que nous voulons faire pour cette application.

Vous utiliserez la fonction TinyDB.GetValue pour récupérer les données GPS stockées. Étant donné que vous devez récupérer l'adresse, la latitude et la longitude stockées, vous aurez besoin de trois appels à GetValue. Comme pour No Texting While Driving, vous devrez vérifier si des données sont effectivement disponibles (si c'est la première fois que vous lancez votre application, TinyDB.GetValue renverra un texte vide).

En guise de défi, voyez si vous pouvez créer ces blocs puis comparez votre création aux blocs illustrés à la figure 7-6.

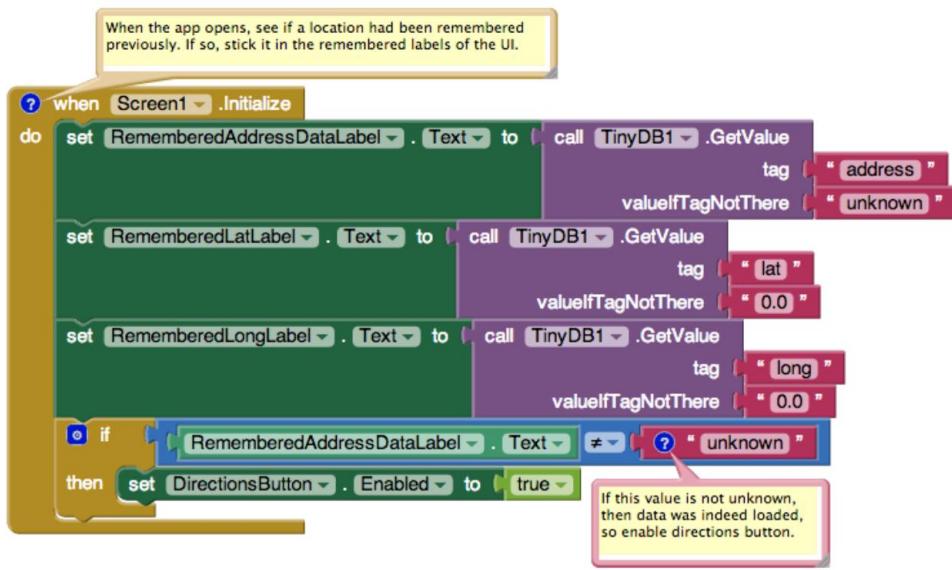


Figure 7-6. Lorsque l'application démarre, chargez-la à l'emplacement mémorisé depuis la base de données

Comment fonctionnent les blocs

Pour comprendre ces blocages, envisagez deux cas d'utilisation : un utilisateur ouvrant l'application pour la première fois et l'utilisateur l'ouvrant plus tard, après avoir préalablement enregistré les données de localisation. La première fois que l'utilisateur ouvre l'application, il n'y aura aucune donnée de localisation à charger dans la base de données.

Lors de lancements successifs, si des données sont stockées, vous souhaitez charger les données de localisation précédemment stockées depuis la base de données.

Les blocs appellent TinyDB1.GetValue trois fois, une fois pour chacun des champs de données que vous avez stockés précédemment : « adresse », « lat » et « long ». Le paramètre valueIfTagNotThere est défini sur une valeur par défaut pour chacun de sorte que s'il n'y a pas encore de données dans la base de données, les étiquettes seront définies sur les valeurs par défaut (les mêmes que celles définies dans le concepteur).

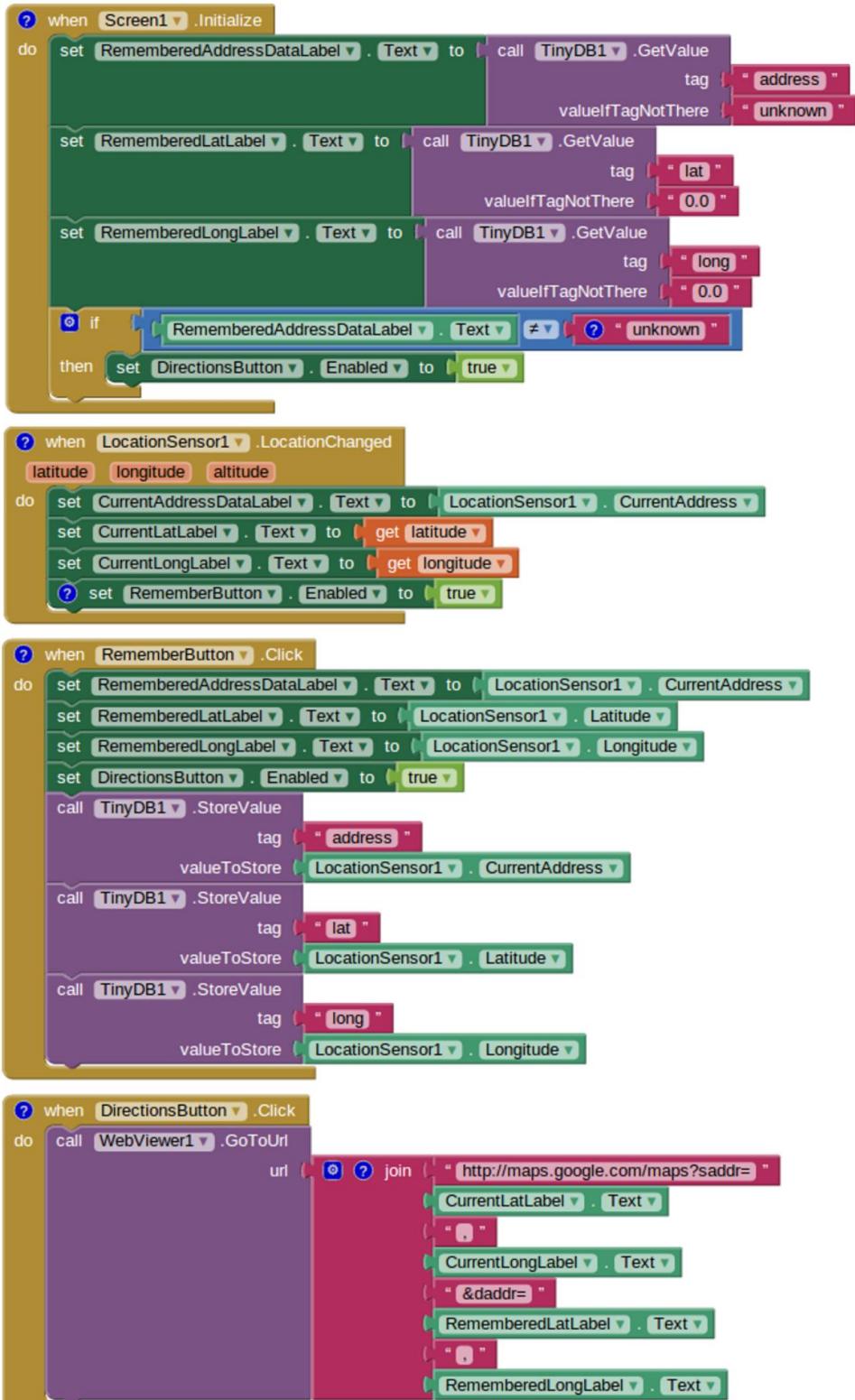
Le bloc if est utilisé pour déterminer si le DirectionsButton doit être activé. Cela devrait être le cas si les données ont effectivement été extraites de la base de données. Le test utilisé consiste à comparer le RememberedAddressDataLabel à sa valeur par défaut, inconnue. S'il n'est pas inconnu, il doit avoir été remplacé par une adresse mémorisée.



Testez votre application Téléchargez la nouvelle version de l'application sur votre téléphone et testez à nouveau. Cliquez sur et assurez-vous que les lectures sont enregistrées. Fermez ensuite l'application et rouvrez-la. Les données mémorisées apparaissent-elles ?

L'application complète : Android, où est ma voiture ?

La figure 7-7 montre les blocs finaux pour la version complète d'Android, Où est ma voiture ? application.



Variantes

Voici quelques variantes que vous pouvez expérimenter :

- Créez Android, Où est tout le monde ?, une application qui permet à un groupe de personnes de se localiser mutuellement. Que vous fassiez une randonnée dans les bois ou que vous soyez séparés au parc, cette application pourrait vous aider à gagner du temps et peut-être même des vies. Les données de cette application sont partagées, vous devrez donc utiliser une base de données Web et le composant TinyWebDB au lieu de TinyDB. Voir le chapitre 22 pour plus d'informations.
- Créez une application Breadcrumb qui suit vos déplacements en enregistrant chaque changement d'emplacement dans une liste. Vous ne devez enregistrer un nouveau fil d'Ariane que si l'emplacement a changé d'un certain montant ou si un certain temps s'est écoulé, car même un léger mouvement peut générer une nouvelle lecture d'emplacement.

Vous devrez stocker les emplacements enregistrés dans une liste. Voir le chapitre 19 pour obtenir de l'aide.

Résumé

Voici quelques-unes des idées que nous avons abordées dans ce tutoriel :

- Le composant LocationSensor peut signaler la latitude, la longitude et l'adresse actuelle de l'utilisateur. Son événement LocationChanged est déclenché lorsque le capteur obtient sa première lecture et lorsque la lecture change (l'appareil a bougé). Pour plus d'informations sur le LocationSensor, voir le chapitre 23.
- Le composant WebViewer affiche n'importe quelle page Web, y compris Google Maps. Si vous souhaitez afficher les directions entre les coordonnées GPS, l'URL sera au format suivant, mais vous devez remplacer les exemples de données affichés ici par les coordonnées GPS réelles : <http://maps.google.com/maps/?saddr=0.1&daddr=0.2,0.2>
- Vous utilisez join pour rassembler (concaténer) des éléments de texte séparés en un seul objet texte. Vous pouvez l'utiliser pour concaténer des données dynamiques avec du texte statique. Avec l'URL Maps, les coordonnées GPS sont les données dynamiques.
- Grâce à TinyDB, vous pouvez stocker des données de manière persistante dans la base de données du téléphone. Alors que les données d'une variable ou d'une propriété sont perdues à la fermeture d'une application, les données stockées dans la base de données peuvent être chargées à chaque ouverture de l'application. Pour plus d'informations sur TinyDB et les bases de données, voir le chapitre 22.

Quiz des présidents

Le Presidents Quiz est un jeu-questionnaire sur les anciens dirigeants des États-Unis. Bien que ce quiz concerne les présidents, vous pouvez l'utiliser comme modèle pour créer des quiz ou des guides d'étude sur n'importe quel sujet.

Dans les chapitres précédents, vous avez été présenté à certains concepts fondamentaux de programmation.

Maintenant, vous êtes prêt pour quelque chose de plus difficile.

Vous constaterez que ce chapitre nécessite un saut conceptuel en termes de compétences en programmation et de pensée abstraite. En particulier, vous utiliserez deux variables de liste pour stocker les données (dans ce cas, les questions et réponses du quiz) et vous utiliserez une variable d'index pour suivre où se trouve l'utilisateur dans le quiz. Lorsque vous aurez terminé, vous disposerez des connaissances nécessaires pour créer des applications de quiz et de nombreuses autres applications nécessitant un traitement de liste.

Ce chapitre suppose que vous connaissez les bases d'App Inventor : utiliser le Concepteur de composants pour créer l'interface utilisateur et utilisation de l'éditeur de blocs pour spécifier les gestionnaires d'événements et programmer le comportement des composants. Si vous n'êtes pas familier avec ces principes fondamentaux, assurez-vous de revoir les chapitres précédents avant de continuer.

Vous allez concevoir le quiz de manière à ce que l'utilisateur passe de question en question en cliquant sur un bouton Suivant et reçoit des commentaires sur les réponses.

Figure 8-1.



Ce que vous apprendrez

Cette application, illustrée à la figure 8-1, couvre :

- Définir des variables de liste pour stocker les questions et réponses dans des listes.
- Séquençage d'une liste à l'aide d'un index ; chaque fois que l'utilisateur clique sur Suivant, vous afficherez la question suivante.
- Utiliser des comportements conditionnels (if) : effectuer certaines opérations uniquement dans des conditions spécifiques. Vous utiliserez un bloc if pour gérer le comportement de l'application lorsque l'utilisateur atteint la fin du quiz.
- Changer d'image pour afficher une image différente pour chaque question du quiz.



Figure 8-2. Le Quiz des Présidents en action

Commencer

Accédez au site Web App Inventor et démarrez un nouveau projet. Donnez au projet le nom « PresidentsQuiz » et définissez le titre de l'écran sur « Presidents Quiz ». Connectez votre appareil ou émulateur pour des tests en direct.

Vous devrez télécharger les images suivantes pour le quiz à partir du

Site Web appinventor.org sur votre ordinateur :

- <http://appinventor.org/bookFiles/PresidentsQuiz/roosChurch.gif>
- <http://appinventor.org/bookFiles/PresidentsQuiz/nixon.gif>
- <http://appinventor.org/bookFiles/PresidentsQuiz/carterChina.gif>
- <http://appinventor.org/bookFiles/PresidentsQuiz/atomic.gif>

Vous chargerez ces images dans votre projet dans la section suivante.

Conception des composants

L'application Presidents Quiz dispose d'une interface simple pour afficher les questions et permettre à l'utilisateur d'y répondre. Vous pouvez créer les composants à partir de l'instantané du Concepteur de composants illustré à la figure 8-2.

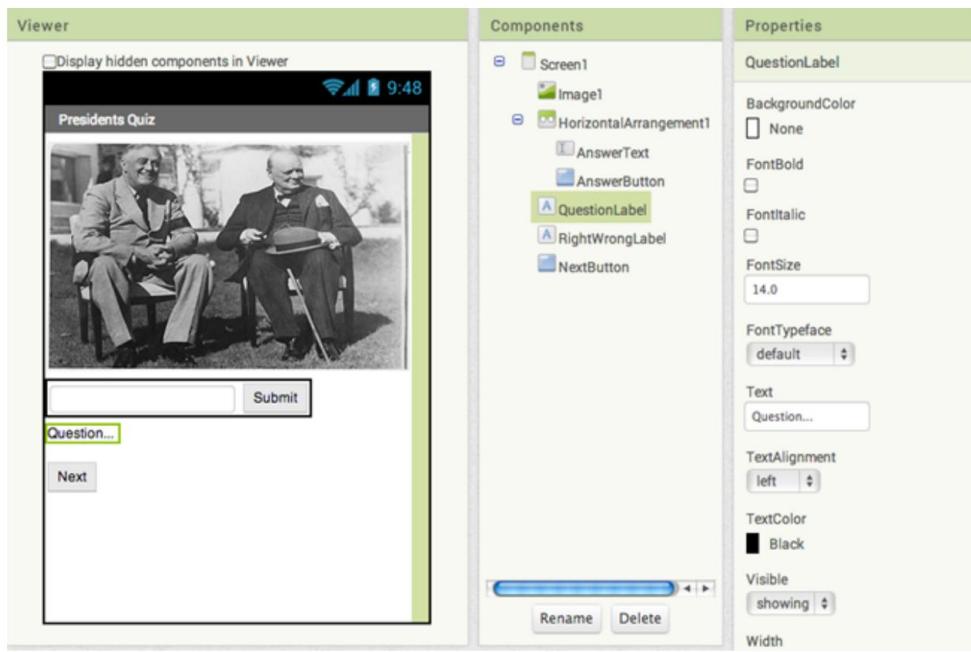


Figure 8-3. Le quiz des présidents dans le Designer

Pour créer cette interface, chargez d'abord les images que vous avez téléchargées dans le projet. Dans la zone Média, cliquez sur Ajouter et sélectionnez l'un des fichiers téléchargés (par exemple, roosChurch.gif). Faites de même pour les trois autres images. Ajoutez ensuite les composants répertoriés dans le tableau 8-1.

Tableau 8-1. Composants de l'application Presidents Quiz

Type de composant	Groupe de palettes	Comment vous l'appellerez	But
Image	Image de l'interface utilisateur	utilisateur1	L'image affichée avec le question.
Étiquette	Étiquette de question de l'interface utilisateur		Affiche la question actuelle.
Disposition de l'arrangement horizontal		Disposition horizontale1	Mettez le texte de la réponse et le bouton une rangée.
Zone de texte	Texte de réponse de l'interface utilisateur		L'utilisateur saisira sa réponse ici.
Bouton	Bouton de réponse de l'interface utilisateur		L'utilisateur clique dessus pour soumettre un répondre.
Étiquette	Interface utilisateur RightWrongLabel		Afficher « correct ! » ou « incorrect! »
Bouton	Interface utilisateur Bouton Suivant		L'utilisateur clique dessus pour passer à la question suivante.

132 Chapitre 8 : Quiz des présidents

1. Définissez Image1.Picture sur le fichier image roosChurch.gif, la première image qui doit apparaître.
Définissez sa largeur sur « Remplir le parent » et sa hauteur sur 200.
2. Définissez QuestionLabel.Text sur « Question... » (vous saisirez la première question dans le Editeur de blocs).
3. Définissez AnswerText.Hint sur « Entrer une réponse ». Définissez sa propriété Text sur vide. Se déplacer dans HorizontalArrangement1.
4. Remplacez AnswerButton.Text par « Soumettre » et déplacez-le dans Disposition horizontale1.
5. Remplacez NextButton.Text par « Suivant ».
6. Remplacez RightWrongLabel.Text par vide.

Ajout de comportements aux composants

Vous devrez programmer les comportements suivants :

- Lorsque l'application démarre, la première question apparaît, y compris son correspondant image.
- Lorsque l'utilisateur clique sur le NextButton, la deuxième question apparaît. Quand il est cliqué à nouveau, la troisième question apparaît, et ainsi de suite.
- Lorsque l'utilisateur atteint la dernière question et clique sur le bouton Suivant, le premier la question devrait réapparaître.
- Lorsque l'utilisateur répond à une question, l'application indiquera si elle est correcte.

Vous coderez ces comportements un par un, en les testant au fur et à mesure.

DÉFINITION DES LISTES DE QUESTIONS ET RÉPONSES

Pour commencer, définissez deux variables de liste basées sur les éléments du tableau 8-2 : QuestionList pour contenir la liste des questions et AnswerList pour contenir la liste des réponses correspondantes.

La figure 8-3 montre les deux listes que vous allez créer dans l'éditeur de blocs.

Tableau 8-2. Variables pour conserver les listes de questions et réponses

Type de bloc	Objectif du tiroir
initialisez global à (« QuestionList »)	Variables Stockez la liste des questions (renommez-la QuestionList).
initialiser global à ("AnswerList")	Variables Stocke la liste des réponses (renommez-la AnswerList).
fais une liste	Listes Insérez les éléments de la QuestionList.

Type de bloc	Objectif du tiroir
texte (trois d'entre eux)	Texte Questions.
fais une liste	Listes Insérez les éléments de la AnswerList.
texte (trois d'entre eux)	Texte Les réponses.

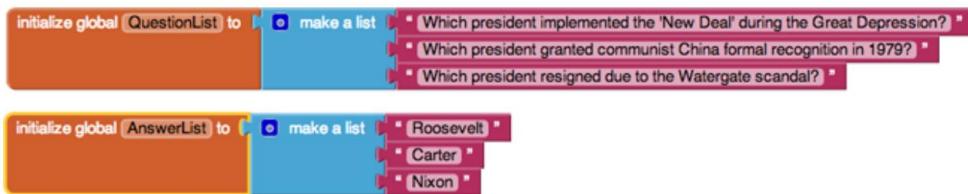


Figure 8-4. Les listes pour le quiz

DÉFINITION DE LA VARIABLE D'INDICE

L'application doit garder une trace de la question actuelle lorsque l'utilisateur clique sur le bouton NextButton. pour continuer le quiz. Vous définirez une variable nommée currentQuestionIndex pour ceci, et la variable servira d'index à la fois dans la QuestionList et Liste de réponses. Le Tableau 8-3 répertorie les blocs dont vous aurez besoin pour ce faire, et la Figure 8-4 montre ce à quoi ressemblera cette variable.

Tableau 8-3. Crédit de l'index

Type de bloc	Objectif du tiroir
initialiser global à (« IndexQuestionActuel »)	Variables Contiennent l'index (position) de la question/réponse actuelle.
numéro 1)	Mathématiques Définissez la valeur initiale de currentQuestionIndex sur 1 (la première question).

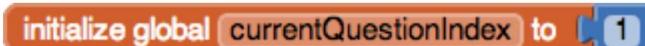


Figure 8-5. Initialisation de la variable d'index avec une valeur de 1

AFFICHAGE DE LA PREMIÈRE QUESTION

Maintenant que vous avez défini les variables dont vous avez besoin, vous pouvez spécifier le mode interactif de l'application. comportement. Comme pour toute application, il est important de travailler progressivement et d'en définir une. comportement à la fois. Pour commencer, pensez uniquement aux questions et, plus précisément, affichant la première question de la liste au lancement de l'application. Nous reviendrons et traiter les images et les réponses un peu plus tard.

[134 Chapitre 8 : Quiz des présidents](#)

Vous voulez que vos blocs de code fonctionnent quelles que soient les questions spécifiques posées la liste. De cette façon, si vous décidez de modifier les questions ou de créer un nouveau quiz en copiant et en modifiant cette application, vous n'aurez qu'à modifier les questions réelles dans les définitions de la liste, et vous n'aurez pas besoin de modifier les gestionnaires d'événements.

Donc, pour ce premier comportement, vous ne voulez pas faire référence directement à la première question : « Quel président a mis en œuvre le « New Deal » pendant la Grande Dépression ? Au lieu de cela, vous souhaitez faire référence, de manière abstraite, à la première socket de la QuestionList (quelle que soit la question spécifique qui s'y trouve). De cette façon, les blocs fonctionneront toujours même si vous modifiez la question dans ce premier socket.

Vous sélectionnez des éléments particuliers dans une liste avec le bloc de sélection d'éléments de liste . Le bloc vous demande de préciser la liste et un index (une position dans la liste). Si une liste comporte trois éléments, vous pouvez saisir 1, 2 ou 3 comme index.

Pour ce premier comportement, lors du lancement de l'application, vous souhaitez sélectionner le premier élément dans QuestionList et le placer dans QuestionLabel. Rappel depuis Android, où est ma voiture ? app, si vous souhaitez que quelque chose se produise au lancement de votre application, vous programmez ce comportement dans le gestionnaire d'événements Screen1.Initialize . Vous pouvez utiliser les blocs répertoriés dans le Tableau 8-4.

Tableau 8-4. Blocs pour charger la question initiale au démarrage de l'application

Type de bloc	Tiroir	But
Écran1.Initialiser	Écran1	Gestionnaire d'événements déclenché au démarrage de l'application.
définissez QuestionLabel.Text sur QuestionLabel Mettez la première question dans QuestionLabel.		
sélectionner un élément de la liste	Listes	Sélectionnez la première question dans QuestionList.
obtenir la liste de questions globale	Variables	La liste dans laquelle sélectionner des questions.
numéro 1)	Mathématiques	Sélectionnez la première question en utilisant un indice de 1.

Comment fonctionnent les blocs

L' événement Screen1.Initialize est déclenché au démarrage de l'application. Comme représenté sur la Figure 8-5, le premier élément de la variable QuestionList est sélectionné et placé dans QuestionLabel.Text. Ainsi, lorsque l'application sera lancée, l'utilisateur verra la première question.



Figure 8-6. Sélection de la première question



Testez votre application Cliquez sur Connecter et connectez-vous à votre appareil ou au émulateur pour les tests en direct. Lorsque votre application se charge, voyez-vous le premier élément de QuestionList, « Quel président a mis en œuvre le Le « New Deal » pendant la Grande Dépression ?

ITÉRER LES QUESTIONS

Maintenant, programmez le comportement du NextButton. Vous avez déjà défini le currentQuestionIndex pour mémoriser la question actuelle. Lorsque l'utilisateur clique sur le NextButton, l'application doit incrémenter le currentQuestionIndex (c'est-à-dire le modifier de 1 à 2 ou de 2 à 3, et ainsi de suite). Vous utiliserez ensuite la valeur résultante de currentQuestionIndex pour sélectionner la nouvelle question à afficher. Comme défi, voyez si vous pouvez construire ces blocs vous-même. Lorsque vous avez terminé, comparez vos résultats contre la figure 8-6.



Figure 8-7. Passer à la question suivante

Comment fonctionnent les blocs

La première rangée de blocs incrémentera la variable `currentQuestionIndex`. Si `currentQuestionIndex` contient un 1, il est remplacé par 2. S'il contient un 2, il est remplacé par 3, et ainsi de suite. Une fois la variable `currentQuestionIndex` modifiée, l'application l'utilise pour sélectionner la nouvelle question à afficher. Lorsque l'utilisateur clique sur `NextButton` pour la première fois, les blocs d'incrément changeront `currentQuestionIndex` de 1 à 2, de sorte que l'application sélectionnera le deuxième élément de `QuestionList`, « Quel président a accordé la reconnaissance formelle à la Chine communiste en 1979 ? » La deuxième fois que vous cliquez sur `NextButton`, `currentQuestionIndex` sera défini entre 2 et 3 et l'application sélectionnera la troisième question de la liste : « Quel président a démissionné à cause du scandale du Watergate ? »



Remarque Prenez une minute pour comparer les blocs de `Bouton Suivant`. Cliquez à ceux du gestionnaire d'événements `Screen.Initialize`. Dans le blocs `Screen.Initialize`, l'application a utilisé un élément de liste de sélection avec un numéro concret (1) pour sélectionner l'élément de liste. Dans ces blocs, vous sélectionnez l'élément de liste en utilisant une variable comme `index`. L'application ne choisit pas le premier élément de la liste, ni le deuxième, ni le troisième ; il choisit l'élément `currentQuestionIndexh`, et donc un élément différent sera sélectionné chaque fois que l'utilisateur clique sur `NextButton`. Il s'agit d'une utilisation très courante d'un `index` : incrémenter sa valeur pour rechercher et afficher ou traiter des éléments dans une liste.



Testez votre application Testez le comportement du NextButton pour voir si le l'application fonctionne correctement. Cliquez sur le bouton Suivant du téléphone. Le téléphone affiche-t-il la deuxième question « Quel président a-t-il accordé une reconnaissance formelle à la Chine communiste en 1979 ? Il devrait, et la troisième question devrait apparaître lorsque vous cliquez sur le NextButton à nouveau. Mais si vous cliquez à nouveau, vous devriez voir un erreur : « Tentative d'obtention de l'élément 4 d'une liste de longueur 3. » L'application il y a un bug ! Sais-tu quel est le problème ? Essayez de le imaginer sortir avant de continuer.

Le problème avec le code jusqu'à présent est qu'il s'incrémentera simplement au suivant question à chaque fois sans aucun souci de la fin du quiz. Quand currentQuestionIndex est déjà 3 et l'utilisateur clique sur le bouton Next, l'application change currentQuestionIndex de 3 à 4. Il appelle ensuite l'élément de liste sélectionné pour obtenir le élément currentQuestionIndexth : dans ce cas, le quatrième élément. Parce qu'il n'y a que trois éléments dans la variable QuestionList, l'appareil Android ne sait pas quoi faire faire. En conséquence, il affiche une erreur et force l'application à se fermer. Comment pouvons-nous laisser l'application sais-tu qu'il est arrivé à la fin du quiz ?

L'application doit poser une question lorsque l'utilisateur clique sur le bouton NextButton et l'exécuteur. différents blocs en fonction de la réponse. Parce que vous savez que votre application contient trois questions, une façon de poser la question serait : « La variable est-elle currentQuestionIndex supérieur à 3 ? Si la réponse est oui, vous devez définir currentQuestionIndex revient à 1 et ramène l'utilisateur à la première question. Le les blocs dont vous aurez besoin pour cela sont répertoriés dans le tableau 8-5.

Tableau 8-5. Blocs pour vérifier la valeur d'index pour la fin de la liste

Type de bloc	Tiroir	But
si	Contrôle	Déterminez si l'utilisateur est sur la dernière question.
>=	Mathématiques	Testez si currentQuestionIndex est supérieur à 3.
obtenir l'index global des questions actuelles	Faites glisser depuis initialiser le bloc	Mettez ceci dans le côté gauche de =.
numéro 3	Mathématiques	Mettez ceci à droite de = car 3 est le nombre de éléments de la liste.
définir l'index global currentQuestionIndex sur	Faites glisser depuis initialiser le bloc	Réglez sur 1 pour revenir à la première question.
numéro 1	Mathématiques	Définissez l'indice sur 1.

Les blocs doivent apparaître comme illustré dans la figure 8-7.

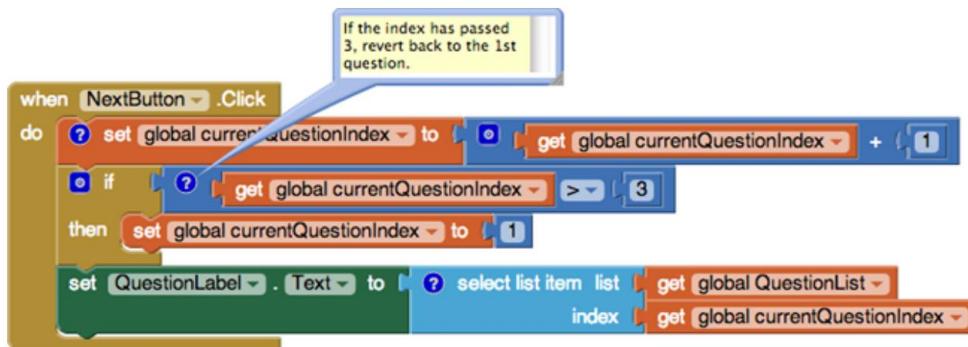


Figure 8-8. Vérifier si l'index dépasse la dernière question

Comment fonctionnent les blocs

Lorsque l'utilisateur clique sur NextButton, l'application incrémente l'index comme auparavant.

Mais ensuite, comme le montre la figure 8-8, il vérifie si currentQuestionIndex est supérieur à 3, qui correspond au nombre de questions. S'il est supérieur à 3, currentQuestionIndex est remis à 1 et la première question est affichée. S'il est égal à 3 ou moins, les blocs du bloc if ne sont pas exécutés et la question actuelle est affichée comme d'habitude.



Testez votre application Décrochez le téléphone et cliquez sur le bouton Suivant. La deuxième question : « Quel président a accordé la reconnaissance formelle à la Chine communiste en 1979 ? devrait apparaître dans le QuestionLabel sur le téléphone, comme auparavant. Lorsque vous cliquez à nouveau, la troisième question devrait apparaître sur le téléphone. Maintenant, pour le comportement que vous testez réellement : si vous cliquez à nouveau, vous devriez voir la première question (« Quel président a mis en œuvre le « New Deal » pendant la Grande Dépression ? ») apparaître sur le téléphone.

RENDRE LE QUIZ FACILE À MODIFIER

Si vos blocages pour le NextButton fonctionnent, félicitez-vous ; vous êtes sur la bonne voie pour devenir programmeur ! Mais que se passe-t-il si vous ajoutez une nouvelle question (et réponse) au quiz ? Vos blocs fonctionneraient-ils toujours ? Pour explorer cela, ajoutez d'abord une quatrième question à QuestionList et une quatrième réponse dans AnswerList, comme le montre la figure 8-8.

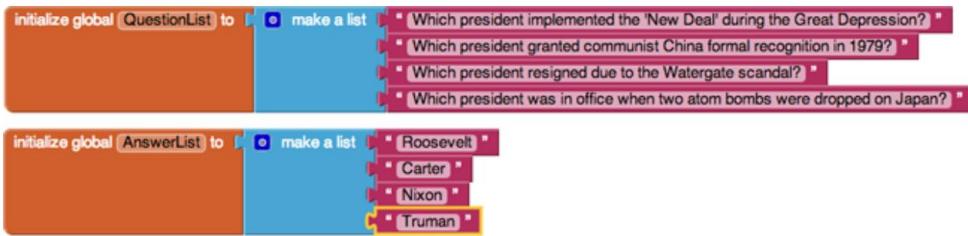


Figure 8-9. Ajouter un élément aux deux listes



Testez votre application Cliquez plusieurs fois sur le bouton Suivant. Vous remarquerez que la quatrième question n'apparaît jamais, quel que soit le nombre de fois que vous cliquez sur Suivant. Sais-tu quel est le problème? Avant de poursuivre votre lecture, voyez si vous pouvez corriger les blocs pour que la quatrième question apparaisse.

Le problème ici est que le test permettant de déterminer si l'utilisateur a répondu à la dernière question est trop spécifique ; il demande si la variable `currentQuestionIndex` est supérieure à 3. Vous pouvez simplement changer le chiffre 3 en 4 et l'application fonctionnera à nouveau correctement. Le problème avec cette solution, cependant, est que chaque fois que vous modifiez les listes de questions et de réponses, vous devez également penser à apporter cette modification au test if.

De telles dépendances dans un programme informatique conduisent souvent à des bugs, notamment en tant qu'application. gagne en complexité. Une bien meilleure stratégie consiste à concevoir les blocs de manière à ce qu'ils fonctionnent quel que soit le nombre de questions. Une telle généralité facilite la tâche si vous, en tant que programmeur, souhaitez personnaliser votre quiz sur un autre sujet. C'est également essentiel si la liste avec laquelle vous travaillez change de manière dynamique : par exemple, pensez à une application de quiz qui permet à l'utilisateur d'ajouter de nouvelles questions (vous la construirez au chapitre 10). Pour qu'un programme soit plus général, il ne peut pas faire référence à des nombres concrets tels que 3, car cela ne fonctionne que pour des quiz de trois questions.

Ainsi, au lieu de demander si la valeur de `currentQuestionIndex` est supérieure au nombre spécifique 3, demandez si elle est supérieure au nombre d'éléments dans `QuestionList`. Si l'application pose cette question plus générale, elle fonctionnera même si vous ajoutez ou supprimez des éléments de la liste de questions. Modifions le gestionnaire d'événements `NextButton.Click` pour remplacer le test précédent qui faisait directement référence à 3. Vous aurez besoin des blocs répertoriés dans le tableau 8-6.

Tableau 8-6. Blocs pour vérifier la longueur de la liste

Type de bloc	Tiroir	But
longueur de la liste	Listes	Demandez combien d'éléments se trouvent dans <code>QuestionList</code> .

Type de bloc	Tiroir	But
obtenir la liste de questions globale	Faire glisser depuis le bloc d'initialisation	Mettez-le dans la prise « list » de la longueur de la liste.

Comment fonctionnent les blocs

Le test if compare maintenant currentQuestionIndex à la longueur de QuestionList, comme le montre la figure 8-11. Ainsi, si currentQuestionIndex est 5 et que la longueur de QuestionList est 4, alors currentQuestionIndex sera remis à 1. Notez que, comme les blocs ne font plus référence à 3 ou à un nombre spécifique, le comportement fonctionnera quel que soit le nombre de blocs. les éléments sont dans la liste.

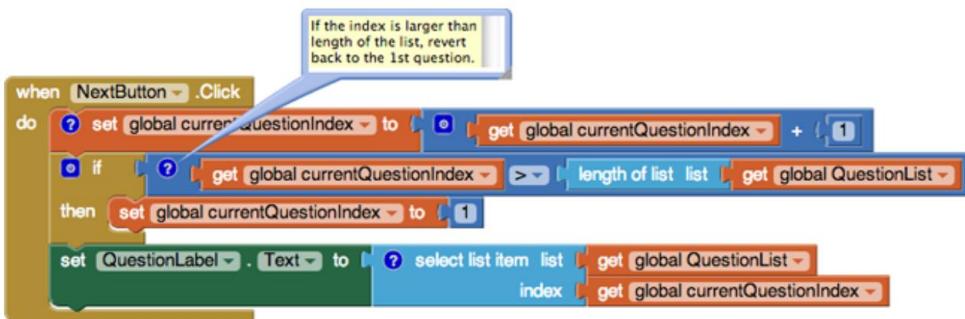


Figure 8-10. Vérifier si l'index est plus grand que la longueur de la liste (au lieu de 3)



Testez votre application Lorsque vous cliquez sur le bouton Suivant, l'application passe-t-elle désormais en revue les quatre questions, passant à la première après la quatrième ?

CHANGEMENT D'IMAGE POUR CHAQUE QUESTION

Maintenant que vous avez programmé tous les comportements pour parcourir les questions (et que vous avez rendu votre code plus intelligent et plus flexible en le rendant plus abstrait), votre prochaine tâche consiste également à faire fonctionner correctement les images. À l'heure actuelle, l'application affiche la même image quelle que soit la question posée. Vous pouvez modifier cela afin qu'une image relative à chaque question apparaisse lorsque l'utilisateur clique sur NextButton.

Plus tôt, vous avez ajouté quatre images comme support pour le projet. Vous allez maintenant créer une troisième liste, PictureList, avec les noms de fichiers des images comme éléments. Vous modifierez également le gestionnaire d'événements NextButton.Click pour changer d'image à chaque fois, tout comme vous changez le texte de la question. (Si vous envisagez déjà d'utiliser currentQuestionIndex ici, vous êtes sur la bonne voie !) Tout d'abord, créez une PictureList et initialisez-la avec les noms des fichiers image. Assurez-vous que les noms sont exactement les mêmes que ceux

noms de fichiers que vous avez chargés dans la section Media du projet. La figure 8-10 montre comment le les blocs pour la PictureList devraient ressembler.

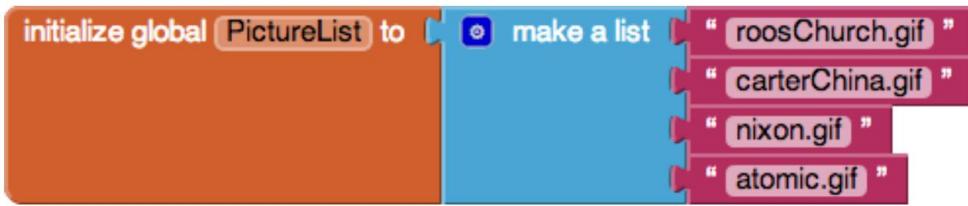


Figure 8-11. La PictureList avec les noms de fichiers d'images comme éléments

Ensuite, modifiez le gestionnaire d'événements NextButton.Click pour qu'il change l'image qui apparaît pour chaque question. La propriété Image.Picture est utilisée pour modifier la image affichée. Pour modifier NextButton.Click, vous aurez besoin des blocs répertoriés dans Tableau 8-7.

Tableau 8-7. Des blocs pour ajouter l'image qui accompagne la question

Type de bloc	Tiroir	But
définir Image1.Picture sur	Image1	Réglez ceci pour changer l'image.
sélectionner un élément de la liste	Listes	Sélectionnez l'image correspondant au courant question.
obtenir une liste d'images globale	Sortez de bloc d'initialisation	Sélectionnez un nom de fichier dans cette liste.
obtenir l'index global des questions actuelles	Sortez de bloc d'initialisation	Sélectionnez l'élément currentQuestionIndex.

Comment fonctionnent les blocs

Le currentQuestionIndex sert d'index à la fois pour la QuestionList et pour le Liste d'images. Tant que vous avez correctement configuré vos listes de manière à ce que la première question correspond à la première image, la seconde à la seconde, et ainsi de suite, l'index unique peut servir les deux listes, comme le montre la figure 8-11. Par exemple, la première photo, roosChurch.gif, est une photo du président Franklin Delano Roosevelt (assis avec les Britanniques). Premier ministre Winston Churchill), et « Roosevelt » est la réponse à la première question.

142 Chapitre 8 : Quiz des présidents

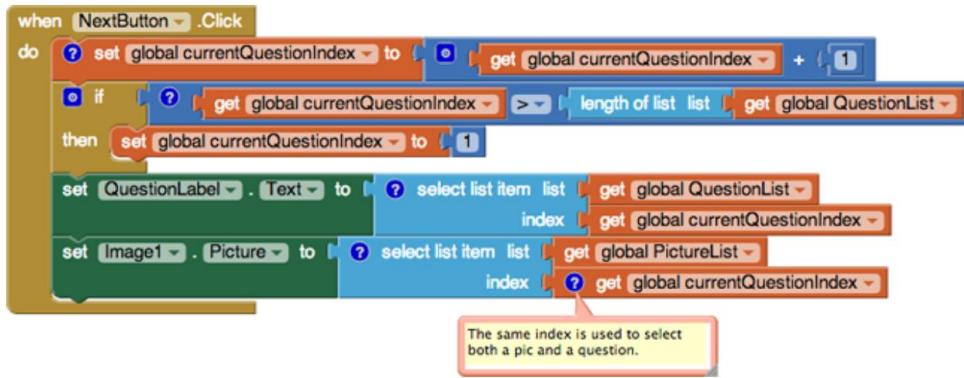


Figure 8-12. Sélection de l'image currentQuestionIndex à chaque fois



Testez votre application Cliquez sur Suivant plusieurs fois. Est-ce qu'une image différente apparaît-il chaque fois que vous cliquez sur le bouton NextButton ?

VÉRIFICATION DES RÉPONSES DE L'UTILISATEUR

Jusqu'à présent, vous avez créé une application qui passe simplement en revue les questions et les réponses. (associé à une image de la réponse). C'est un excellent exemple d'applications qui utilisent des listes, mais pour être une véritable application de quiz, elle doit donner aux utilisateurs des informations sur la pertinence de leurs réponses. correct. Ajoutons les blocs pour faire exactement cela. Notre interface est configurée pour que l'utilisateur tape une réponse dans AnswerText, puis clique sur AnswerButton. L'application doit comparer l'entrée de l'utilisateur avec la réponse à la question actuelle, en utilisant un if else bloc pour vérifier. Le RightWrongLabel doit ensuite être modifié pour indiquer si le la réponse est correcte. De nombreux blocs sont nécessaires pour programmer ce comportement, tous qui sont répertoriés dans le tableau 8-8.

Tableau 8-8. Blocs pour indiquer si une réponse est correcte

Type de bloc	Tiroir	But
AnswerButton.Cliquez	Bouton de réponse	Déclenché lorsque l'utilisateur clique sur le Bouton de réponse.
sinon	Contrôle	Si la réponse est correcte, faites une chose ; sinon, en faire un autre.
=	Mathématiques	Demandez si la réponse est correcte.
RéponseTexte.Texte	Texte de réponse	Contient la réponse de l'utilisateur.
sélectionner un élément de la liste	Listes	Sélectionnez la réponse actuelle dans AnswerList.
obtenir la liste de réponses globale	Sortez de bloc d'initialisation	La liste à sélectionner.

Type de bloc	Tiroir	But
obtenir l'index global des questions actuelles	Sortez de bloc d'initialisation	Le numéro actuel de la question (et de la réponse).
définir RightWrongLabel.Text sur RightWrongLabel		Signalez la réponse ici.
texte (« correct ! »)	Texte	Affichez-le si la réponse est bonne.
définir RightWrongLabel.Text sur RightWrongLabel		Signalez la réponse ici.
texte (« incorrect ! »)	Texte	Affichez-le si la réponse est fausse.

Comment fonctionnent les blocs

La figure 8-12 montre comment le test if else demande si la réponse à l'utilisateur fourni (AnswerText.Text) est égal à l' élément currentQuestionIndex dans la Liste de réponses. Si currentQuestionIndex est 1, l'application comparera la réponse de l'utilisateur avec le premier élément de AnswerList, « Roosevelt ». Si currentQuestionIndex est 2, l'application comparez la réponse de l'utilisateur avec la deuxième réponse de la liste, « Carter », et ainsi de suite. Si le résultat du test est positif, la branche then est exécutée et le RightWrongLabel est défini corriger! Si le test est faux, la branche else est exécutée et le RightWrongLabel est réglé sur « incorrect ! »

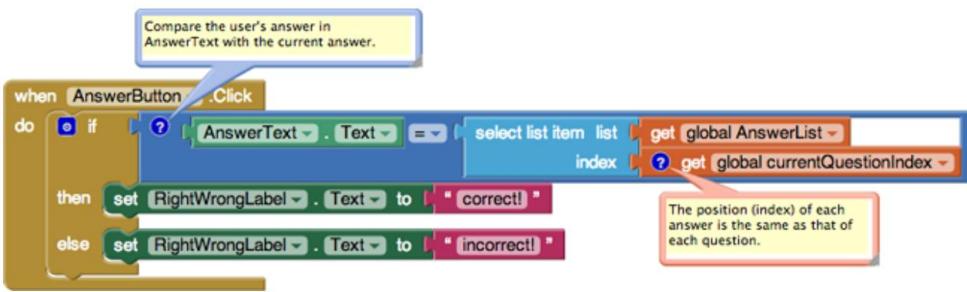


Figure 8-13. Vérification de la réponse



Testez votre application Essayez de répondre à l'une des questions. Cela devrait indiquer si vous avez répondu à la question exactement comme spécifié dans la liste de réponses. Testez avec un correct et un incorrect répondre. Vous remarquerez probablement que pour qu'une réponse soit marquée comme correct, il doit correspondre exactement à ce qui est spécifique au cas vous avez entré dans la AnswerList. Assurez-vous également de tester ces choses travailler sur des questions successives.

L'application devrait fonctionner, mais vous remarquerez peut-être que lorsque vous cliquez sur le bouton Suivant, la bonne!" ou "incorrect!" le texte et la réponse précédente sont toujours là, comme indiqué dans

144 Chapitre 8 : Quiz des présidents

Figure 8-13, même si vous regardez la question suivante. C'est assez inoffensif, mais les utilisateurs de votre application remarqueront certainement de tels problèmes d'interface utilisateur. Pour effacer RightWrongLabel et AnswerText, vous placerez les blocs répertoriés dans le tableau 8-9 dans le gestionnaire d'événements NextButton.Click .

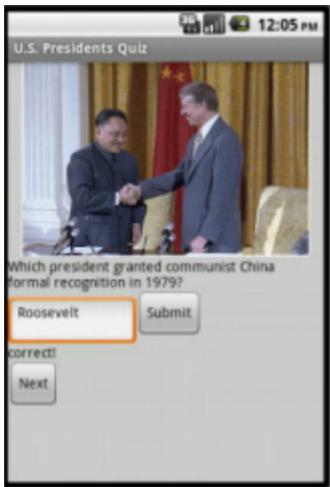


Figure 8-14. Le quiz en action avec la réponse précédente apparaissant alors qu'elle ne devrait pas

Tableau 8-9. Blocs pour effacer le RightWrongLabel

Type de bloc	Tiroir	But
définissez RightWrongLabel.Text sur RightWrongLabel	C'est l'étiquette à effacer.	
texte ("")	Texte	Lorsque l'utilisateur clique sur NextButton, effacez le précédent commentaire de la réponse.
définir AnswerText.Text sur	Texte de réponse	La réponse de l'utilisateur à la question précédente.
texte ("")	Texte	Lorsque l'utilisateur clique sur le bouton Next, effacez la réponse précédente.

Comment fonctionnent les blocs

Comme le montre la Figure 8-14, en cliquant sur le bouton Suivant, l'utilisateur passe au suivant question, donc les deux premières lignes du gestionnaire d'événements effacent RightWrongLabel et le texte de réponse.

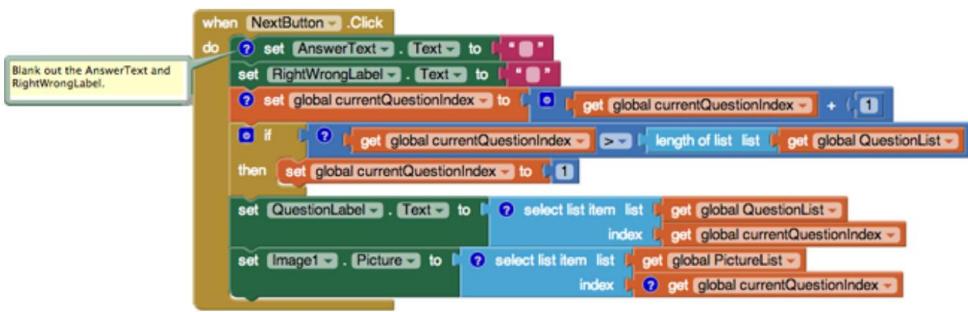


Figure 8-15. La PictureList avec les noms de fichiers d'images comme éléments



Testez votre application Répondez à une question et cliquez sur Soumettre, puis cliquez sur le bouton Suivant. Votre réponse précédente et ses commentaires disparaîtront?

L'application complète : le quiz des présidents

La figure 8-15 montre la configuration du bloc final pour le quiz des présidents.

146 Chapitre 8 : Quiz des présidents

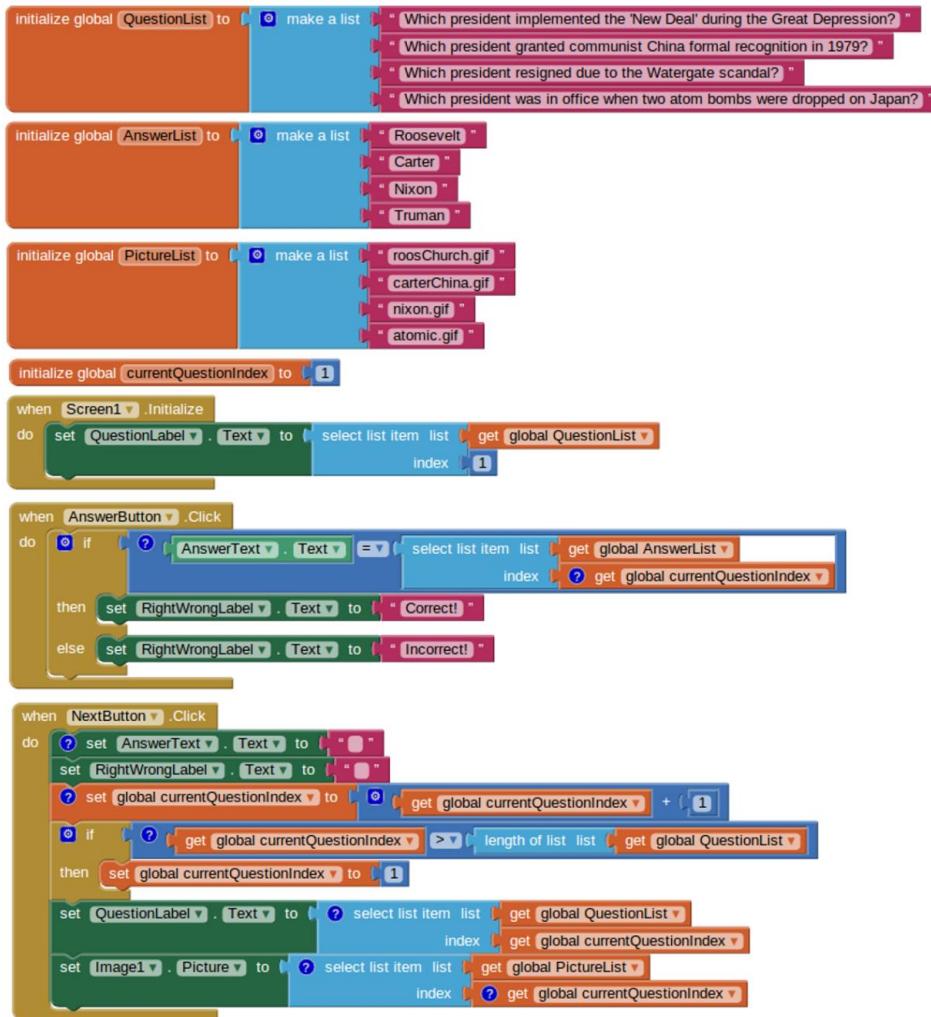


Figure 8-16. Les blocs du Quiz des Présidents

Variantes

Une fois que ce quiz sera opérationnel, vous souhaiterez peut-être explorer certaines variantes, telles que les suivantes :

- Au lieu de simplement montrer des images pour chaque question, essayez de diffuser un extrait sonore ou une courte vidéo. Avec le son, vous pouvez transformer votre quiz en une application Name That Tune.
- Le quiz est très rigide quant à ce qu'il accepte comme réponse valable. Il existe plusieurs façons d'améliorer cela, en tirant parti des blocs de traitement de texte dans le tiroir de texte. Une solution consiste à utiliser le bloc uppercase dans le tiroir Texte pour

convertissez la réponse de l'utilisateur et la réponse réelle en majuscules avant de comparer. Une autre consiste à utiliser le bloc `text.contains` pour voir si la réponse de l'utilisateur est contenu dans la réponse réelle. Une autre option consiste à fournir plusieurs réponses pour chaque question et à vérifier en les itérant (`foreach`) pour voir s'il y en a une qui correspond.

- Une autre façon d'améliorer la vérification des réponses consiste à transformer le quiz pour qu'il soit à choix multiples. Vous aurez besoin d'une liste supplémentaire pour contenir les choix de réponses pour chaque question. Les réponses possibles seront une liste de listes, chaque sous-liste contenant les choix de réponses pour une question particulière. Utilisez le composant `ListPicker` pour donner à l'utilisateur la possibilité de choisir une réponse. Vous pouvez en savoir plus sur les listes au chapitre 19.

Résumé

Voici quelques-unes des idées que nous avons abordées dans ce tutoriel :

- La plupart des applications assez sophistiquées contiennent des données (souvent stockées dans des listes) et des comportements gestionnaires d'événements.
- Utilisez un bloc `ifelse` pour vérifier les conditions. Pour plus d'informations sur les conditionnels, voir le chapitre 18.
- Les blocs dans les gestionnaires d'événements doivent faire référence uniquement de manière abstraite aux éléments de la liste et à la taille de la liste afin que l'application fonctionne même si les données de la liste sont modifiées.
- Les variables d'index suivent la position actuelle d'un élément dans une liste. Lorsque vous les incrémentez, utilisez un bloc `if` pour gérer le comportement de l'application lorsque l'utilisateur atteint la fin de la liste.

Xylophone

Figure 9-1.



Il est difficile de croire qu'utiliser la technologie pour enregistrer et la lecture de musique ne remonte qu'à 1878, lorsque Edison a breveté le phonographe. Depuis, nous avons parcouru beaucoup de chemin : avec les synthétiseurs musicaux, les CD, l'échantillonnage et le remixage, les téléphones qui diffusent de la musique et même le brouillage longue distance sur Internet. Dans ce chapitre, vous participerez à cette tradition en créant une application Xylophone qui enregistre et joue de la musique.

Ce que vous construirez

Avec l'application illustrée à la figure 9-1 (crée à l'origine par Liz Looney de l'équipe App Inventor), tu peux :

- Jouez huit notes différentes en touchant les boutons de couleur sur le écran.
- Appuyez sur un bouton Lecture pour rejouer les notes que vous avez jouées.
- Joué plus tôt.
- Appuyez sur un bouton Réinitialiser pour que l'application efface toutes les notes que vous avez jouées précédemment afin que vous puissiez saisir une nouvelle chanson.

Ce que vous apprenez

Ce didacticiel couvre les concepts suivants :

- Utilisation d'un seul composant Sound pour jouer différents fichiers audio.
- Utilisation du composant Clock pour mesurer et appliquer des délais entre les actions.

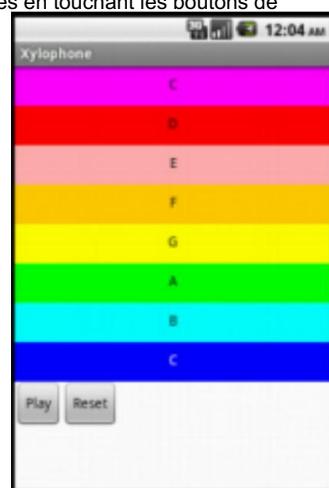


Figure 9-2. L'interface utilisateur de l'application Xylophone

150 Chapitre 9 : Xylophone

- Décider quand créer une procédure.
- Créer une procédure qui s'appelle toute seule.
- Utilisation avancée des listes, y compris l'ajout d'éléments, leur accès et la suppression des éléments liste.

Commencer

Connectez-vous au site Web App Inventor et démarrez un nouveau projet. Nommez-le "Xylophone", et définissez également le titre de l'écran sur « Xylophone ». Connectez votre application à votre appareil ou émulateur.

Conception des composants

Cette application comporte 13 composants différents (dont 8 constituent le clavier), qui sont répertoriés dans le tableau 9-1. Parce qu'il y en a tellement, ça deviendrait assez ennuyeux de tout créer d'entre eux avant de commencer à écrire notre programme, nous allons donc décomposer l'application en ses pièces fonctionnelles et construisez-les séquentiellement en faisant des allers-retours entre les Designer et l'éditeur de blocs, comme nous l'avons fait avec l'application Ladybug Chase au chapitre 5.

Tableau 9-1. Tous les composants de l'application Xylophone

Type de composant	Groupe de palettes	Comment vous l'appellerez	But
Bouton	Bouton de l'interface	utilisateur 1	Jouez la touche C grave.
Bouton	Bouton de l'interface	utilisateur2	Jouez la touche D.
Bouton	Bouton de l'interface	utilisateur3	Jouez sur la touche E.
Bouton	Bouton de l'interface	utilisateur4	Jouez sur la touche F.
Bouton	Bouton de l'interface	utilisateur5	Jouez sur la touche G.
Bouton	Bouton de l'interface	utilisateur6	Jouez une touche.
Bouton	Bouton de l'interface	utilisateur7	Jouez la touche B.
Bouton	Bouton de l'interface	utilisateur8	Jouez la touche High C.
Son	Médias	Son1	Jouez les notes.
Bouton	Bouton de lecture de l'interface utilisateur		Rejouez la chanson.
Bouton	Bouton de réinitialisation de l'interface utilisateur		Réinitialisez la mémoire des morceaux.
Disposition de l'arrangement horizontal		Disposition horizontale1	Placez les boutons Play et Reset à côté de l'autre.
Horloge	Horloge de l'interface	utilisateur1	Gardez une trace des délais entre Remarques.

Création du clavier

Notre interface utilisateur comprendra un clavier à huit notes pour une gamme majeure pentatonique (sept notes) allant du do grave au do aigu. Nous créerons ce clavier musical dans cette section.

CRÉATION DES PREMIERS BOUTONS DE NOTE

Commencez par créer les deux premières touches du xylophone, que nous implémenterons sous forme de boutons.

1. Dans la catégorie Interface utilisateur, faites glisser un bouton sur l'écran. Laissez son nom comme Button1. Nous voulons que ce soit une longue barre magenta, comme celle d'un xylophone, alors définissez ses propriétés comme suit :
 - Remplacez la propriété BackgroundColor par Magenta.
 - Remplacez la propriété Text par « C ».
 - Définissez la propriété Largeur sur « Remplir le parent » afin qu'elle s'étende sur toute la longueur l'écran.
 - Définissez la propriété Height sur 40 pixels.
2. Répétez l'opération pour un deuxième bouton, nommé Button2, en le plaçant sous Button1. Utilisez les mêmes valeurs de propriété Largeur et Hauteur, mais définissez sa propriété BackgroundColor sur Red et sa propriété Text sur « D ».

(Plus tard, nous répéterons l'étape 2 pour six autres boutons de note.)

La vue dans Component Designer devrait ressembler à la figure 9-2.

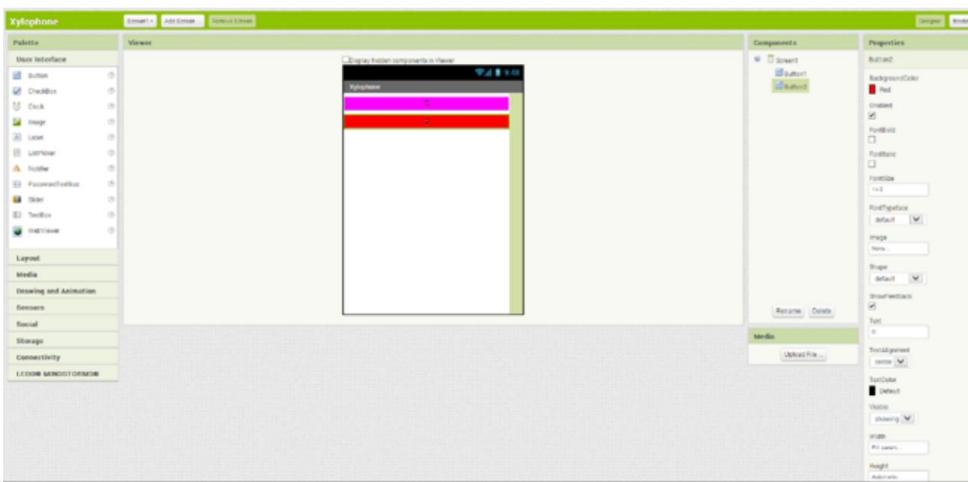


Figure 9-3. Placer des boutons pour créer un clavier

152 Chapitre 9 : Xylophone

L'affichage de votre téléphone devrait être similaire, même s'il n'y en aura pas.
espace vide entre les deux boutons colorés.

AJOUT DU COMPOSANT SONORE

Nous ne pouvons pas avoir de xylophone sans sons, alors faites glisser un composant Sound en laissant son nom Sound1. Modifiez la propriété MinimumInterval de sa valeur par défaut de 500 millisecondes à 0. Cela nous permet de jouer le son aussi souvent que nous le souhaitons, au lieu de devoir attendre une demi-seconde (500 millisecondes) entre les lectures. Ne définissez pas sa propriété Source , que nous définirons dans l'éditeur de blocs.

Téléchargez les fichiers sonores : <http://appinventor.org/bookFiles/Xylophone/1.wav> et <http://appinventor.org/bookFiles/Xylophone/2.wav>. Contrairement aux chapitres précédents, où il était possible de changer les noms des fichiers multimédias, il est important d'utiliser ces noms exacts pour des raisons qui deviendront bientôt claires. Vous pouvez télécharger les six fichiers sonores restants lorsque vous y serez invité plus tard.

CONNEXION DES SONS AUX BOUTONS

Le comportement que nous devons programmer est qu'un fichier sonore soit joué lorsque le bouton correspondant est cliqué. Plus précisément, si l'on clique sur Button1 , nous aimerions lire 1.wav ; si l'on clique sur Button2 , nous aimerions jouer à 2.wav ; et ainsi de suite. Nous pouvons configurer cela dans l'éditeur de blocs, comme le montre la figure 9-3, en procédant comme suit :

1. Depuis le tiroir Button1, faites glisser le bloc Button1.Click .
2. Depuis le tiroir Sound1, faites glisser le bloc Sound1.Source défini et placez-le dans le Bouton 1.Cliquez sur le bloc.
3. Tapez « texte » pour créer un bloc de texte. (C'est plus rapide que d'aller dans l'onglet Intégré puis dans le tiroir Texte, même si cela fonctionnerait aussi.) Définissez sa valeur de texte sur « 1.wav » et placez-la dans le bloc Sound1.Source .
4. Ajoutez un bloc Sound1.Play .

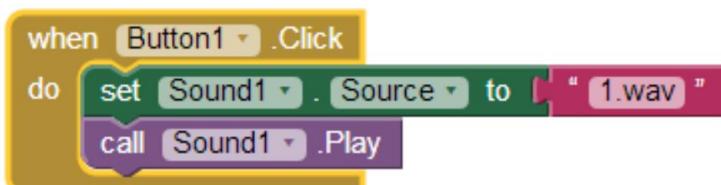


Figure 9-4. Jouer un son lorsqu'un bouton est cliqué

Nous pourrions faire la même chose pour Button2, comme le montre la figure 9-4 (en changeant simplement la valeur du texte), mais le code serait terriblement répétitif.



Figure 9-5. Ajouter plus de sons

Un code répété est un bon signe que vous devez créer une procédure, ce que vous avez déjà fait dans le jeu MoleMash au chapitre 3 et dans le jeu Ladybug Chase au chapitre 5. Plus précisément, nous allons créer une procédure qui prend un nombre comme paramètre. , définit la source de Sound1 sur le fichier approprié et joue le son. Il s'agit d'un autre exemple de refactoring : améliorer l'implémentation d'un programme sans modifier son comportement, un concept introduit dans le didacticiel MoleMash. Nous pouvons utiliser le bloc de jointure du tiroir Texte pour combiner le numéro (par exemple, 1) et le texte « .wav » pour créer le nom de fichier approprié (par exemple « 1.wav »). Voici les étapes pour créer la procédure dont nous avons besoin :

1. Sous l'onglet Intégré, accédez au tiroir Procédures et faites glisser le vers la procédure blocue. (Sauf indication contraire, vous devez choisir la version avec « faire », et non « résultat ».)
2. Ajoutez le paramètre en cliquant sur la petite icône bleue du bloc de procédure , en faisant glisser sur une entrée et en changeant son nom de « x » à « numéro ». Vous souhaiterez peut-être consulter la figure 5-6 du chapitre 5.
3. Cliquez sur le nom de la procédure, qui est par défaut « procédure » et définissez-le sur "JouerNote".
4. Faites glisser le bloc Sound1.Source de Button1.Click dans PlayNote à droite du mot « do ». Déplacez également le bloc Sound1.Play dans PlayNote.
5. Faites glisser le bloc 1.wav dans la corbeille.
6. Depuis le tiroir Texte, faites glisser le bloc de jointure dans le socket de Sound1.Source .
7. Tapez « numéro » et déplacez-le vers le socket supérieur du bloc de jointure (si ce n'est pas le cas). déjà là).
8. Depuis le tiroir Texte, faites glisser le bloc de texte dans le deuxième socket de la jointure. bloc.
9. Remplacez la valeur du texte par « .wav ». (N'oubliez pas de ne pas taper les guillemets.)
10. Dans le tiroir Procédures, faites glisser un bloc d'appel PlayNote et placez-le dans le corps vide de Button1.Click.
11. Tapez « 1 » et placez-le dans la prise « numéro ».

154 Chapitre 9 : Xylophone

Désormais, lorsque Button1 est cliqué, la procédure PlayNote sera appelée, avec son paramètre numérique ayant la valeur 1. Elle doit définir Sound1.Source sur « 1.wav » et jouer le son.

Créez un bloc Button2.Click similaire avec un appel à PlayNote avec un paramètre de 2. (Vous pouvez copier le bloc d'appel PlayNote existant et le déplacer dans le corps de Button2.Click, en veillant à modifier le paramètre.) Votre programme devrait ressembler à la figure 9-5.

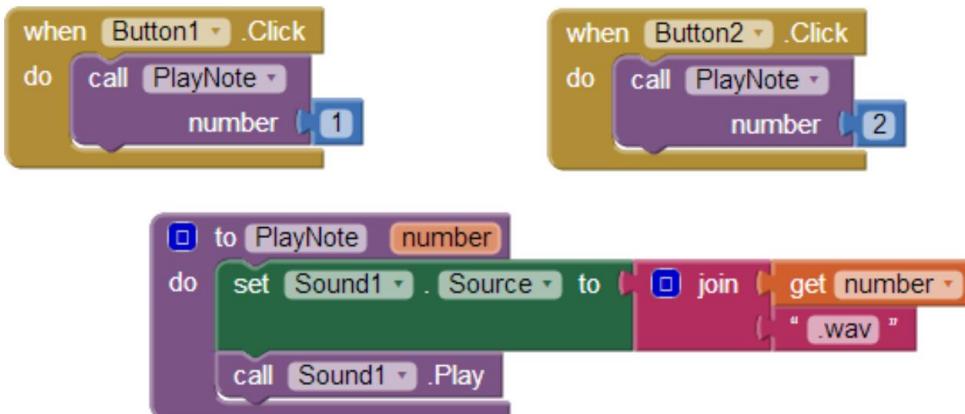


Figure 9-6. Créer une procédure pour jouer une note

INSTRUCTION À ANDROID DE CHARGER LES SONS

Si vous avez essayé les appels précédents vers PlayNote, vous avez peut-être été déçu de ne pas entendre le son que vous attendiez ou de rencontrer une erreur ou un retard inattendu. En effet, Android doit charger les sons au moment de l'exécution, ce qui entraîne un certain décalage avant qu'ils puissent être lus. Ce problème ne s'est pas produit plus tôt car les noms de fichiers placés dans la propriété Source d'un composant Sound dans le Designer sont automatiquement chargés au démarrage du programme. Étant donné que nous ne définissons Sound1.Source qu'après le démarrage du programme, ce processus d'initialisation n'a pas lieu. Nous devons charger explicitement les sons au démarrage du programme, comme le montre la figure 9-6.

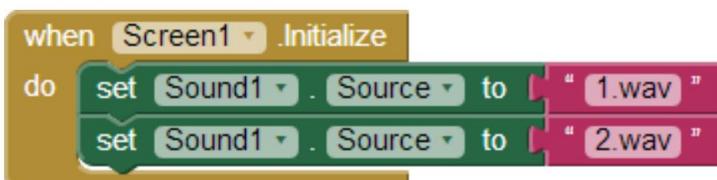


Figure 9-7. Chargement des sons au lancement de l'application



Testez votre application Touchez les boutons et vérifiez si les notes sont jouées sans délai. (Si vous n'entendez rien, assurez-vous que le volume multimédia de votre téléphone n'est pas réglé sur sourdine.)

MISE EN ŒUVRE DES NOTES RESTANTES

Maintenant que les deux premiers boutons et notes sont implémentés et fonctionnent, ajoutez les six notes restantes en revenant dans le Designer et en téléchargeant les fichiers audio :

- <http://appinventor.org/bookFiles/Xylophone/3.wav>
- <http://appinventor.org/bookFiles/Xylophone/4.wav>
- <http://appinventor.org/bookFiles/Xylophone/5.wav>
- <http://appinventor.org/bookFiles/Xylophone/6.wav> •
- <http://appinventor.org/bookFiles/Xylophone/7.wav>
- <http://appinventor.org/bookFiles/Xylophone/8.wav>

Ensuite, créez six nouveaux boutons, en suivant les mêmes étapes que pour le précédent. deux mais en définissant leurs propriétés Text et BackgroundColor comme suit :

- Bouton 3 (« E », rose)
- Bouton 4 (« F », Orange)
- Bouton 5 (« G », Jaune)
- Bouton 6 (« A », vert)
- Bouton 7 (« B », Cyan)
- Bouton 8 (« C », Bleu)

Vous pouvez également modifier la propriété TextColor de Button8 en White, comme le montre la figure 9-7, afin qu'elle soit plus lisible.

156 Chapitre 9 : Xylophone

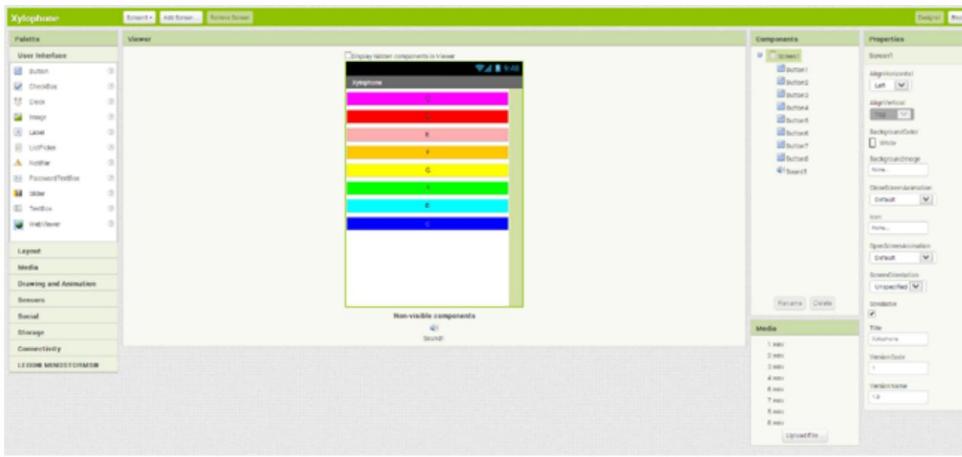


Figure 9-8. Placer les boutons et sons restants dans le Concepteur de composants

De retour dans l'éditeur de blocs, créez des blocs de clic pour chacun des nouveaux boutons avec appels appropriés à PlayNote. De même, ajoutez chaque nouveau fichier audio à Screen.Initialize, comme indiqué dans la figure 9-8.

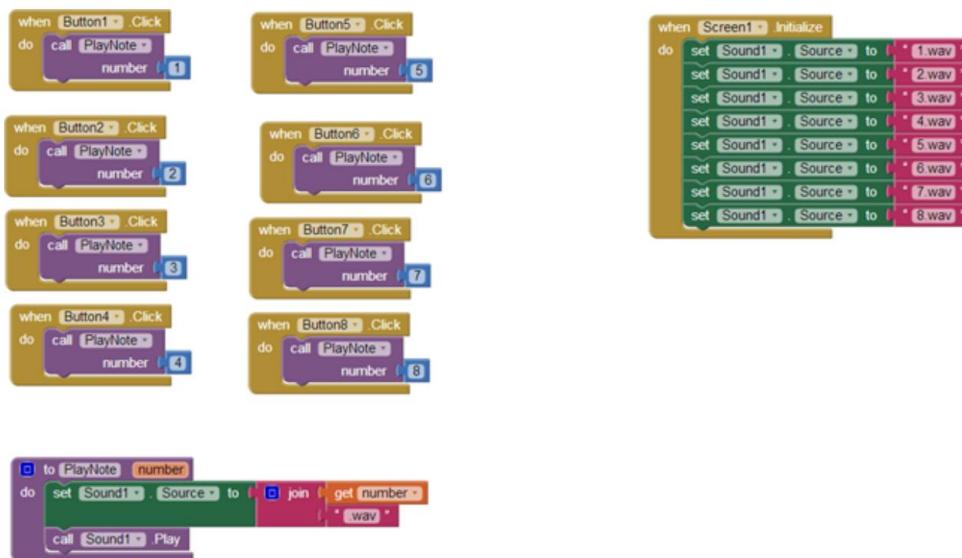
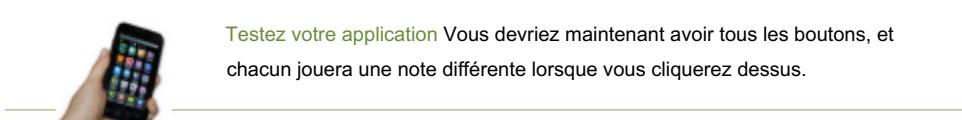


Figure 9-9. Programmation des événements de clic de bouton pour qu'ils correspondent à toutes les touches du clavier



Enregistrement et lecture de notes

Jouer des notes en appuyant sur des boutons est amusant, mais pouvoir enregistrer et lire des chansons est encore mieux. Pour implémenter la lecture, nous devrons conserver un enregistrement des notes jouées. En plus de mémoriser les hauteurs (fichiers sonores) qui ont été jouées, nous devons également enregistrer le temps entre les notes, sinon nous ne pourrons pas faire la distinction entre deux notes jouées en succession rapide et deux jouées avec un silence de 10 secondes. entre eux.

Notre application maintiendra deux listes, chacune comportant une entrée pour chaque note jouée :

- des notes, qui contiendront les noms des fichiers son dans l'ordre dans lequel ils ont été joués.
- les heures, qui enregistreront les instants auxquels les notes ont été jouées.

Remarque Avant de continuer, vous souhaiterez peut-être revoir les listes, qui sont couvertes dans le quiz des présidents au chapitre 8 et au chapitre 19.



Nous pouvons obtenir les informations de synchronisation à partir d'un composant Clock , que nous utiliserons également pour synchroniser correctement les notes pour la lecture.

AJOUT DES COMPOSANTS

Dans le Designer, vous devrez ajouter un composant Clock et des boutons Play et Reset, que vous placerez dans un HorizontalArrangement :

1. Dans le tiroir Capteurs, faites glisser un composant Horloge . Il apparaîtra dans le Rubrique « Composants non visibles ». Décochez sa propriété TimerEnabled car nous ne voulons pas que sa minuterie se déclenche jusqu'à ce que nous le lui disons pendant la lecture.
2. Accédez au tiroir Mise en page et faites glisser un composant HorizontalArrangement sous le bouton existant. Définissez sa propriété width sur « Fill parent ».
3. Dans le tiroir Interface utilisateur, faites glisser un bouton. Renommez-le "PlayButton" et définissez sa propriété Text sur « Play ».
4. Faites glisser un autre bouton, en le plaçant à droite de PlayButton. Renommez le nouveau bouton "ResetButton" et définissez sa propriété Text sur "Reset".

La vue Designer devrait ressembler à la figure 9-9.

158 Chapitre 9 : Xylophone

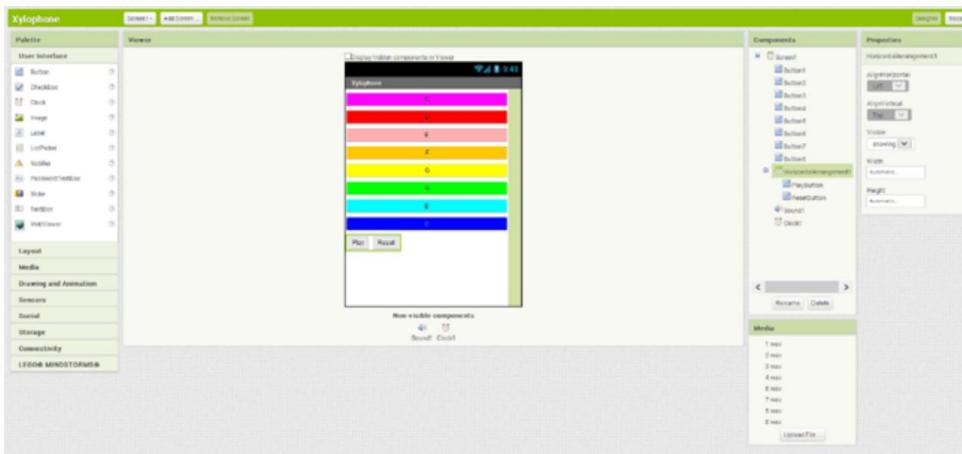


Figure 9-10. Ajout de composants pour l'enregistrement et la lecture des sons

NOTES ET HORAIRES D'ENREGISTREMENT

Nous devons maintenant ajouter le comportement correct dans l'éditeur de blocs. Nous devrons conserver des listes de notes et d'heures et ajouter des éléments aux listes chaque fois que l'utilisateur appuie sur un bouton.

1. Créez une nouvelle variable en accédant au tiroir Variables et en faisant glisser un global d'initialisation à bloquer depuis le tiroir Définition.
2. Changez le nom de la variable en « notes ».
3. Ouvrez le tiroir Listes et faites glisser un bloc de création de liste vide vers l'extérieur, en le plaçant dans le socket du global d'initialisation à bloquer.

Cela définit une nouvelle variable nommée « notes » comme étant une liste vide. Répétez les étapes pour une autre variable, que vous devez nommer « fois ». Ces nouveaux blocs devraient ressembler à ceux de la figure 9-10.



Figure 9-11. Initialisez deux variables pour stocker les notes et les informations de timing

Comment fonctionnent les blocs

Chaque fois qu'une note est jouée, nous devons enregistrer à la fois le nom du fichier sonore (dans la liste des notes) et l'instant auquel elle a été jouée (dans la liste des heures). Enregistrer

l'instant dans le temps, nous utiliserons le bloc Clock1.Now , qui renvoie l'instant dans le temps actuel (par exemple, le 12 mars 2011, 8:33:14), à la milliseconde la plus proche. Ces valeurs, obtenues via les blocs Sound1.Source et Clock1.Now , doivent être ajoutées respectivement aux listes de notes et d'heures , comme le montre la figure 9-11.

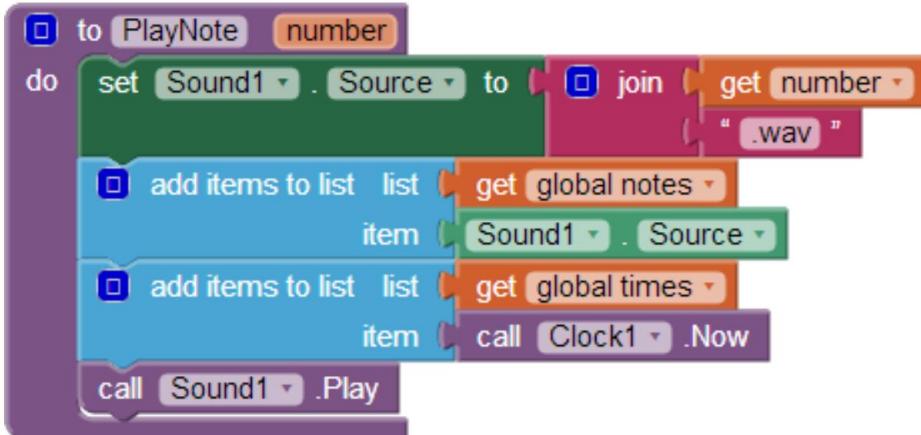


Figure 9-12. Ajout des sons joués à la liste

160 Chapitre 9 : Xylophone

Par exemple, si vous jouez à « Row, Row, Row Your Boat » [CCCDE], vos listes finiront par contenir cinq entrées, qui pourraient apparaître comme suit :

- remarques : 1.wav, 1.wav, 1.wav, 2.wav, 3.wav
- heures [dates omises] : 12:00:01, 12:00:02, 12:00:03, 12:00:03.5, 12:00:04

Lorsque l'utilisateur appuie sur le bouton Réinitialiser, nous souhaitons que les deux listes reviennent à leur état vide d'origine. Étant donné que l'utilisateur ne verra aucun changement, il est intéressant d'ajouter un petit bloc Sound1.Vibrate pour indiquer que le clic sur la touche a été enregistré. La figure 9-12 montre les blocs pour ce comportement.

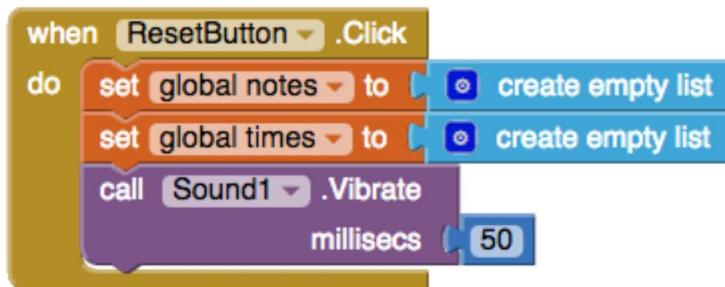


Figure 9-13. Fournir des commentaires lorsque l'utilisateur réinitialise l'application

LECTURE DES NOTES

À titre d'expérience de réflexion, voyons d'abord comment implémenter la lecture de notes sans nous soucier du timing. Nous pourrions (mais nous ne le ferons pas) le faire en créant ces blocs comme le montre la figure 9-13 :

- Un nombre variable pour garder une trace de la note sur laquelle nous sommes.
- Une nouvelle procédure, PlayBackNote, qui joue cette note et passe au prochain.
- Code à exécuter lorsque PlayButton est enfoncé, qui définit le compte sur 1 et appelle PlayBackNote sauf s'il n'y a pas de notes enregistrées.

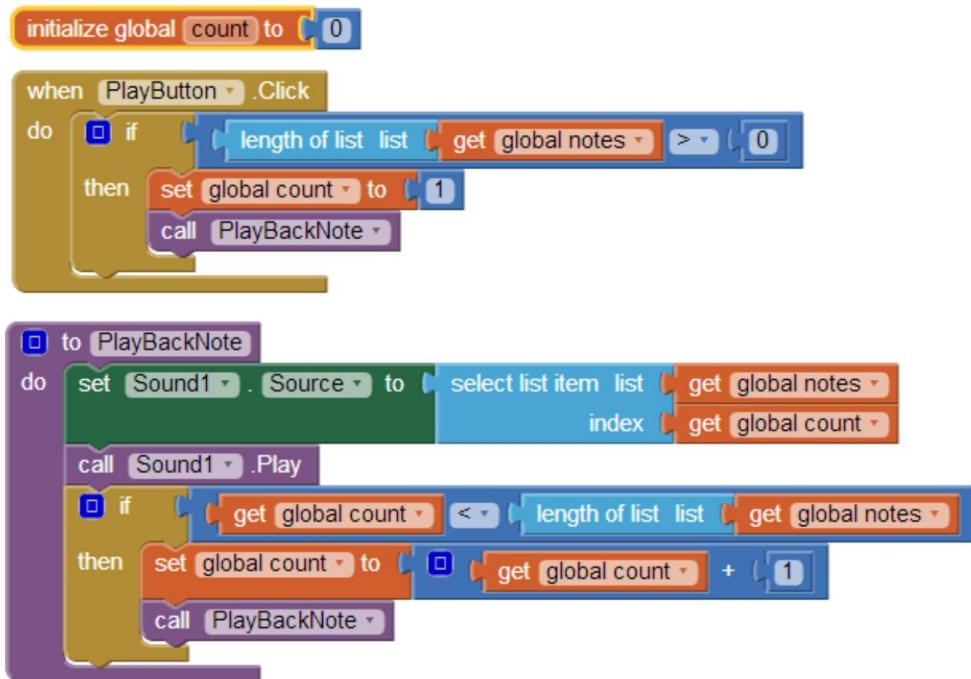


Figure 9-14. Lecture des notes enregistrées

Comment fonctionnent les blocs

C'est peut-être la première fois que vous voyez une procédure s'appeler elle-même. Même si à première vue cela peut paraître faux, il s'agit en fait d'un concept informatique important et puissant appelé récursion.

Pour avoir une meilleure idée du fonctionnement de la récursivité, voyons ce qui se passe si un utilisateur joue/enregistre trois notes (1.wav, 3.wav et 6.wav) puis appuie sur le bouton Lecture. Tout d'abord, PlayButton.Click commence à s'exécuter. Étant donné que la longueur des notes de la liste est de 3, ce qui est supérieur à 0, le nombre est défini sur 1 et PlayBackNote est appelé :

1. La première fois que PlayBackNote est appelé, compte = 1 :
 - Sound1.Source est défini sur le premier élément des notes, qui est 1.wav.
 - Sound1.Play est appelé et joue cette note.
 - Étant donné que le compte (1) est inférieur à la longueur des notes (3), le compte est incrémenté à 2, et PlayBackNote est à nouveau appelé.

2. La deuxième fois que PlayBackNote est appelé, count = 2 :
 - Sound1.Source est défini sur le deuxième élément des notes, qui est 3.wav.
 - Sound1.Play est appelé et joue cette note.

162 Chapitre 9 : Xylophone

- Étant donné que le compte (2) est inférieur à la longueur des notes (3), le compte est incrémenté à 3, et PlayBackNote est à nouveau appelé.

3. La troisième fois que PlayBackNote est appelé, compte = 3 :

- Sound1.Source est défini sur le troisième élément des notes, qui est 6.wav.
- Sound1.Play est appelé et joue cette note.
- Parce que le compte (3) n'est pas inférieur à la longueur des notes (3), rien d'autre se produit et la lecture est terminée.



Remarque Bien que la récursivité soit puissante, elle peut également être dangereuse. À titre d'expérience de réflexion, demandez-vous ce qui se serait passé si le programmeur avait oublié d'insérer les blocs dans PlayBackNote qui incrémentaient le nombre.

Bien que la récursion soit correcte, il existe un problème différent avec le précédent exemple : presque aucun temps ne s'écoule entre un appel à Sound1.Play et le suivant, donc chaque note est interrompue par la note suivante, sauf la dernière. Aucune note (sauf la dernière) n'est autorisée à se terminer avant que la source de Sound1 ne soit modifiée et que Sound1.Play ne soit à nouveau appelé. Pour obtenir le comportement correct, nous devons implémenter un délai entre les appels à PlayBackNote.

LECTURE DES NOTES AVEC DES RETARDS APPROPRIÉS

Nous mettrons en œuvre le délai en réglant la minuterie de l'horloge sur le temps écoulé entre la note actuelle et la note suivante. Par exemple, si la note suivante est jouée 3 000 millisecondes (3 secondes) après la note actuelle, nous définirons Clock1.TimerInterval sur 3 000, après quoi PlayBackNote devra être rappelé.

Apportez les modifications indiquées dans la figure 9-14 au corps du bloc if dans PlayBackNote et créez et remplissez le gestionnaire d'événements Clock1.Timer , qui spécifie ce qui doit se passer lorsque le minuteur se déclenche.

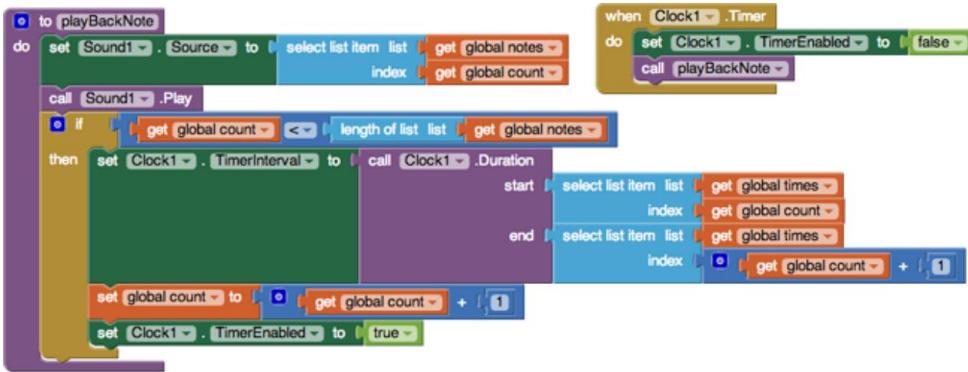


Figure 9-15. Ajout de délais entre les notes

Comment fonctionnent les blocs

Supposons le contenu suivant pour les deux listes :

- remarques : 1.wav, 3.wav, 6.wav
- horaires : 12:00:00, 12:00:01, 12:00:04

Comme le montre la figure 9-14, PlayButton.Click définit le compte sur 1 et appelle PlayBackNote.

1. La première fois que PlayBackNote est appelé, compte = 1 :

- Sound1.Source est défini sur le premier élément des notes, qui est « 1.wav ».
- Sound1.Play est appelé et joue cette note.
- Étant donné que le nombre (1) est inférieur à la durée des notes (3), Clock1.TimerInterval est défini sur la durée entre le premier (12:00:00) et le deuxième élément en heures (12:00:01) : 1 seconde. le compte est incrémenté à 2. Clock1.Timer est activé et commence le compte à rebours.

Rien d'autre ne se passe pendant 1 seconde, moment auquel Clock1.Timer s'exécute, désactiver temporairement la minuterie et appeler PlayBackNote.

2. La deuxième fois que PlayBackNote est appelé, count = 2 :

- Sound1.Source est défini sur le deuxième élément des notes, qui est « 3.wav ».
- Sound1.Play est appelé et joue cette note.
- Étant donné que le nombre (2) est inférieur à la durée des notes (3), Clock1.TimerInterval est défini sur la durée entre le deuxième (12:00:01) et le troisième élément en heures (12:00:04) : 3 secondes. le compte est incrémenté à 3. Clock1.Timer est activé et commence le compte à rebours.

164 Chapitre 9 : Xylophone

Rien d'autre ne se passe pendant 3 secondes, moment auquel Clock1.Timer s'exécute, désactiver temporairement la minuterie et appeler PlayBackNote.

3. La troisième fois que PlayBackNote est appelé, compte = 3 :

- Sound1.Source est défini sur le troisième élément des notes, qui est « 6.wav ».
- Sound1.Play est appelé et joue cette note.
- Étant donné que le compte (3) n'est pas inférieur à la longueur des notes (3), rien d'autre ne se produit. La lecture est terminée.

L'application complète : Xylophone

La figure 9-15 montre la configuration finale du bloc pour l'application Xylophone.

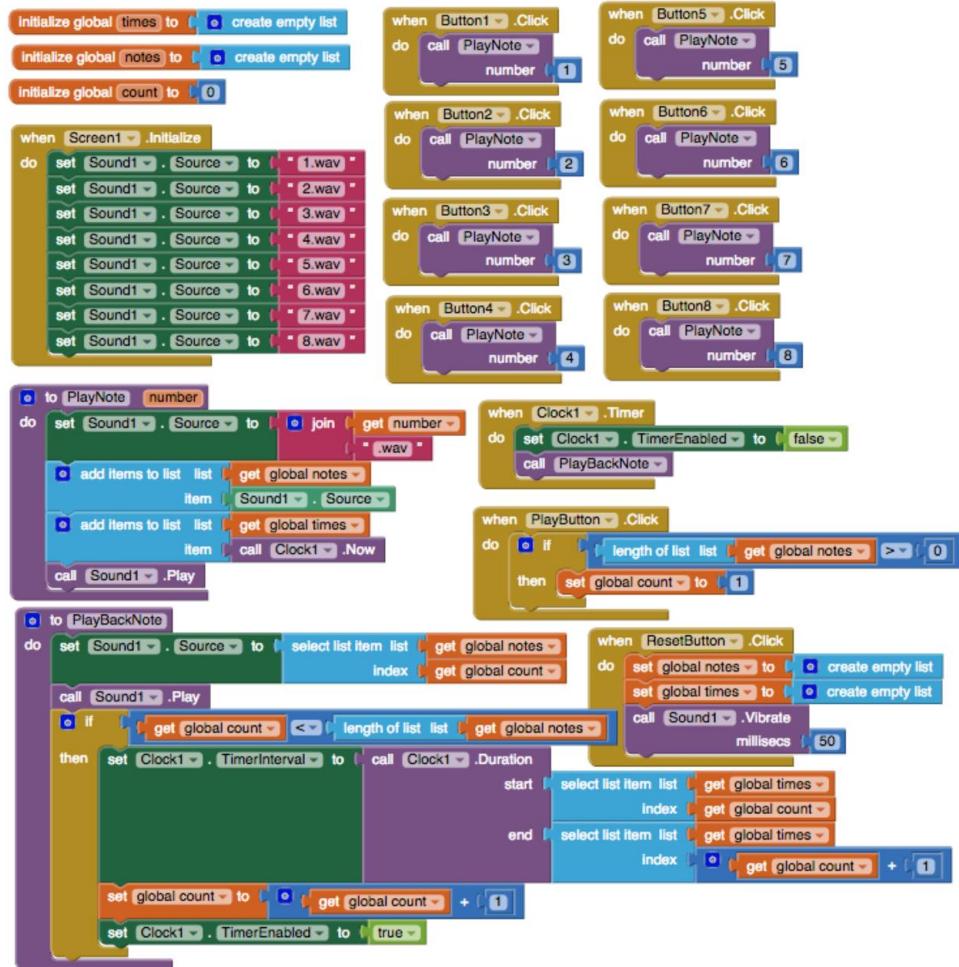


Figure 9-16. Les blocs pour Xylophone

Variantes

Voici quelques scénarios alternatifs à explorer :

- Actuellement, rien n'empêche un utilisateur de cliquer sur ResetButton pendant la lecture, ce qui entraînerait le blocage du programme. (Peux-tu comprendre pourquoi?)
Modifiez PlayButton.Click pour qu'il désactive ResetButton. Pour le réactiver une fois la chanson terminée, modifiez le bloc if dans PlayButton. Cliquez sur un bloc if else et réactivez ResetButton dans la partie else .
- De même, l'utilisateur peut actuellement cliquer sur PlayButton alors qu'une chanson est déjà jouant. (Pouvez-vous comprendre ce qui va se passer ?) Faites en sorte que PlayButton.Click

désactive PlayButton et change son texte en "Playing..." Vous pouvez le réactiver et réinitialiser le texte dans un bloc ifelse , comme décrit dans la puce précédente.

- Ajoutez un bouton avec le nom d'une chanson, comme « Für Elise ». Si l'utilisateur clique dessus, remplissez les listes de notes et de temps avec les valeurs correspondantes, définissez le nombre sur 1 et appelez PlayBackNote. Pour définir les heures appropriées, le bloc Clock1.MakeInstantFromMillis vous sera utile.
- Si l'utilisateur appuie sur une note, s'en va et fait autre chose, puis revient des heures plus tard et appuie sur une note supplémentaire, les notes feront partie de la même chanson, ce qui n'est probablement pas ce que l'utilisateur voulait. Améliorez le programme en 1) arrêtant l'enregistrement après un intervalle de temps raisonnable, comme une minute ; ou, 2) limiter la durée utilisée pour Clock1.TimerInterval en utilisant le bloc max du tiroir Math.
- Indiquez visuellement quelle note est jouée en modifiant l'apparence du bouton, par exemple en modifiant son Text, BackgroundColor ou ForegroundColor.

Résumé

Voici quelques-unes des idées que nous avons abordées dans ce tutoriel :

- Vous pouvez lire différents fichiers audio à partir d'un seul composant Sound en modifiant sa propriété Source . Cela nous a permis d'avoir un composant sonore au lieu de huit. Assurez-vous simplement de charger les sons lors de l'initialisation pour éviter les retards (Figure 9-6).
- Les listes peuvent fournir à un programme une mémoire, avec un enregistrement des actions de l'utilisateur stockées dans la liste et ensuite récupérées et retraitées. Nous avons utilisé cette fonctionnalité pour enregistrer et lire une chanson.
- Vous pouvez utiliser le composant Clock pour déterminer l'heure actuelle. Soustraire deux valeurs temporelles nous donnent le temps entre deux événements.
- Vous pouvez définir la propriété TimerInterval pour Clock dans le programme, par exemple en définissant la durée entre le début de deux notes.
- Il est non seulement possible mais parfois souhaitable qu'une procédure s'appelle elle-même. Il s'agit d'une technique puissante appelée récursivité. Lorsque vous écrivez une procédure récursive, assurez-vous qu'il existe un cas de base dans lequel la procédure se termine, plutôt que de s'appeler elle-même, sinon le programme bouclera à l'infini.

MakeQuiz et TakeQuiz

Vous pouvez personnaliser l'application Presidents Quiz au chapitre 8 pour créer n'importe quel quiz, mais seul le programmeur peut modifier les questions et les réponses. Les parents, les enseignants ou les autres utilisateurs de l'application n'ont aucun moyen de créer leurs propres quiz ou de modifier les questions du quiz (à moins qu'ils ne souhaitent eux aussi apprendre à utiliser App Inventor !).

Dans ce chapitre, vous allez créer une application MakeQuiz qui permet un « enseignant » créé des quiz à l'aide d'un formulaire de saisie. Les questions et réponses du quiz seront stockées dans une base de données Web afin que les « étudiants » puissent accéder à une application TakeQuiz distincte et passer le test. En créant ces deux applications, vous ferez un autre saut conceptuel important : apprendre à créer des applications avec des données générées par les utilisateurs et partagées entre les applications et

utilisateurs.

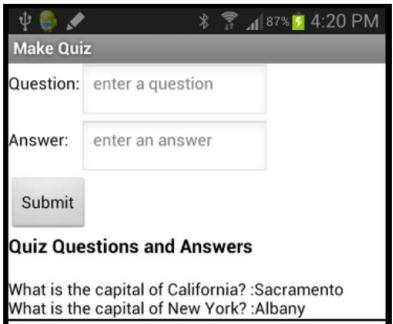


Figure 10-2. L'application MakeQuiz en action

Quiz des présidents.

Voici les comportements que vous coderez pour la première application, MakeQuiz :

- L'utilisateur saisit des questions et des réponses dans un formulaire de saisie.
- Les couples question-réponse s'affichent.
- Les questions et réponses du quiz sont stockées dans une base de données Web.

Figure 10-1.



Les parents peuvent créer des applications de quiz amusantes pour leurs enfants lors d'un long voyage en voiture, les enseignants des écoles primaires peuvent créer des quiz « Math Blaster » et les étudiants peuvent créer des quiz pour aider leurs groupes d'étude à se préparer à une finale. Ce chapitre s'appuie sur le Quiz des présidents du chapitre 8, donc si vous n'avez pas terminé cette application, vous devriez le faire avant de continuer ici.

Vous allez concevoir deux applications, MakeQuiz pour l'enseignant (voir Figure 10-1) et TakeQuiz pour l'élève, qui ressembleront à l'application

168 Chapitre 10 : MakeQuiz et TakeQuiz

La deuxième application que vous allez créer, TakeQuiz, fonctionnera de la même manière que l'application Presidents Quiz que vous avez déjà créée. En fait, vous utiliserez l'application Presidents Quiz comme point de départ. TakeQuiz différera dans le sens où les questions posées seront celles qui ont été saisies dans la base de données via MakeQuiz.

Ce que vous apprenez

Le Quiz des Présidents est un exemple d'application avec des données statiques : peu importe le nombre de fois que vous répondez au quiz, les questions sont toujours les mêmes car elles sont codées en dur dans l'application ; c'est-à-dire que les questions et réponses font partie des blocs. Les applications d'actualités, les blogs et les applications de réseaux sociaux telles que Facebook et Twitter fonctionnent avec des données dynamiques, ce qui signifie que les données peuvent changer au fil du temps. Souvent, ces informations dynamiques sont générées par l'utilisateur : l'application permet aux utilisateurs de saisir, de modifier et de partager des informations. Avec MakeQuiz et TakeQuiz, vous apprendrez à créer une application qui gère les données partagées générées par les utilisateurs.

Si vous avez terminé l'application Xylophone (Chapitre 9), vous avez déjà été présenté aux listes dynamiques ; dans cette application, les notes de musique jouées par l'utilisateur sont enregistrées dans des listes. Les applications avec de telles données générées par l'utilisateur sont plus complexes et les blocs sont plus abstraits car ils ne reposent pas sur des données statiques prédéfinies. Vous définissez des variables de liste, mais vous les définissez sans éléments spécifiques. Lorsque vous programmez votre application, vous devez imaginer que les listes soient remplies avec les données fournies par l'utilisateur final.

Ce didacticiel couvre les concepts App Inventor suivants :

- Formulaires de saisie permettant à l'utilisateur de saisir des informations.
- Utiliser une liste indexée avec pour chacun d'afficher les éléments de plusieurs listes.
- Données de liste persistantes : MakeQuiz enregistrera les questions et réponses du quiz dans une base de données Web et TakeQuiz les chargera à partir de la même base de données.
- Partage de données : vous stockerez les données dans une base de données Web à l'aide du composant TinyWebDB (au lieu du composant TinyDB utilisé dans les chapitres précédents).

Commencer

Connectez-vous au site Web App Inventor et démarrez un nouveau projet. Nommez-le « MakeQuiz » et définissez le titre de l'écran sur « Make Quiz ». Connectez votre application à votre appareil ou émulateur pour des tests en direct.

Conception des composants

Utilisez le Concepteur de composants pour créer l'interface de MakeQuiz. Quand tu as fini, c'est devrait ressembler à la Figure 10-2 (il y a aussi des instructions plus détaillées après l'instantané).

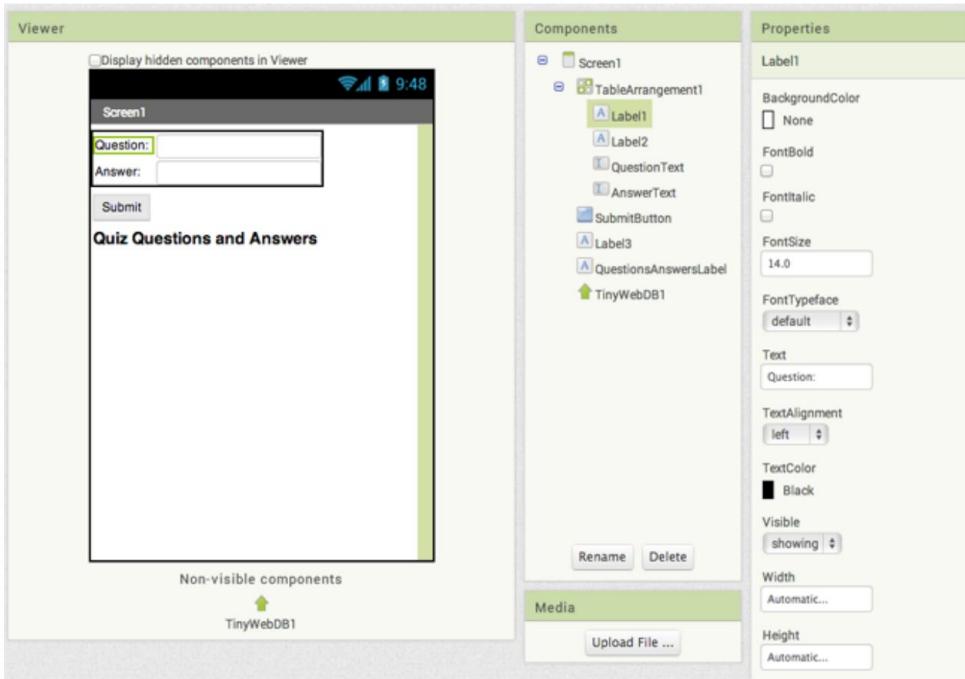


Figure 10-3. MakeQuiz dans le concepteur de composants

Vous pouvez créer l'interface utilisateur illustrée à la figure 10-2 en faisant glisser le composants répertoriés dans le tableau 10-1. Faites glisser chaque composant de la palette vers le Viewer et nommez-le comme spécifié dans le tableau. Notez que vous pouvez laisser le libellé d'en-tête noms (Label1 – Label4) comme valeurs par défaut (vous ne les utiliserez pas dans l'éditeur de blocs de toute façon).

Tableau 10-1. Tous les composants de l'application MakeQuiz

Type de composant	Groupe de palettes	Comment vous l'appellerez	But
Disposition de l'arrangement de table		TableArrangement1	Formatize le formulaire, y compris la question et répondre.
Étiquette	Étiquette de l'interface utilisateur	Label1	L'invite « Question : ».
Zone de texte	Texte de la question	AnswerText	L'utilisateur saisit des questions ici.

170 Chapitre 10 : MakeQuiz et TakeQuiz

Type de composant	Groupe de palettes	Comment vous l'appellerez	But
Étiquette	Étiquette de l'interface	utilisateur2	L'invite « Réponse : ».
Zone de texte	Texte de réponse de l'interface utilisateur		L'utilisateur saisit les réponses ici.
Bouton	Bouton de soumission de l'interface utilisateur		L'utilisateur clique dessus pour soumettre une paire d'assurance qualité.
Étiquette	Étiquette de l'interface	utilisateur3	Affichez « Questions et réponses du quiz ».
Étiquette	Interface utilisateur QuestionsAnswersLabel	Afficher les paires QA précédemment saisies.	
MinusculeWebDB	Stockage	MinusculeWebDB1	Stockage Web pour les paires QA.

Définissez les propriétés des composants de la manière suivante :

1. Définissez le texte de Label1 sur « Question », le texte de Label2 sur « Réponse » et le texte de Label3 à « Questions et réponses du quiz ».
2. Définissez la taille de police de Label3 sur 18 et cochez la case FontBold.
3. Définissez l'indice de QuestionText sur « Entrer une question » et l'indice de AnswerText à « Entrer une réponse ».
4. Définissez le texte du bouton Submit sur « Soumettre ».
5. Définissez le texte de QuestionsAnswersLabel sur « Questions et réponses du quiz ».
6. Déplacez le QuestionText, le AnswerText et leurs étiquettes associées dans TableArrangement1.

Si vous regardez les propriétés de TinyWebDB, vous remarquerez qu'il a une propriété ServiceURL (voir Figure 10-3). Cette propriété spécifie un service de base de données Web, spécialement configuré pour fonctionner avec le composant TinyWebDB , où vos données partagées seront stocké. Par défaut, le service Web auquel il fait référence est celui configuré par le MIT App Inventor. équipe sur <http://appinvintinywebdb.appspot.com>. Vous utiliserez ce service par défaut dans ce tutoriel pendant que vous travaillez ; cependant, il est important de savoir que toute personne utilisant App Inventor stockera des informations sur ce même service Web et que les données que votre application met il sera vu par tous et pourrait même être écrasé par quelqu'un.

Le service par défaut est uniquement destiné aux tests. Il est assez simple (et gratuit !) de configurer votre posséder un tel service, ce que vous souhaitez faire si vous créez une application qui sera déployée avec de vrais utilisateurs. Pour l'instant, continuez et terminez ce didacticiel, mais lorsque vous aurez terminé prêt, les instructions pour configurer votre propre service Web se trouvent sur « TinyWebDB et API compatibles TinyWebDB » à la page 368.



Figure 10-4. Avec TinyWebDB.ServiceURL, vous pouvez spécifier l'URL d'une base de données Web que vous avez configurée

Ajout de comportements aux composants

Comme avec l'application Presidents Quiz, vous définirez d'abord quelques variables globales pour le QuestionList et AnswerList, mais cette fois vous ne fournirez pas de questions fixes et réponses.

CRÉATION DE LISTES DE QUESTIONS ET RÉPONSES VIDES

Les blocs des listes doivent ressembler à celui illustré à la figure 10-4.



Figure 10-5. Les listes de MakeQuiz commencent vides

Les listes sont définies avec le bloc créer une liste vide , au lieu du bloc créer une liste . En effet, avec les applications MakeQuiz et TakeQuiz, toutes les données seront créées par l'utilisateur de l'application (il s'agit de données dynamiques générées par l'utilisateur).

ENREGISTREMENT DES ENTRÉES DE L'UTILISATEUR

Le premier comportement que vous créerez concerne la gestion des entrées de l'utilisateur. Plus précisément, lorsque l'utilisateur saisit une question et une réponse et clique sur Soumettre, vous utiliserez l'ajout d'éléments aux blocs de liste pour mettre à jour la QuestionList et la AnswerList. Les blocs doivent apparaître comme illustré dans la figure 10-5.

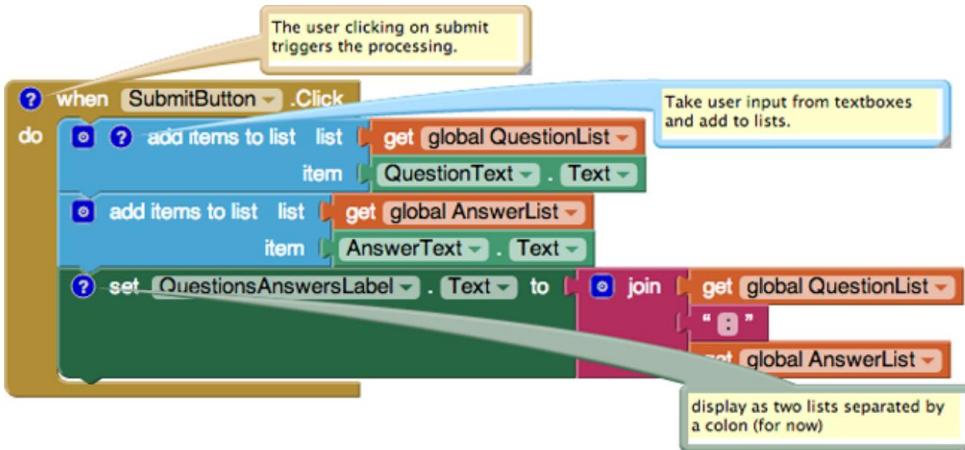


Figure 10-6. Ajouter de nouvelles entrées aux listes

Comment fonctionnent les blocs

Le bloc Ajouter des éléments à la liste ajoute chaque élément à la fin d'une liste. Comme le montre la figure 10-5, l'application prend le texte que l'utilisateur a saisi dans les zones de texte QuestionText et AnswerText et l'ajoute à la liste correspondante.

Les blocs d'ajout d'éléments à la liste mettent à jour les variables QuestionList et AnswerList , mais ces modifications ne sont pas encore affichées à l'utilisateur. La troisième rangée de blocs affiche ces listes en les concaténant (en les joignant) avec deux points insérés entre elles. Par défaut, App Inventor affiche les listes entourées de parenthèses et d'espaces entre les éléments : par exemple, "(élément1 élément2 élément3)". Bien entendu, ce n'est pas la manière idéale d'afficher les listes, mais cela vous permettra pour l'instant de tester le comportement de l'application. Plus tard, vous créerez une méthode plus sophistiquée d'affichage des listes qui affiche chaque paire question-réponse sur une ligne distincte.

SUPPRIMER LA QUESTION ET LA RÉPONSE

Rappelez-vous de l'application Presidents Quiz que lorsque vous passiez à la question suivante de la liste, vous deviez effacer la réponse de l'utilisateur à la question précédente. Dans cette application, lorsqu'un utilisateur soumet une paire question-réponse, vous souhaiterez effacer les zones de texte QuestionText et AnswerText afin qu'elles soient prêtes pour une nouvelle entrée au lieu d'afficher la précédente. Les blocs doivent apparaître comme ceux illustrés à la figure 10-6.

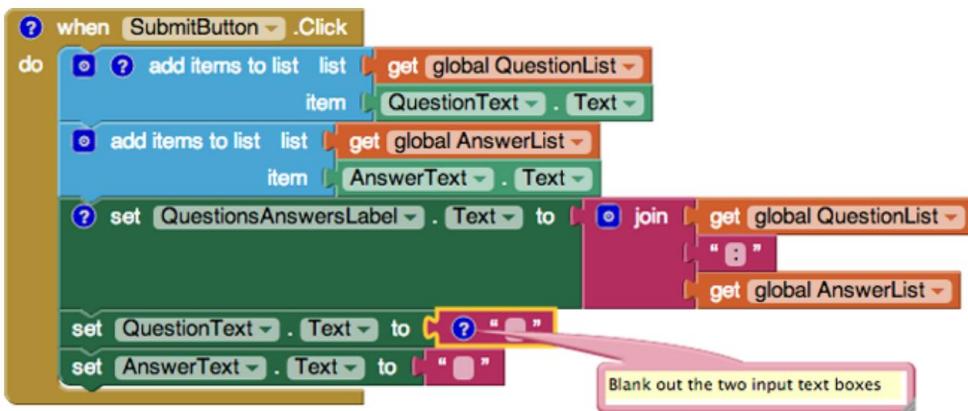


Figure 10-7. Suppression des zones de texte des questions et réponses après la soumission



Testez votre application Testez le comportement en saisissant quelques paires de questions-réponses. Au fur et à mesure que vous les ajoutez, apparaissent-ils sous le formulaire dans QuestionsAnswersLabel ?

Comment fonctionnent les blocs

Lorsque l'utilisateur soumet une nouvelle question et réponse, elles sont ajoutées à leurs listes respectives et affichées. À ce stade, le texte de QuestionText et AnswerText est masqué par des blocs de texte vides.

AFFICHAGE DES PAIRES QUESTION-RÉPONSE SUR PLUSIEURS LIGNES

Dans l'application que vous avez créée jusqu'à présent, les listes de questions et de réponses sont affichées séparément et en utilisant le format d'affichage de liste par défaut pour App Inventor. Donc, si vous faisiez un quiz sur les capitales des États et que vous aviez saisi deux paires de questions et réponses, cela pourrait être apparaître comme :

(Quelle est la capitale de la Californie ? Quelle est la capitale de New York ? : Sacramento Albany)

Ce n'est évidemment pas une interface utilisateur idéale pour le concepteur de quiz. Un meilleur affichage montrerait chaque question avec sa réponse correspondante, avec une paire question-réponse par ligne, comme ceci :

Quelle est la capitale de la Californie ? : Sacramento
Quelle est la capitale de New York ? : Albany La

technique permettant d'afficher une seule liste avec chaque élément sur une ligne distincte est décrite au chapitre 20 ; vous souhaiterez peut-être lire ce chapitre avant de continuer.

La tâche ici est un peu plus compliquée car vous avez affaire à deux listes.

En raison de sa complexité, vous placerez les blocs d'affichage des données dans une procédure

174 Chapitre 10 : MakeQuiz et TakeQuiz

nommé displayQAs et appelez cette procédure à partir du gestionnaire d'événements SubmitButton.Click .

Pour afficher les paires question-réponse sur des lignes distinctes, vous devrez procéder comme suit :

- Utilisez un pour chaque bloc pour parcourir chaque question de la QuestionList.
- Utilisez une variable AnswerIndex afin de pouvoir saisir chaque réponse au fur et à mesure que vous parcourez les questions.
- Utilisez join pour créer un objet texte avec chaque paire question/réponse, ainsi qu'un caractère de nouvelle ligne (\n) séparant chaque paire.

Les blocs doivent apparaître comme illustré dans la figure 10-7.



Figure 10-8. La procédure displayQAs

Comment fonctionnent les blocs

La procédure displayQAs encapsule tous les blocs d'affichage des données.

En utilisant une procédure, vous n'aurez pas à copier les blocs nécessaires à l'affichage de la liste plus d'une fois dans l'application : vous pouvez simplement appeler displayQAs lorsque vous devez afficher les listes.

Le for each vous permet uniquement de parcourir une seule liste. Dans ce cas, il y a deux listes, QuestionList et AnswerList. Le for each est utilisé pour parcourir la QuestionList, mais vous devez également sélectionner une réponse au fur et à mesure que vous avancez dans les questions. Pour ce faire, vous utilisez une variable d'index, comme cela a été fait avec currentQuestionIndex dans le didacticiel Presidents Quiz du chapitre 8. Dans ce cas, la variable d'index, answerIndex, est utilisée pour suivre la position dans la AnswerList au fur et à mesure que chaque étape passe. la liste de questions.

AnswerIndex est défini sur 1 avant le début de for each . Dans chaque cas, AnswerIndex est utilisé pour sélectionner la réponse actuelle dans AnswerList , puis il est incrémenté. À chaque itération de for each , la question et la réponse actuelles sont concaténées à la fin de la propriété QuestionsAnswersLabel.Text , avec deux points entre elles.

APPEL DE LA PROCÉDURE DISPLAYQAS

Vous disposez désormais d'une procédure pour afficher les paires question-réponse, mais cela ne vous aidera que si vous lappelez lorsque vous en avez besoin. Modifiez le gestionnaire d'événements SubmitButton.Click en appelant displayQAs au lieu d'afficher les listes, comme cela a été fait précédemment. Les blocs mis à jour doivent apparaître comme indiqué dans la figure 10-8.

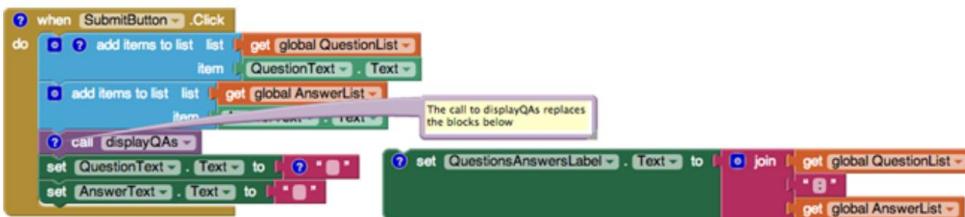


Figure 10-9. Appel de la procédure displayQAs pour remplacer les blocs affichés à droite



Testez votre application Testez le comportement en saisissant quelques paires de questions-réponses. Au fur et à mesure que vous les ajoutez, apparaissent-ils sur des lignes distinctes dans QuestionsAnswersLabel ?

SAUVEGARDER LE QAS DE MANIÈRE PERSISTANTE SUR LE WEB

Jusqu'à présent, vous avez créé une application qui place les questions et réponses saisies dans une liste. Mais que se passe-t-il si le créateur de quiz ferme l'application ? Si vous avez terminé l'application Pas d'envoi de SMS pendant la conduite (Chapitre 4) ou l'application Android, Où est ma voiture ? app (Chapitre 7), vous savez que si vous ne stockez pas les données dans une base de données, elles ne seront pas là lorsque l'utilisateur quittera et redémarrera l'application. Le stockage persistant des données permettra au créateur du quiz d'afficher ou de modifier la dernière mise à jour du quiz à chaque démarrage de l'application. Un stockage persistant est également nécessaire car l'application TakeQuiz a également besoin d'accéder aux données.

Vous êtes déjà familiarisé avec l'utilisation du composant TinyDB pour stocker et récupérer des données dans une base de données. Mais dans ce cas, vous utiliserez plutôt le composant TinyWebDB . Alors que TinyDB stocke les informations directement sur un téléphone, TinyWebDB stocke les données dans des bases de données résidant sur le Web.

Qu'en est-il de la conception de votre application qui mériterait d'utiliser une base de données en ligne au lieu d'une base de données stockée sur le téléphone d'une personne ? Le problème clé ici est que vous créez deux applications qui

176 Chapitre 10 : MakeQuiz et TakeQuiz

les deux doivent accéder aux mêmes données : si l'auteur du quiz stocke les questions et les réponses sur son téléphone, les participants au quiz n'auront aucun moyen d'accéder aux données de leur quiz ! Étant donné que TinyWebDB stocke les données sur le Web, le participant au quiz peut accéder aux questions et réponses du quiz sur un appareil différent de celui du créateur du quiz. (Le stockage de données en ligne est souvent appelé cloud.)

Voici le schéma général permettant de rendre les données de liste (telles que les questions et réponses de notre application) persistantes :

- Stockez une liste dans la base de données chaque fois qu'un nouvel élément y est ajouté.
- Lorsque l'application démarre, chargez la liste de la base de données dans une variable.

Commencez par stocker QuestionList et AnswerList dans la base de données chaque fois que l'utilisateur saisit une nouvelle paire.

Comment fonctionnent les blocs

Les blocs TinyWebDB1.StoreValue stockent les données dans une base de données Web. StoreValue a deux arguments : la balise qui identifie les données et la valeur qui correspond aux données réelles que vous souhaitez stocker. La figure 10-9 montre que la QuestionList est stockée avec une balise « questions », tandis que la AnswerList est stockée avec une balise « réponses ».

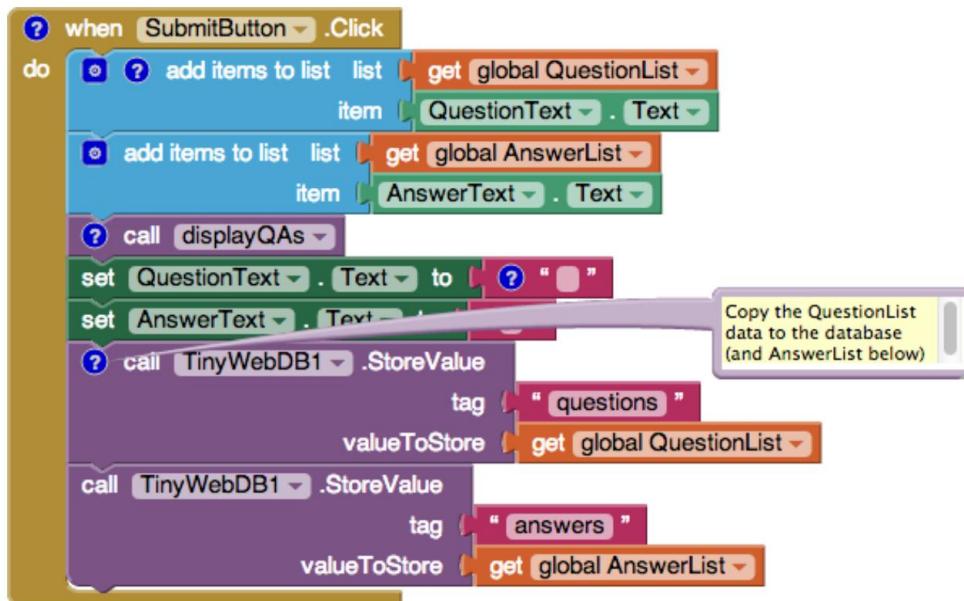


Figure 10-10. Stockage des questions et réponses dans la base de données

Pour votre application, vous devez utiliser des balises plus distinctives que « questions » et « réponses » (par exemple, « DavesQuestions » et « DavesAnswers »). Ceci est important car, à

Au moins au début, vous utilisez le service de base de données Web par défaut pour App Inventor, ce qui signifie que d'autres personnes peuvent écraser vos questions et réponses, y compris d'autres personnes qui suivent ce didacticiel.



Testez votre application Le test de cette partie de l'application est différent des tests que vous avez effectués précédemment car votre application affecte désormais une autre entité, le service TinyDBWeb par défaut. Exécutez l'application, saisissez une question et une réponse, puis ouvrez une fenêtre de navigateur sur le service Web par défaut à l'adresse <http://appinvtinywebdb.appspot.com>. Cliquez ensuite sur « get_value » et saisissez l'une de vos balises (dans cet exemple, « questions » ou « réponses »). Si tout fonctionne correctement, vos listes de questions et réponses devraient apparaître.

Comme mentionné précédemment, le service Web par défaut est partagé entre les programmeurs et les applications, il est donc destiné uniquement aux tests. Lorsque vous serez prêt à déployer votre application auprès de vrais utilisateurs, vous souhaiterez configurer votre propre service de base de données privée. Heureusement, cette opération est simple et ne nécessite aucune programmation (voir « API TinyWebDB et TinyWebDB-Compliant » à la page 368).

CHARGEMENT DES DONNÉES DEPUIS LA BASE DE DONNÉES

L'une des raisons pour lesquelles nous devons stocker les questions et réponses dans une base de données est de permettre à la personne qui crée le quiz de fermer l'application et de la relancer ultérieurement sans perdre les questions et réponses précédemment saisies. (Nous le faisons également pour que le participant au quiz puisse accéder aux questions, mais nous y reviendrons plus tard.) Programmons les blocs pour charger les listes dans l'application à partir de la base de données Web à chaque redémarrage de l'application.

Comme nous l'avons vu dans les chapitres précédents, pour spécifier ce qui doit se passer lorsqu'une application lance, vous programmez le gestionnaire d'événements Screen.Initialize . Dans ce cas, l'application doit demander deux listes à la base de données Web TinyWebDB (les questions et les réponses) afin que Screen1.Initialize effectue deux appels à TinyWebDB.GetValue. Les blocs doivent apparaître comme illustré dans la figure 10-10.

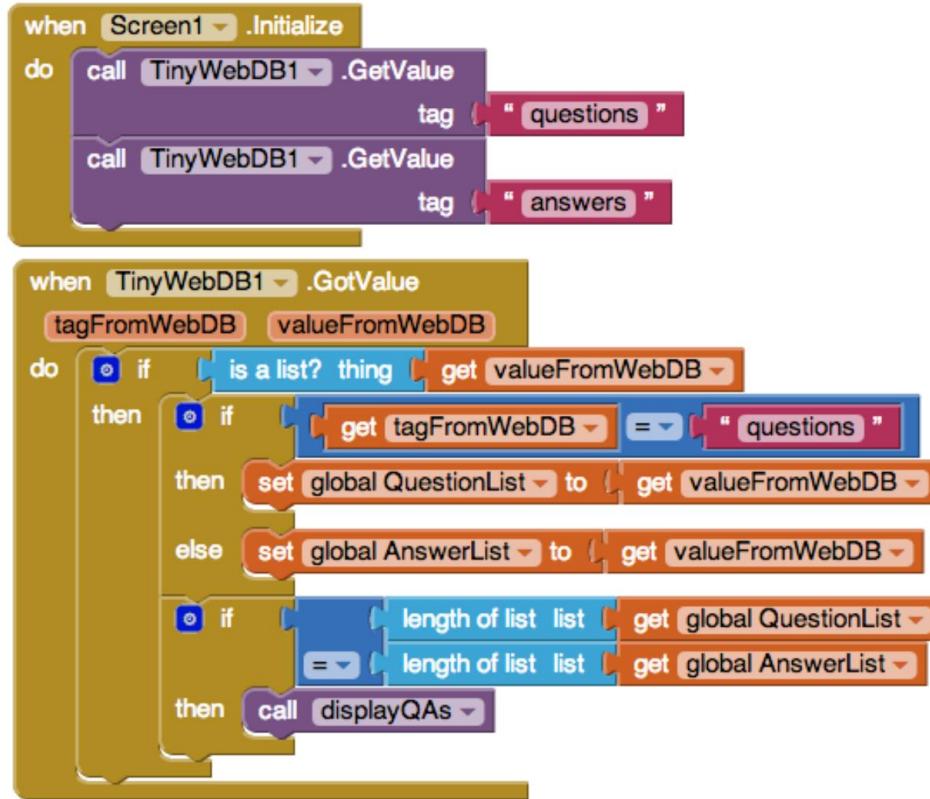


Figure 10-11. Demander les listes à la base de données lorsque l'application s'ouvre et traiter lorsque les listes arrivent

Comment fonctionnent les blocs

Les blocs TinyWebDB.GetValue de la figure 10-10 fonctionnent différemment de TinyDB.GetValue, qui renvoie une valeur immédiatement. TinyWebDB.GetValue demande uniquement les données de la base de données Web ; il ne reçoit pas immédiatement de valeur. Au lieu de cela, lorsque les données arrivent de la base de données Web, un événement TinyWebDB.GetValue est déclenché. Vous devez également programmer ce gestionnaire d'événements pour traiter les données renvoyées.

Lorsque l'événement TinyWebDB.GetValue se produit, les données demandées sont contenues dans un argument nommé valueFromWebDB. La balise que vous avez demandée est contenue dans l'argument tagFromWebDB.

Dans cette application, étant donné que deux requêtes différentes sont faites pour les questions et réponses, GetValue sera déclenché deux fois. Pour éviter de mettre des questions dans votre AnswerList, ou vice versa, votre application doit vérifier la balise pour voir quelle requête est arrivée, puis placer la valeur renvoyée par la base de données dans la liste correspondante (QuestionList ou AnswerList).

Dans Screen.Initialize, l'application appelle TinyWebDB1.GetValue deux fois : une fois pour demander la liste de questions stockée et une fois pour demander la liste de réponses stockée . Lorsque les données arrivent de la base de données Web à partir de l'une ou l'autre requête, l' événement TinyWebDB1.GetValue est déclenché.

L' argument valueFromWebDB de GetValue contient les données renvoyées par la requête de base de données. Vous avez besoin du bloc if externe dans le gestionnaire d'événements car la base de données renverra un texte vide (« ») dans valueFromWebDB si c'est la première fois que l'application est utilisée et qu'il n'y a pas encore de questions et de réponses. En demandant si valueFromWebDB est une liste ?, vous vous assurez que certaines données sont réellement renvoyées.
S'il n'y a aucune donnée, vous contournerez les blocs pour les traiter.

Si des données sont renvoyées (une liste est- elle vraie ?), les blocs vérifient quelle requête est arrivée. La balise identifiant les données est dans tagFromWebDB : ce sera soit « questions » soit « réponses ». Si la balise est « questions », la valeurFromWebDB est placée dans la variable QuestionList. Sinon (sinon), il est placé dans AnswerList. (Si vous avez utilisé des balises autres que « questions » et « réponses », vérifiez-les plutôt.)

Vous souhaitez afficher les listes uniquement une fois les deux arrivées (GetValue a été déclenché deux fois). Pouvez-vous imaginer comment vous sauriez avec certitude que les deux listes sont chargées à partir de la base de données ? Les blocs affichés utilisent un test if pour vérifier si le les longueurs des listes sont les mêmes, car cela ne peut être vrai que si les deux listes ont été renvoyées.
Si tel est le cas, la procédure pratique displayQAs que vous avez écrite précédemment est appelée pour afficher les données chargées.

L'application complète : MakeQuiz

La figure 10-11 montre les blocs pour l'ensemble de l'application MakeQuiz.

TakeQuiz : une application pour répondre au quiz dans le Base de données

Vous disposez désormais d'une application MakeQuiz qui stockera un quiz dans une base de données Web. Créer TakeQuiz, l'application qui charge dynamiquement le quiz, est plus simple. Vous pouvez le construire en apportant quelques modifications au Quiz des Présidents que vous avez complété au chapitre 8 (si vous n'avez pas terminé ce didacticiel, faites-le maintenant avant de continuer).

Commencez par ouvrir votre application Presidents Quiz dans App Inventor, en choisissant Enregistrer sous, puis nommer le nouveau projet « TakeQuiz ». Cela laissera votre application Presidents Quiz inchangée, mais vous pourrez désormais utiliser ses blocs comme base pour TakeQuiz.

180 Chapitre 10 : MakeQuiz et TakeQuiz

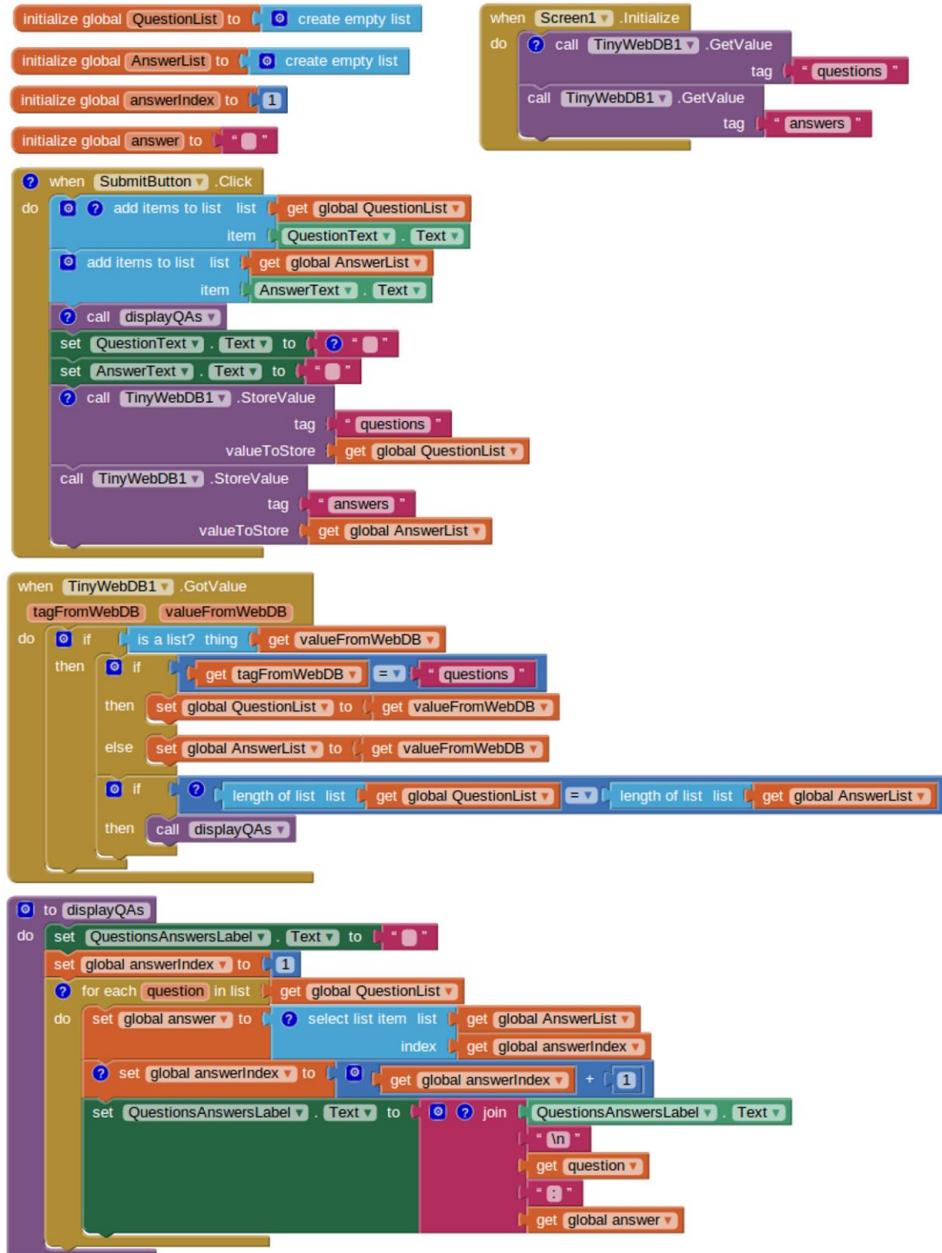


Figure 10-12. Les blocs pour MakeQuiz

Ensuite, apportez les modifications suivantes dans le Designer :

1. Cette version n'affichera pas d'images avec chaque question, supprimez donc d'abord les références aux images de l'application TakeQuiz. Dans le Concepteur de composants,

choisissez chaque image dans la palette Média et supprimez-la. Ensuite, supprimez le composant Image1 , ce qui supprimera toutes les références à celui-ci de l'éditeur de blocs.

2. Étant donné que TakeQuiz fonctionnera avec des données de base de données résidant sur le Web, faites glisser un composant TinyWebDB dans l'application.
3. Parce que vous ne voulez pas que l'utilisateur réponde ou clique sur le NextButton jusqu'à ce que les questions soient chargées, décochez la propriété Enabled du AnswerButton et Bouton Suivant.

Maintenant, modifiez les blocs pour que le quiz remis à l'utilisateur soit chargé depuis la base de données. Premièrement, comme il n'y a pas de questions et de réponses fixes, supprimez tous les blocs de texte de questions et de réponses réels des blocs de création d'une liste dans QuestionList et AnswerList. Les blocs résultants doivent apparaître comme indiqué dans la figure 10-12.



Figure 10-13. Les listes de questions et réponses commencent désormais vides

Vous pouvez également supprimer complètement la PictureList ; cette application ne traitera pas les images. Maintenant, modifiez votre Screen1.Initialize pour qu'il appelle TinyWebDB.GetValue deux fois pour charger les listes, comme vous l'avez fait dans MakeQuiz. Les blocs doivent ressembler à ceux de la figure 10-13.

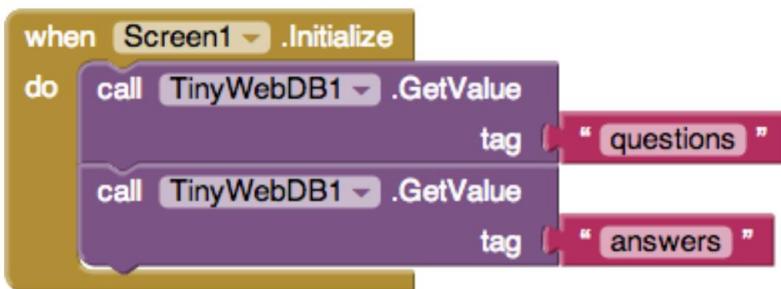


Figure 10-14. Demander les questions et réponses de la base de données Web

Enfin, faites glisser un gestionnaire d'événements TinyWebDB.GotValue . Ce gestionnaire d'événements devrait ressembler à celui utilisé dans MakeQuiz, mais ici vous souhaitez afficher uniquement la première question et aucune des réponses. Essayez d'abord d'effectuer ces modifications vous-même, puis examinez les blocs de la figure 10-14 pour voir s'ils correspondent à votre solution.

182 Chapitre 10 : MakeQuiz et TakeQuiz



Figure 10-15. GotValue gère les données qui arrivent du Web

COMMENT FONCTIONNENT LES BLOCS

Lorsque l'application démarre, Screen1.Initialize est déclenché et l'application demande les questions et réponses à la base de données Web. Lorsque chaque requête arrive, le gestionnaire d'événements TinyWebDB.GotValue est déclenché. L'application vérifie d'abord s'il y a effectivement des données dans valueFromWebDB en utilisant une liste ? Si elle trouve des données, l'application demande quelle requête est arrivée, à l'aide de tagFromWebDB, et place la valeur FromWebDB dans la liste appropriée. Si la QuestionList est en cours de chargement, la première question est sélectionnée dans QuestionList et affichée. Si AnswerList est en cours de chargement, AnswerButton et NextButton sont activés afin que l'utilisateur puisse commencer à passer le test.

Ce sont tous les changements dont vous avez besoin pour TakeQuiz. Si vous avez ajouté des questions et réponses avec MakeQuiz et que vous exécutez TakeQuiz, les questions qui apparaissent doivent être celles que vous avez saisies.

L'application complète : TakeQuiz

La figure 10-15 montre les blocs pour l'ensemble de l'application TakeQuiz.

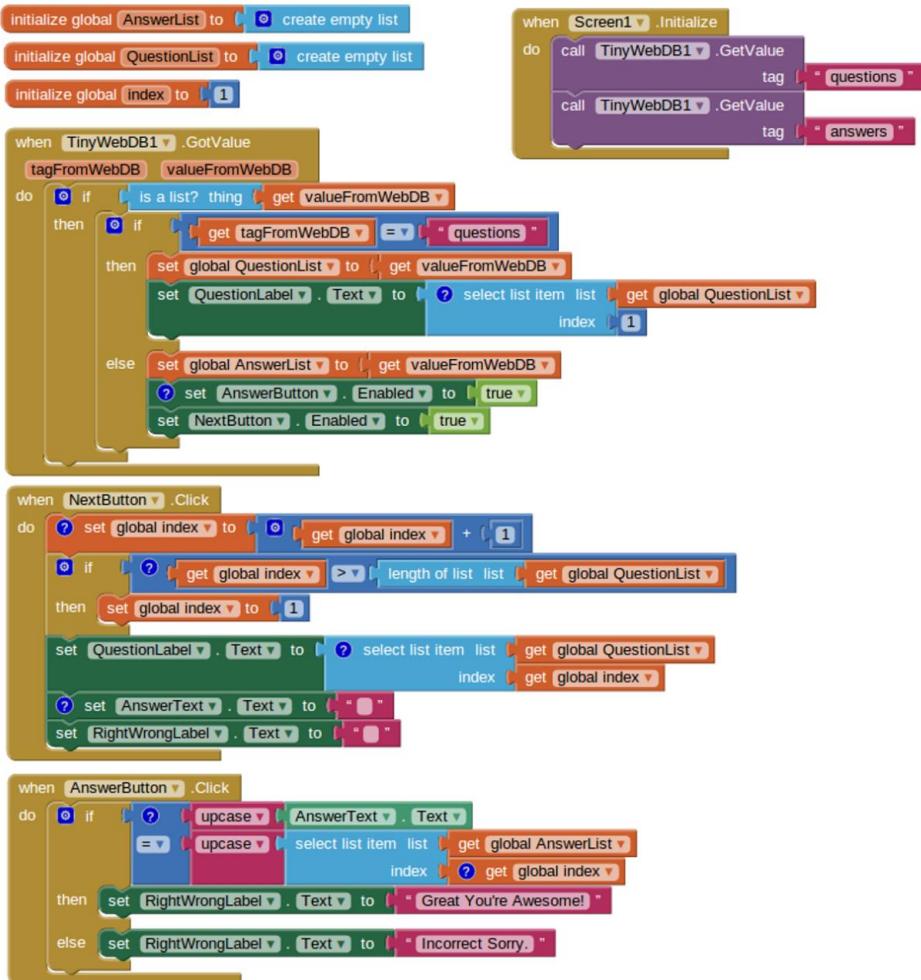


Figure 10-16. Les derniers blocs pour TakeQuiz

Variantes

Une fois que MakeQuiz et TakeQuiz fonctionnent, vous souhaiterez peut-être explorer certaines des variantes suivantes :

- Autorisez le créateur de quiz à spécifier une image pour chaque question. C'est un peu compliqué car TinyWebDB ne permet pas de stocker des images. Par conséquent, les images devront être des URL vers des images sur le Web, et le créateur du quiz devra saisir ces URL comme troisième élément dans le formulaire MakeQuiz. Notez que vous pouvez définir la propriété Picture d'un composant Image sur une URL.

184 Chapitre 10 : MakeQuiz et TakeQuiz

- Autoriser le créateur de quiz à supprimer des éléments des questions et réponses. Vous pouvez laisser l'utilisateur choisir une question à l'aide du composant ListPicker , et vous pouvez supprimer un élément avec le bloc Supprimer l'élément de liste (n'oubliez pas de supprimer des deux listes et de mettre à jour la base de données). Pour obtenir de l'aide sur ListPicker et la suppression de liste, consultez le chapitre 19.
- Laissez le créateur du quiz nommer le quiz. Vous devrez stocker le nom du quiz sous une balise différente dans la base de données et vous devrez charger le nom avec le quiz dans TakeQuiz. Après avoir chargé le nom, utilisez-le pour définir la propriété Screen.Title afin qu'elle apparaisse lorsque l'utilisateur répond à un quiz.
- Autoriser la création de plusieurs quiz nommés. Vous aurez besoin d'une liste de quiz, et vous pouvez utiliser chaque nom de quiz comme (une partie de) la balise pour stocker ses questions et réponses.

Résumé

Voici quelques-uns des concepts que nous avons abordés dans ce chapitre :

- Les données dynamiques sont des informations saisies par l'utilisateur de l'application ou chargées à partir d'une base de données. Un programme qui fonctionne avec des données dynamiques est plus abstrait. Pour plus d'informations, voir le chapitre 19.
- Vous pouvez stocker des données de manière persistante dans une base de données Web avec TinyWebDB composant.
- Vous récupérez des données d'une base de données TinyWebDB en les demandant avec TinyWebDB.GetValue. Lorsque la base de données Web renvoie les données, l'événement TinyWebDB.GotValue est déclenché. Dans le gestionnaire d'événements TinyWebDB.GotValue , vous pouvez placer les données dans une liste ou les traiter d'une manière ou d'une autre.
- Les données TinyWebDB peuvent être partagées entre plusieurs téléphones et applications. Pour plus d'informations sur les bases de données (Web), voir le chapitre 22.

Centre de diffusion

FrontlineSMS est un outil logiciel utilisé dans les pays en développement pour surveiller les élections, diffuser les changements météorologiques et connecter les personnes qui n'ont pas accès au Web mais qui en ont téléphones et connectivité mobile. C'est l'idée originale de Ken Banks, un pionnier de l'utilisation de la technologie mobile pour aider les personnes dans le besoin.

Le logiciel sert de hub pour les SMS communication au sein d'un groupe. Les gens envoient un code spécial pour rejoindre le groupe, après quoi ils reçoivent des messages diffusés depuis le hub. Pour les endroits sans accès à Internet, le hub de diffusion peut constituer une connexion vitale avec le monde extérieur.

Dans ce chapitre, vous allez créer une application de hub de diffusion qui fonctionne de manière similaire à FrontlineSMS. mais fonctionne sur un téléphone Android. Avoir le hub lui-même sur un appareil mobile signifie que l'administrateur peut être en déplacement, ce qui est particulièrement important dans des situations controversées, telles que l'observation d'élections et les négociations sur les soins de santé.

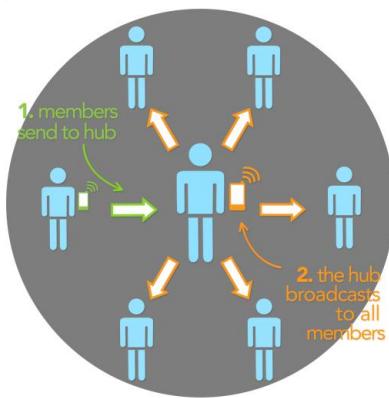
Dans ce chapitre, vous allez créer une application de hub de diffusion pour le fictif FlashMob Dance Team (FMDT), un groupe qui utilise le hub pour organiser des danses flash mob n'importe où et n'importe quand. Les gens s'inscriront au groupe en envoyant « joinFMDT » au hub, et toute personne inscrite pourra diffuser des messages à tous les autres membres du groupe.

Votre application traitera les messages texte reçus de la manière suivante :

1. Si le message texte est envoyé par une personne qui ne figure pas encore dans la liste de diffusion, l'application répond par un texte invitant la personne à rejoindre la liste de diffusion.
2. Si le message texte avec le code spécial « joinFMDT » est reçu, l'application ajoute l'expéditeur à la liste de diffusion.
3. Si le message texte est envoyé à partir d'un numéro déjà dans la liste de diffusion, le message est diffusé à tous les numéros de la liste.

Cette application est plus compliquée que l'application No Text While Driving du chapitre 4, mais vous allez la créer une fonctionnalité à la fois, en commençant par la première réponse automatique.

Figure 11-1.



message qui invite les gens à se joindre. Au moment où vous aurez terminé, vous aurez une assez bonne idée de la façon d'écrire des applications en utilisant le texte SMS comme interface utilisateur. Voulez-vous écrire une application de vote par texte comme celles utilisées dans les émissions télévisées de talents, ou la prochaine grande application d'envoi de SMS en groupe ? Vous apprendrez comment faire ici !

Ce que vous apprenez

Le didacticiel couvre les concepts App Inventor suivants, dont certains vous sont probablement déjà familiers :

- Le composant Texting pour l'envoi de textes et le traitement des textes reçus.
- Répertorier les variables et les données dynamiques : dans ce cas, pour garder une trace de la liste des téléphones. Nombres.
- Le pour chaque bloc pour permettre à une application de répéter des opérations sur une liste de données. Dans ce cas, vous utiliserez chacun pour diffuser des messages vers la liste des numéros de téléphone.
- Le composant TinyDB pour stocker les données de manière persistante. Cela signifie que si vous fermez l'application puis la relancez, la liste des numéros de téléphone sera toujours là.

Commencer

Vous aurez besoin d'un téléphone avec service SMS pour tester ou exécuter cette application. Vous devrez également recruter des amis pour vous envoyer des SMS afin de tester pleinement l'application.

Connectez-vous au site Web App Inventor et démarrez un nouveau projet. Nommez-le « BroadcastHub » et définissez également le titre de l'écran sur « Broadcast Hub ». Ensuite, connectez votre appareil ou émulateur pour des tests en direct.

Conception des composants

Broadcast Hub facilite la communication entre les téléphones mobiles. Il n'est pas nécessaire que l'application soit installée sur ces téléphones, ni même qu'il s'agisse de smartphones ; ils communiqueront par SMS avec votre application. Ainsi, dans ce cas, l'interface utilisateur de votre application est réservée à l'administrateur du groupe.

L'interface utilisateur pour l'administrateur est simple : elle affiche la diffusion en cours liste; c'est-à-dire la liste des numéros de téléphone inscrits au service et tous les SMS qu'il reçoit et diffuse.

Pour créer l'interface, ajoutez les composants répertoriés dans le tableau 11-1.

Tableau 11-1. Composants de l'interface utilisateur pour Broadcast Hub

Type de composant	Groupe de palettes	Comment l'appellerez-vous	But
Étiquette	Étiquette de l'interface utilisateur1		Il s'agit de l'en-tête « Numéros de téléphone enregistrés » ci-dessus. la liste des numéros de téléphone.
Étiquette	Interface utilisateur BroadcastListLabel	Affiche les numéros de téléphone enregistrés.	
Étiquette	Étiquette de l'interface utilisateur2		Il s'agit de l'en-tête « Journal d'activité » au-dessus du journal. information.
Étiquette	Étiquette de journal de l'interface utilisateur		Afficher un historique des textes reçus et diffusés.
Envoyer des SMS	Sociale	Envoyer des SMS1	Traitez les textes.
MinusculeDB	Interface utilisateur TinyDB1		Stockez la liste des numéros de téléphone enregistrés.

Lorsque vous ajoutez les composants, définissez les propriétés suivantes :

1. Définissez la largeur de chaque étiquette sur « Remplir le parent » afin qu'elle s'étende sur tout le téléphone.
horizontalement.
2. Définissez la taille de police des étiquettes d'en-tête (Label1 et Label2) sur 18 et vérifiez leur Boîtes FontBold.
3. Définissez la hauteur de BroadcastListLabel et LogLabel sur 200 pixels. Ils montreront plusieurs lignes.
4. Définissez la propriété Text de BroadcastListLabel sur « Liste de diffusion... ».
5. Définissez la propriété Text de LogLabel sur vide.
6. Définissez la propriété Text de Label1 sur « Numéros de téléphone enregistrés ».
7. Définissez la propriété Text de Label2 sur « Journal d'activité ».

La figure 11-1 montre la disposition de l'application dans Component Designer.

188 Chapitre 11 : Hub de diffusion

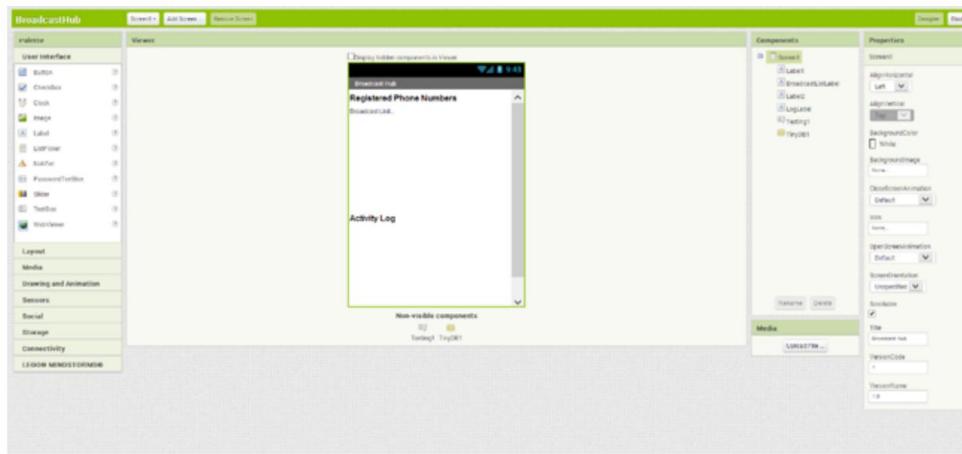


Figure 11-2. Hub de diffusion dans le concepteur de composants

Ajout de comportements aux composants

L'activité de Broadcast Hub n'est pas déclenchée par la saisie d'informations par l'utilisateur ou cliquer sur un bouton ; il s'agit plutôt de SMS provenant d'autres téléphones. Pour traiter ces textes et stockez les numéros de téléphone qui les ont envoyés dans une liste, vous aurez besoin des éléments suivants comportements :

- Lorsque le message texte est envoyé par une personne qui ne figure pas déjà dans la liste de diffusion, l'application répond par un texte qui invite l'expéditeur à se joindre.
- Lorsque le message texte « joinFMDT » est reçu, enregistrez l'expéditeur dans le cadre de la liste de diffusion.
- Lorsque le message texte est envoyé à partir d'un numéro déjà dans la liste de diffusion, le message est diffusé à tous les numéros de la liste.

RÉPONDRE AUX TEXTES ENTRANTS

Vous commencerez par créer le premier comportement : lorsque vous recevez un SMS, envoyez un message à l'expéditeur en l'invitant à s'inscrire en vous envoyant « joinFMDT » par SMS. Vous allez besoin des blocs répertoriés dans le tableau 11-2.

Tableau 11-2. Blocs pour ajouter la fonctionnalité permettant d'inviter des personnes au groupe par SMS

Type de bloc	Tiroir	But
SMS1.MessageReceived	Envoyer des SMS1	Déclenché lorsque le téléphone reçoit un SMS.
définissez Texting1.PhoneNumber sur	Envoyer des SMS1	Définissez le numéro du texte de retour.

Type de bloc	Tiroir	But
obtenir un numéro	Faites glisser depuis Événement MessageReçu gestionnaire	L'argument de MessageReceived. Il s'agit du numéro de téléphone de l'expéditeur.
définir Texting1.Message	Envoyer des SMS1	Définissez le message d'invitation à envoyer.
texte ("Pour rejoindre cette liste de diffusion, envoyez 'joinFMDT' à ce numéro »)	Texte	Le message d'invitation.
SMS1.SendMessage	Envoyer des SMS1	Envoyez-le!

Comment fonctionnent les blocs

Si vous avez complété l'application Pas de SMS pendant la conduite au chapitre 4, ces blocs devraient sembler familier. Texting1.MessageReceived est déclenché lorsque le téléphone reçoit un message texte. La figure 11-2 montre comment les blocs du gestionnaire d'événements définissent le PhoneNumber et Message du composant Texting1 , puis envoyez le message.

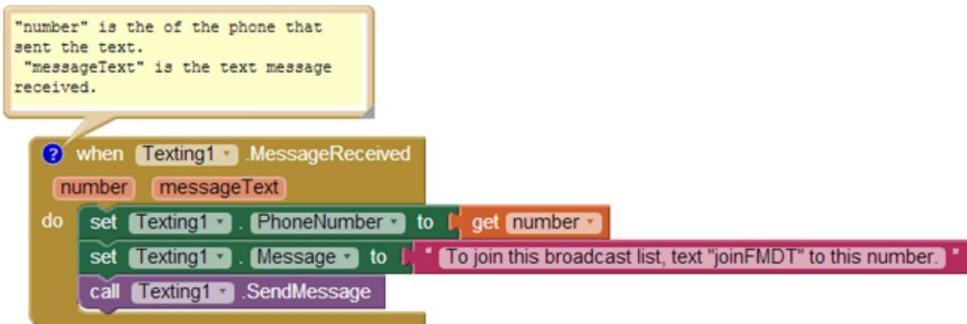


Figure 11-3. Renvoyer le message d'invitation après avoir reçu un SMS



Testez votre application Vous aurez besoin d'un deuxième téléphone pour tester ce comportement ; vous ne voulez pas vous envoyer de SMS, car cela pourrait tourner en boucle pour toujours ! Si vous n'avez pas d'autre téléphone, vous pouvez vous inscrire sur Google Voix ou un service similaire et envoyer des SMS à partir de ce service à votre téléphone. Depuis le deuxième téléphone, envoyez le SMS « bonjour » à le téléphone exécutant l'application. Le deuxième téléphone devrait alors recevoir un SMS l'invitant à rejoindre le groupe.

AJOUTER DES NUMÉROS À LA LISTE DE DIFFUSION

Il est temps de créer les blocs pour le deuxième comportement : lorsque le message texte « joinFMDT » est reçu, l'expéditeur est ajouté à la liste de diffusion. Tout d'abord, vous devrez définir une variable de liste, BroadcastList, pour stocker les numéros de téléphone enregistrés. Depuis

190 Chapitre 11 : Hub de diffusion

dans le tiroir Variables, faites glisser un bloc global d'initialisation et nommez-le "Liste de diffusion". Initialisez-le sur une liste vide en utilisant un bloc de création de liste à partir du Tiroir de listes, comme le montre la figure 11-3 (nous ajouterons la fonctionnalité pour construire cette liste prochainement).



Figure 11-4. La variable BroadcastList pour stocker la liste des numéros enregistrés

Ensuite, modifiez le gestionnaire d'événements Texting1.MessageReceived afin qu'il ajoute le numéro de téléphone de l'expéditeur à la BroadcastList si le message reçu est « joinFMDT ». Vous aurez besoin d'un bloc if else pour vérifier le message et d'un bloc ajouter un élément à la liste pour ajoutez le nouveau numéro à la liste. L'ensemble complet des blocs dont vous aurez besoin est répertorié dans le tableau 11-3. Après avoir ajouté le numéro à la liste, affichez la nouvelle liste dans BroadcastListLabel.

Tableau 11-3. Blocs pour vérifier un message texte et ajouter l'expéditeur à la liste de diffusion

Type de bloc	Tiroir	But
sinon	Contrôle	En fonction du message reçu, procédez différemment des choses.
=	Mathématiques	Déterminer si messageText est égal à "rejoignez FMDT."
obtenir le texte du message	Sortez de Événement MessageReçu gestionnaire	Branchez-le dans le bloc =.
texte (« rejoindreFMDT »)	Texte	Branchez-le dans le bloc =.
ajouter des éléments à la liste	Listes	Ajoutez le numéro de l'expéditeur à BroadcastList.
obtenir la liste de diffusion globale	Faire glisser depuis la variable bloc d'initialisation.	La liste.
obtenir un numéro	Sortez de Événement MessageReçu gestionnaire	Branchez-le en tant qu'élément d'ajout d'éléments à la liste.
définir BroadcastListLabel.Text sur	Étiquette de liste de diffusion	Affichez la nouvelle liste.
Liste de diffusion globale	Faire glisser depuis la variable bloc d'initialisation	Branchez-le pour régler le BroadcastListLabel.Text à bloquer.
définissez Texting1.Message sur	Envoyer des SMS1	Préparez l'envoi de SMS pour renvoyer un message au expéditeur.
texte (« Félicitations, vous... »)	Texte	Félicitez l'expéditeur pour avoir rejoint le groupe.

Comment fonctionnent les blocs

La première rangée de blocs illustrée dans la figure 11-4 définit Texting1.PhoneNumber sur le numéro de téléphone du message qui vient d'être reçu ; nous savons que nous allons répondre à l'expéditeur, donc cela configure cela. L'application demande ensuite si le messageText était le code spécial « joinFMDT ». Si tel est le cas, le numéro de téléphone de l'expéditeur est ajouté à la BroadcastList et un message de félicitations est envoyé. Si le messageText est autre chose que « joinFMDT », le message de réponse répète le message d'invitation. Après le bloc if else , le message de réponse est envoyé (rangée du bas des blocs).

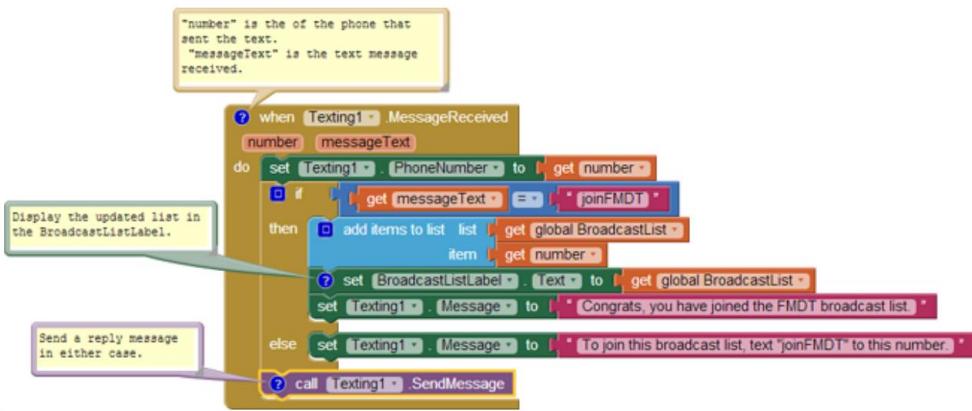


Figure 11-5. Si le message entrant est « joinFMDT », ajoutez l'expéditeur à BroadcastList



Testez votre application Depuis un téléphone n'exécutant pas l'application, envoyez le message texte « joinFMDT » au téléphone exécutant l'application. Vous devriez voir le numéro de téléphone répertorié dans l'interface utilisateur sous « Numéros de téléphone enregistrés ». Le téléphone « expéditeur » devrait également recevoir le message de félicitations sous forme de texte de réponse. Essayez également d'envoyer un message autre que « joinFMDT » pour vérifier si le message d'invitation est toujours envoyé correctement.

DIFFUSER LES MESSAGES

Ensuite, vous ajouterez le comportement afin que l'application diffuse les messages reçus aux numéros de BroadcastList, mais uniquement si le message provient d'un numéro déjà stocké dans cette liste. Cette complexité supplémentaire nécessitera davantage de blocs de contrôle, dont un autre if else et un pour chacun. Vous aurez besoin d'un bloc if else supplémentaire pour vérifier si le numéro est dans la liste, et d'un pour chaque bloc pour diffuser le message à chaque numéro de la liste. Vous devrez également déplacer les blocs if else du précédent

192 Chapitre 11 : Hub de diffusion

comportement et insérez-les dans la partie else du nouveau if else. Tous les supplémentaires les blocs dont vous aurez besoin sont répertoriés dans le tableau 11-4.

Tableau 11-4. Blocs pour vérifier si l'expéditeur est déjà dans le groupe

Type de bloc	Tiroir	But
sinon	Contrôle	Selon que l'expéditeur est déjà présent la liste, faites différentes choses.
est dans la liste ?	Listes	Vérifiez si quelque chose se trouve dans une liste.
obtenir la liste de diffusion globale	Faire glisser depuis la variable bloc d'initialisation	Branchez-le dans la prise « list » de est dans la liste ?
obtenir un numéro	Sortez de Événement MessageReçu gestionnaire	Branchez-le dans la prise « chose » de la liste ?
pour chaque	Contrôle	Envoyez à plusieurs reprises un message à tous les membres dans la liste.
obtenir la liste de diffusion globale	Faire glisser depuis la variable bloc d'initialisation	Branchez-le dans la prise « liste » de chacun.
définissez Texting1.Message sur	Envoyer des SMS1	Définissez le message.
obtenir le texte du message	Sortez du Événement MessageReçu	Le message qui a été reçu et sera diffuser.
définissez Texting1.PhoneNumber sur	Envoyer des SMS1	Définissez le numéro de téléphone.
obtenir l'article	Faites glisser pour chaque bloc	Maintenez l'élément actuel de BroadcastList ; c'est un numéro (de téléphone).

Comment fonctionnent les blocs

L'application est devenue suffisamment complexe pour nécessiter un bloc if else imbriqué , qui vous pouvez le voir sur la figure 11-5. Un bloc if else imbriqué est un bloc branché sur le prise du if ou bien partie d'un autre, externe if else. Dans ce cas, le if else externe La succursale vérifie si le numéro de téléphone du message reçu est déjà dans le liste. Si tel est le cas, le message est relayé à toutes les personnes figurant dans la liste. Si le numéro n'est pas dans la liste, le test imbriqué est effectué : les blocs vérifient si le messageText est égal à « rejoignez FMDT » et effectuez une branche de deux manières en fonction de la réponse.

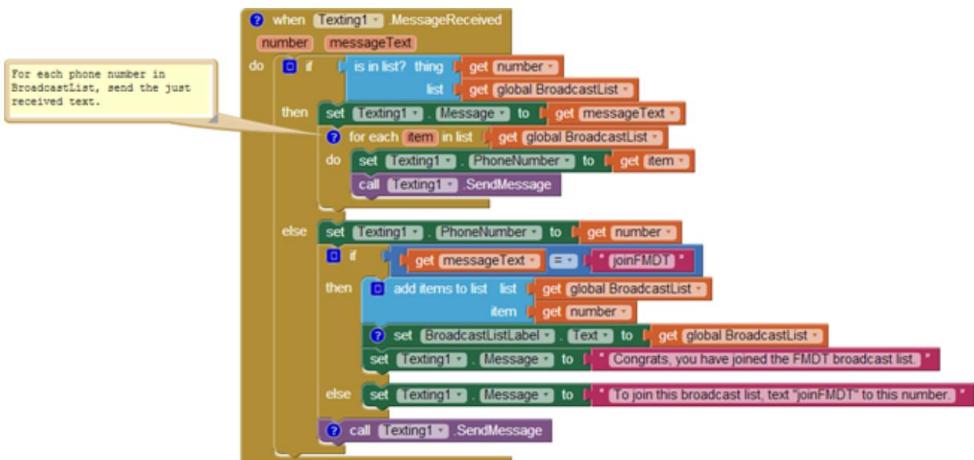


Figure 11-6. Les blocs vérifient si l'expéditeur est déjà dans le groupe et diffusent le message si c'est le cas

En général, les blocs if et if else peuvent être imbriqués à des niveaux arbitraires, vous donnant le pouvoir de programmer des comportements de plus en plus complexes (voir le chapitre 18 pour plus d'informations sur les blocs conditionnels).

Le message est diffusé en utilisant un for each (dans la clause then externe). Le for each parcourt la BroadcastList et envoie le message à chaque élément. Comme le for each se répète, chaque numéro de téléphone suivant de la BroadcastList est stocké dans l'élément (l'élément est un espace réservé variable pour l'élément en cours de traitement dans le pour chacun). Les blocs à l'intérieur de chaque ensemble Texting.PhoneNumber sur l'élément actuel, puis envoient le message. Pour plus d'informations sur le fonctionnement de chacun , consultez le chapitre 20.



Testez votre application Tout d'abord, enregistrez deux téléphones différents en envoyant « joinFMDT » au téléphone exécutant l'application. Ensuite, envoyez un autre message depuis l'un des téléphones. Les deux téléphones devraient recevoir le SMS (y compris celui qui l'a envoyé).

Embellir l'affichage de la liste

L'application peut désormais diffuser des messages, mais l'interface utilisateur pour l'administrateur de l'application nécessite quelques travaux. Premièrement, la liste des numéros de téléphone s'affiche de manière peu élégante. Plus précisément, lorsque vous placez une variable de liste dans une étiquette, elle affiche la liste avec des espaces entre les éléments, en plaçant autant que possible sur chaque ligne. Ainsi, BroadcastListLabel pourrait afficher la BroadcastList comme ceci :

(+1415111-1111 +1415222-2222 +1415333-3333 +1415444-4444)

194 Chapitre 11 : Hub de diffusion

Pour améliorer ce formatage, créez une procédure nommée `displayBroadcastList` en utilisant les blocs répertoriés dans le tableau 11-5. Cette procédure affiche la liste avec chaque téléphone numéro sur une ligne séparée. Assurez-vous d'appeler la procédure ci-dessous pour ajouter des éléments à bloc de liste pour que la liste mise à jour soit affichée.

Tableau 11-5. Des blocs pour nettoyer l'affichage des numéros de téléphone dans votre liste

Type de bloc	Tiroir	But
à la procédure (« <code>displayBroadcastList</code> »)	Procédures	Créez la procédure (ne choisissez pas le résultat de la procédure).
définir <code>BroadcastListLabel.Text</code> sur <code>BroadcastListLabel</code>		Affichez la liste ici.
texte ("")	Texte	Cliquez sur le texte, puis cliquez sur Supprimer pour créer un objet texte vide.
pour chaque	Contrôle	Parcourez les nombres.
obtenir la liste de diffusion globale	Faire glisser depuis la variable bloc d'initialisation	Branchez-le dans la prise « dans la liste » de chacun.
définir <code>BroadcastListLabel.Text</code> sur <code>BroadcastListLabel</code>		Modifiez-le avec chacun des nombres.
rejoindre le texte	Texte	Créez un objet texte à partir de plusieurs parties.
<code>BroadcastListLabel.Text</code>	Étiquette de liste de diffusion	Ajoutez ceci à l'étiquette à chaque itération de pour chaque.
texte (« <code>\n</code> »)	Texte	Ajoutez un caractère de nouvelle ligne pour que le prochain numéro est sur la ligne suivante.
obtenir l'article	Faites glisser pour chaque bloc.	Le numéro actuel de la liste.

Comment fonctionnent les blocs

Le `for each` dans `displayBroadcastList` ajoute successivement un numéro de téléphone à la fin de l'étiquette, comme le montre la figure 11-6, en plaçant un caractère de nouvelle ligne (`\n`) entre chaque élément afin d'afficher chaque numéro sur une nouvelle ligne.

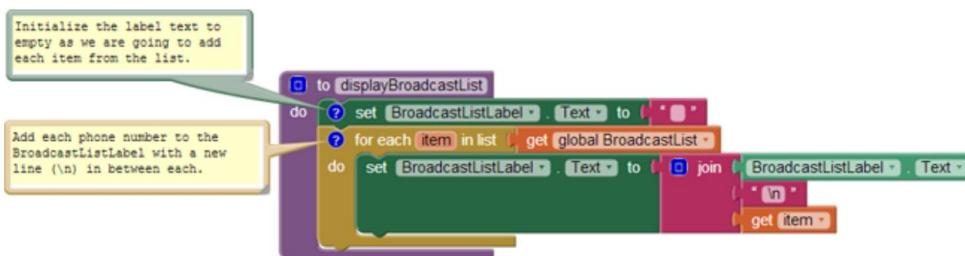


Figure 11-7. Afficher les numéros de téléphone avec un caractère de nouvelle ligne entre chacun

Bien entendu, cette procédure `displayBroadcastList` ne fera rien à moins que vous ne l'appeliez. Appelez-le dans le gestionnaire d'événements `Texting1.MessageReceived`, juste en dessous de l'appel pour ajouter un élément à la liste. L'appel doit remplacer les blocs qui définissent simplement `BroadcastListLabel.Text` sur `BroadcastList`. Vous pouvez trouver l' appel

bloc `displayBroadcastList` dans le tiroir Procédures.

La figure 11-7 montre comment les blocs pertinents dans `Texting1.MessageReceived` le gestionnaire d'événements devrait regarder.

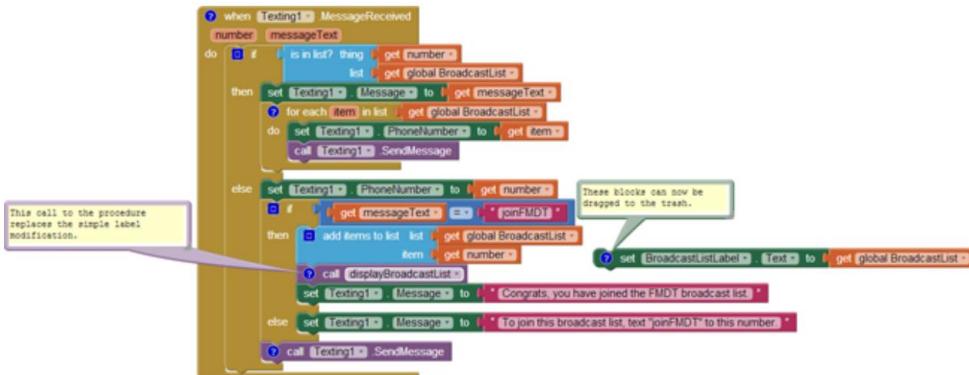


Figure 11-8. Appel de la procédure `displayBroadcastList`

Pour plus d'informations sur l'utilisation de `for each` pour afficher une liste, voir le chapitre 20. Pour plus d'informations sur la création et l'appel de procédures, voir le chapitre 21.



Testez votre application Redémarrez l'application pour effacer la liste, puis enregistrez (à nouveau) au moins deux téléphones différents. Les numéros de téléphone apparaissent-ils sur des lignes distinctes ?

ENREGISTREMENT DES TEXTES DIFFUSÉS

Lorsqu'un SMS est reçu et diffusé sur les autres téléphones, l'application doit enregistrer cette occurrence afin que l'administrateur puisse surveiller l'activité. Dans le Concepteur de composants, vous avez ajouté l'étiquette `LogLabel` à l'interface utilisateur à cet effet. Maintenant, vous allez coder quelques blocs qui changent `LogLabel` à chaque fois qu'un nouveau texte arrive.

Vous devez créer un texte qui dit quelque chose comme « un message du +1415111-2222 a été diffusé ». Le numéro +1415111-2222 n'est pas une donnée fixe ; il s'agit plutôt de la valeur du numéro d'argument fourni avec l'événement `MessageReceived`. Donc, pour construire le texte, vous allez concaténer la première partie, « message de », avec un bloc de numéro d'obtention et enfin avec la dernière partie du message, le texte « diffusion ».

196 Chapitre 11 : Hub de diffusion

Comme vous l'avez fait dans les chapitres précédents, utilisez join pour concaténer les parties en utilisant les blocs répertoriés dans le tableau 11-6.

Tableau 11-6. Des blocs pour construire votre journal des messages diffusés

Type de bloc	Tiroir	But
définir LogLabel.Text sur LogLabel		Affichez le journal ici.
rejoindre	Texte	Créez un objet texte à partir de plusieurs parties.
texte (« message de »)	Texte	Ceci est le message du rapport.
obtenir un numéro	Sortez de Événement MessageReçu gestionnaire	Le numéro de téléphone de l'expéditeur.
texte (« diffusion\n »)	Texte	Ajoutez la dernière partie de « message de diffusion du 111 au 2222 » et inclure une nouvelle ligne.
LogLabel.Text	Etiquette de journal	Ajoutez un nouveau journal aux précédents.

Comment fonctionnent les blocs

Après avoir diffusé le message reçu à tous les numéros de BroadcastList, le programme modifie désormais le LogLabel pour ajouter un rapport du texte qui vient d'être diffusé, comme indiqué dans la figure 11-8. Notez que le message est ajouté au début de la liste au lieu de la fin. De cette façon, le message le plus récent envoyé au groupe apparaît en haut.

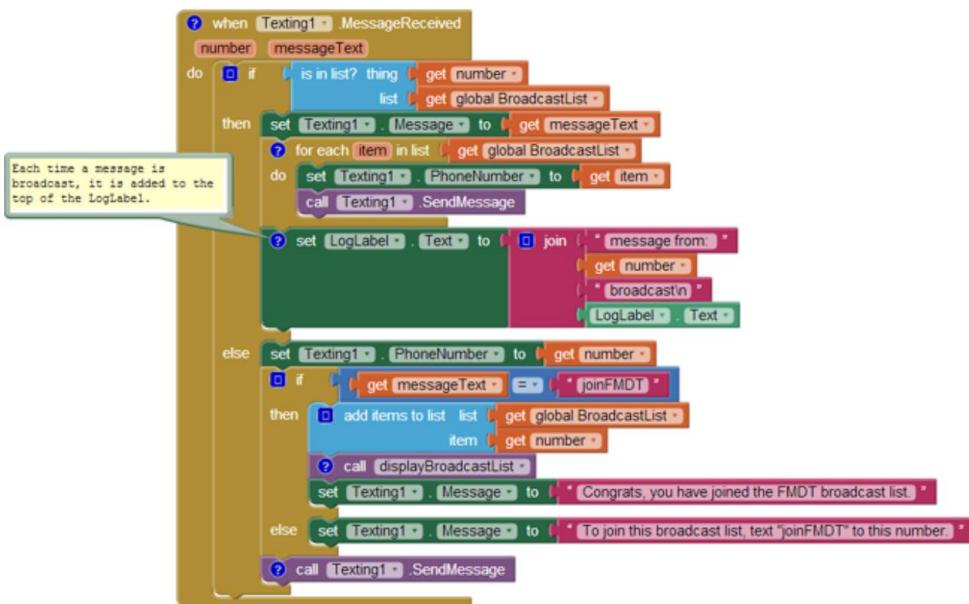


Figure 11-9. Ajout d'un nouveau message de diffusion au journal

Le bloc join crée de nouvelles entrées du formulaire :

message de : 111-2222 diffusé

Chaque fois qu'un texte est diffusé, l'entrée du journal est ajoutée au début (au début) du LogLabel.Text pour que les entrées les plus récentes apparaissent en premier. La façon dont vous organisez le bloc de jointure détermine l'ordre des entrées. Dans ce cas, le nouveau message est ajouté avec les trois premiers sockets de jointure et LogLabel.Text, qui contient les entrées existantes, est branché sur le dernier socket.

Le « \n » dans le texte « broadcast\n » est le caractère de nouvelle ligne qui provoque chaque journal entrée à afficher sur une ligne distincte :

**message de : 1112222 diffusé
message de : 555-6666 diffusé**

Pour plus d'informations sur l'utilisation de for each pour afficher une liste, voir le chapitre 20.

STOCKAGE DE LA LISTE DE DIFFUSION DANS UNE BASE DE DONNÉES

Votre application fonctionne en quelque sorte, mais si vous avez suivi certains des tutoriels précédents, vous avez probablement deviné qu'il y a un problème : si l'administrateur ferme l'application et la relance, la liste de diffusion sera perdue et tout le monde devra s'inscrire. encore.

Pour résoudre ce problème, vous utiliserez le composant TinyDB pour stocker et récupérer la BroadcastList vers et depuis une base de données.

Vous utiliserez un schéma similaire à celui que vous avez utilisé dans l'application MakeQuiz (Chapitre 10) :

- Stockez la liste dans la base de données chaque fois qu'un nouvel élément est ajouté.
- Lorsque l'application démarre, chargez la liste de la base de données dans une variable.

Commencez par coder les blocs répertoriés dans le tableau 11-7 pour stocker la liste dans la base de données. Avec le composant TinyDB , une balise est utilisée pour identifier les données et les distinguer des autres données stockées dans la base de données. Dans ce cas, vous pouvez marquer les données comme « broadcastList ». Vous ajouterez les blocs dans l' événement Texting1.MessageReceived , sous le bloc Ajouter des éléments à la liste .

Tableau 11-7. Blocs pour stocker la liste avec TinyDB

Type de bloc	Tiroir	But
TinyDB1.StoreValue	MinusculeDB1	Stockez les données dans la base de données.
texte (« broadcastList »)	Texte	Branchez-le dans l'emplacement « tag » de StoreValue.
obtenir la liste de diffusion globale	Faire glisser depuis le bloc d'initialisation des variables	Branchez-le dans l'emplacement « valeur » de StoreValue.

198 Chapitre 11 : Hub de diffusion

Comment fonctionnent les blocs

Lorsqu'un SMS « joinFMDT » arrive et que le numéro de téléphone du nouveau membre est ajouté à la liste, TinyDB1.StoreValue est appelée pour stocker la BroadcastList dans la base de données. Le tag (un objet texte nommé BroadcastList) est utilisé afin que vous puissiez récupérer ultérieurement le données. La figure 11-9 montre que la valeur appelée par StoreValue est la variable Liste de diffusion.

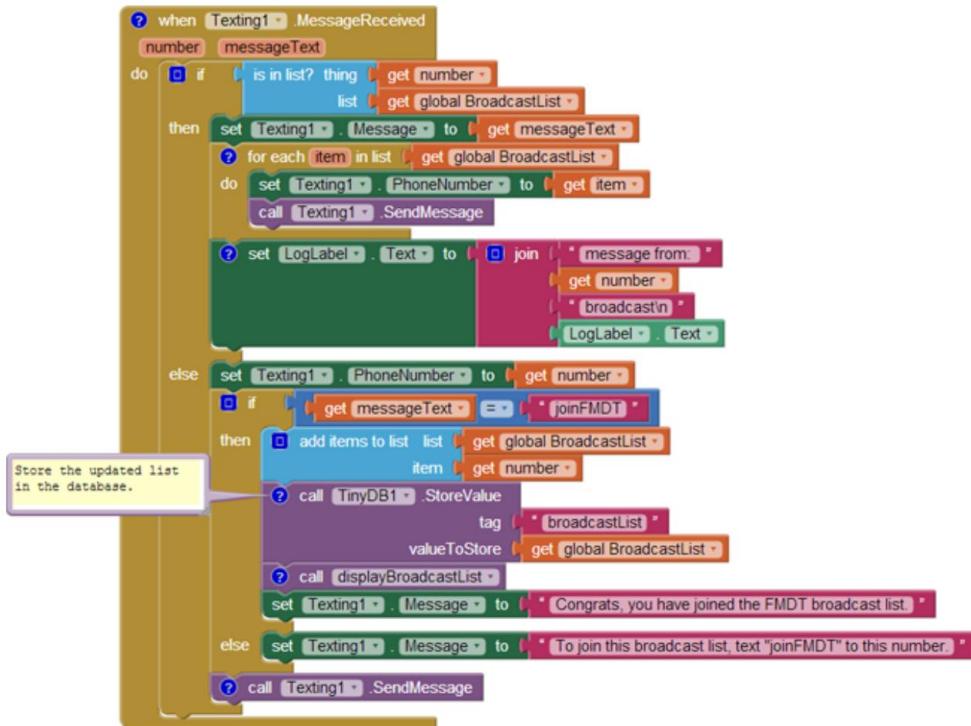


Figure 11-10. Appeler TinyDB pour stocker la BroadcastList

CHARGEMENT DE LA LISTE DE DIFFUSION À PARTIR D'UNE BASE DE DONNÉES

Ajoutez les blocs répertoriés dans le tableau 11-8 pour recharger la liste à chaque fois que l'application lance.

Tableau 11-8. Blocs pour charger à nouveau la liste de diffusion dans l'application lors de son lancement

Type de bloc	Tiroir	But
Écran1.Initialiser	Écran1	Déclenché au lancement de l'application.
TinyDB1.GetValue	MinusculeDB1	Demandez les données de la base de données.
texte (« broadcastList »)	Texte	Branchez-le dans la prise « tag » de GetValue.

Type de bloc	Tiroir	But
	appelez les procédures displayBroadcastList	Après avoir chargé les données, affichez-les.

Lorsque l'application démarre, l' événement Screen1.Initialize est déclenché, donc vos blocs ira dans ce gestionnaire d'événements.

Comment fonctionnent les blocs

Lorsque l'application démarre, l' événement Screen1.Initialize est déclenché. Les blocs illustrés dans la figure 11-10 demandent les données de la base de données avec TinyDB1.GetValue.

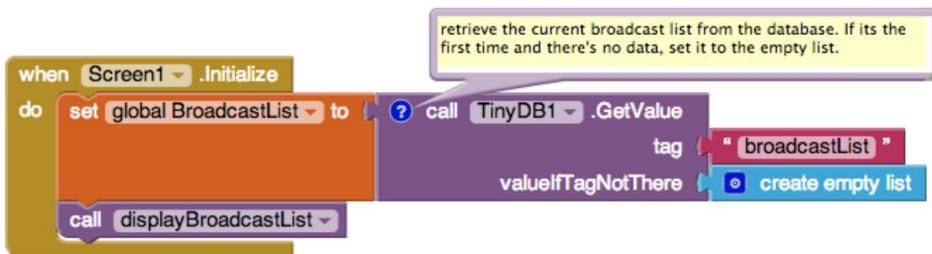


Figure 11-11. Chargement de la BroadcastList à partir de la base de données

Vous appelez TinyDB.GetValue en utilisant la même balise que celle utilisée pour stocker la liste (broadcastList). Dans le cas général, la liste de numéros de téléphone précédemment stockée sera renvoyée et placée dans la variable BroadcastList. Mais TinyDB.GetValue fournit un socket, valueIfTagNotThere, pour spécifier ce que le bloc doit renvoyer s'il n'y a pas encore de données dans la base de données pour cette balise, comme cela se produira lors de la première exécution de cette application. Dans ce cas, une liste vide est renvoyée.



Testez votre application Vous pouvez utiliser les tests en direct pour les applications qui modifient la base de données, mais faites-le avec précaution. Dans ce cas, envoyez un SMS à l'application avec un autre téléphone pour ajouter des numéros à la BroadcastList, puis redémarrez l'application. Vous pouvez redémarrer en mode test en direct en passant au concepteur et en modifiant certaines propriétés, même par exemple en changeant la police d'une étiquette. Notez que pour tester entièrement les applications de base de données, vous devez empaqueter et réellement télécharger l'application sur un téléphone (choisissez « Construire > Application (enregistrer l'apk sur mon ordinateur) »). Après avoir téléchargé votre application, utilisez vos autres téléphones pour envoyer un SMS. pour rejoindre le groupe puis fermez l'application. Si les numéros sont toujours répertoriés lorsque vous relancez l'application, la partie base de données fonctionne.

L'application complète : Broadcast Hub

La figure 11-11 illustre les blocs de l'application Broadcast Hub terminée.

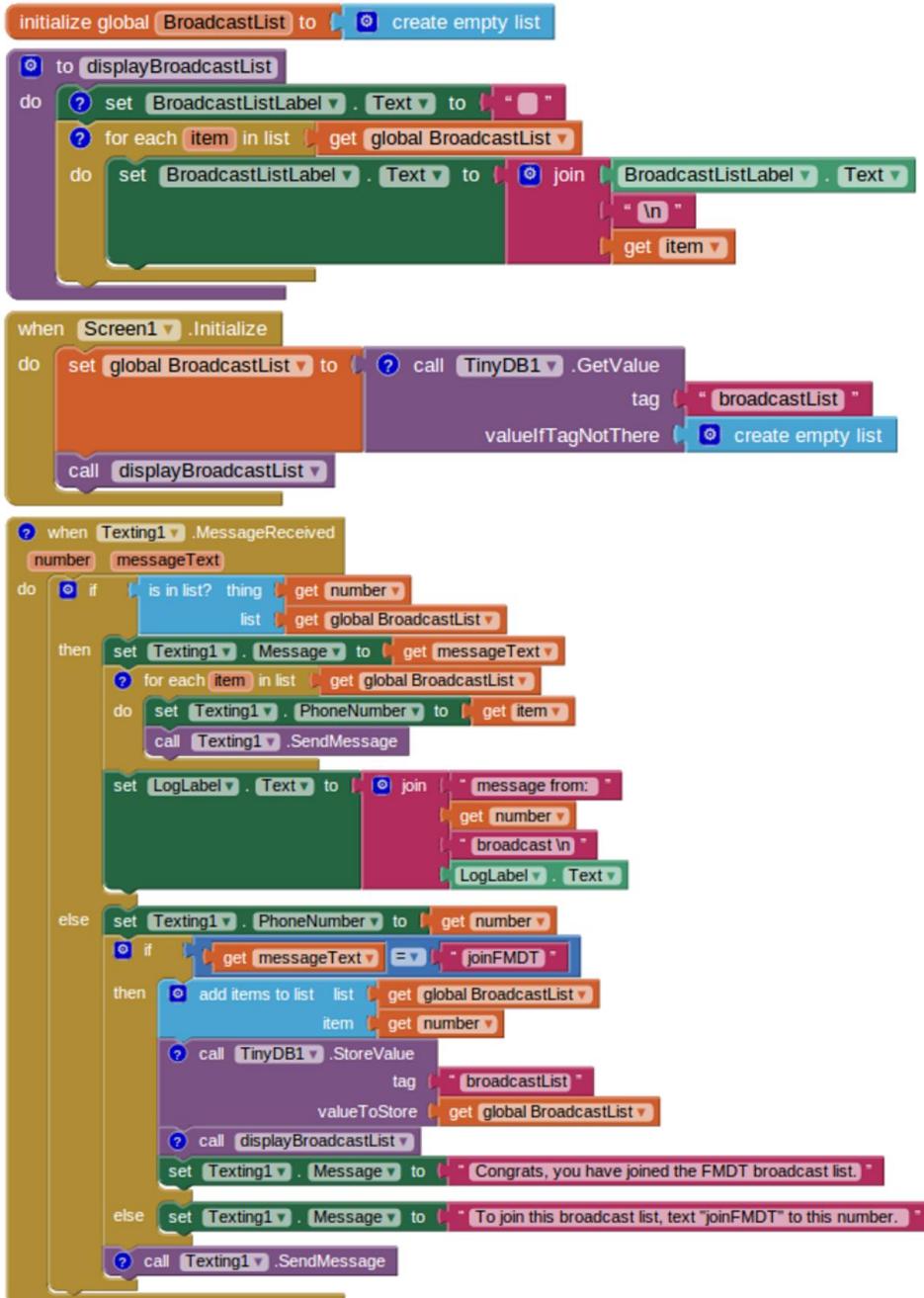


Figure 11-12. L'application complète Broadcast Hub

Variantes

Après avoir célébré la création d'une application aussi complexe, vous souhaiterez peut-être explorer certaines des variantes suivantes :

- L'application diffuse chaque message à tout le monde, y compris au téléphone qui a envoyé le message. Modifiez cela pour que le message soit diffusé à tout le monde sauf à l'expéditeur.
- Autoriser les téléphones clients à se retirer de la liste en envoyant « quitter » par SMS au application. Vous aurez besoin d'un bloc de suppression de la liste .
- Donnez à l'administrateur du hub la possibilité d'ajouter et de supprimer des numéros du liste de diffusion via l'interface utilisateur.
- Laissez l'administrateur du hub spécifier les numéros qui ne doivent pas être autorisés dans le liste.
- Personnalisez l'application pour que tout le monde puisse s'y joindre pour recevoir des messages, mais uniquement les L'administrateur peut diffuser des messages.
- Personnalisez l'application pour que n'importe qui puisse la rejoindre pour recevoir des messages, mais seule une liste fixe de numéros de téléphone peut diffuser des messages au groupe.

Résumé

Voici quelques-uns des concepts que nous avons abordés dans ce didacticiel :

- Les applications peuvent réagir à des événements qui ne sont pas initiés par l'utilisateur de l'application, comme un message texte. en cours de réception. Cela signifie que vous pouvez créer des applications dans lesquelles vos « utilisateurs » se trouvent sur un autre téléphone.
- Vous pouvez utiliser des blocs if else et for each imbriqués pour coder des comportements complexes.
Pour plus d'informations sur les conditions et pour chaque itération, voir respectivement le chapitre 18 et le chapitre 20 .
- Vous pouvez utiliser le bloc de jointure pour créer un objet texte à partir de plusieurs parties.
- Vous pouvez utiliser TinyDB pour stocker et récupérer des données d'une base de données. Un général Le schéma consiste à appeler StoreValue pour mettre à jour la base de données chaque fois que les données changent et à appeler GetValue pour récupérer les données de la base de données au démarrage de l'application.

Robot à distance

Dans ce chapitre, vous allez créer une application qui transforme votre téléphone Android en télécommande pour un robot LEGO MINDSTORMS NXT. L'application aura des boutons pour faire avancer et reculer le robot, tourner à gauche et à droite et s'arrêter.

Vous allez le programmer pour que le robot s'arrête automatiquement s'il détecte un obstacle. L'application utilisera les capacités Bluetooth du téléphone pour communiquer avec le robot.

Les robots LEGO MINDSTORMS sont amusants à jouer, mais ils sont aussi pédagogiques. Les programmes parascolaires utilisent des robots pour enseigner aux enfants des écoles primaires et secondaires des compétences en résolution de problèmes et les initier à l'ingénierie et à la programmation informatique. Les robots NXT sont également utilisés par les enfants âgés de 9 à 14 ans dans les compétitions de robotique de la FIRST Lego League.

Le kit robotique programmable NXT comprend une unité principale appelée NXT Intelligent Brick. Il peut contrôler trois moteurs et quatre capteurs d'entrée. Vous pouvez assembler un robot à partir d'éléments de construction LEGO, d'engrenages, de roues, de moteurs et de capteurs. Le kit est livré avec son propre logiciel pour programmer le robot, mais vous pouvez désormais utiliser App Inventor pour créer des applications Android permettant de contrôler un NXT via la connectivité Bluetooth.

L'application de ce chapitre est conçue pour fonctionner avec un robot doté de roues et d'un capteur à ultrasons, tel que le robot Shooterbot illustré ici. Le Shooterbot est souvent le premier robot que les gens construisent avec l'ensemble LEGO MINDSTORMS NXT 2.0. Il a des roues gauches connectées au port de sortie C, des roues droites connectées au port de sortie B, un capteur de couleur connecté au port d'entrée 3 et un capteur à ultrasons connecté au port d'entrée 4.

Figure 12-1.



Ce que vous apprendrez

Ce chapitre utilise les composants et concepts suivants :

- Le composant BluetoothClient pour la connexion au NXT

- Le composant ListPicker pour fournir une interface utilisateur pour la connexion au NXT
- Le composant NxtDrive pour entraîner les roues du robot
- Le composant NxtUltrasonicSensor pour utiliser le capteur ultrasonique du robot pour détecter les obstacles
- Le composant Notifier pour afficher les messages d'erreur

Commencer

Vous aurez besoin de la version Android 2.0 ou supérieure pour utiliser l'application dans ce chapitre. De plus, pour des raisons de sécurité, les appareils Bluetooth doivent être couplés avant de pouvoir se connecter les uns aux autres. Avant de commencer à créer l'application, vous devrez associer votre Android à votre NXT en suivant ces étapes :

1. Sur le NXT, cliquez sur la flèche droite jusqu'à ce qu'elle indique Bluetooth, puis appuyez sur le bouton carré orange.

2. Cliquez sur la flèche droite jusqu'à ce que le mot Visibilité apparaisse, puis appuyez sur le bouton carré orange.

3. Si la valeur Visibilité est déjà Visible, passez à l'étape 4. Sinon, cliquez sur le bouton gauche ou la flèche droite pour définir la valeur sur Visible.

4. Sur Android, accédez à Paramètres.

Les étapes 5 à 7 peuvent varier légèrement en fonction de votre appareil Android.

5. Assurez-vous que Bluetooth est activé.

6. Cliquez sur « Bluetooth », puis sur « Rechercher des appareils ».

7. Sous « Appareils disponibles », recherchez un appareil nommé « NXT ». Si tu as déjà changé le nom de votre robot, recherchez un nom d'appareil qui correspond au nom de votre robot au lieu de « NXT ».

8. Cliquez sur « NXT » ou sur le nom de votre robot.

9. Sur le NXT, vous devriez voir une invite pour un mot de passe. Appuyez sur le carré orange pour acceptez 1234.

10. Sur Android, vous devriez voir une invite pour le code PIN. Tapez 1234 puis appuyez sur D'ACCORD.

11. Votre robot et votre Android sont désormais couplés.

Connectez-vous au site Web App Inventor à l'adresse ai2.appinventor.mit.edu. Démarrez un nouveau projet et nommez-le « NXTremoteControl », puis définissez le titre de l'écran sur « NXT Remote

Contrôle". Cliquez sur Connecter et configurez votre appareil (ou émulateur) pour des tests en direct (voir <http://appinventor.mit.edu/explore/ai2/setup> pour obtenir de l'aide sur cette configuration).

Conception des composants

Pour cette application, nous devrons créer et définir des comportements à la fois pour les personnes non visibles et visibles. Composants.

COMPOSANTS NON VISIBLES

Avant de créer les composants de l'interface utilisateur, vous allez créer des composants non visibles. composants, répertoriés dans le tableau 12-1 et illustrés dans la figure 12-1, pour contrôler le NXT.

Tableau 12-1. Composants non visibles pour l'application de contrôle Robot NXT

Type de composant	Groupe de palettes	Comment tu l'appelleras	But
Client Bluetooth	Connectivité	Client Bluetooth1	Connectez-vous au NXT.
NxtDrive	LEGO® MINDSTORMS® NxtDrive1		Conduisez les roues du robot.
NxtUltrasonicSensor LEGO® MINDSTORMS® NxtUltrasonicSensor1	Détecte les obstacles.		
Notifiant	Interface utilisateur	Notifiant1	Afficher les messages d'erreur.



Figure 12-2. Les composants non visibles affichés en bas du composant Designer

Définissez les propriétés des composants de la manière suivante :

1. Définissez la propriété BluetoothClient de NxtDrive1 et NxtUltrasonicSensor1 sur «Client Bluetooth1».
2. Vérifiez BelowRangeEventEnabled sur NxtUltrasonicSensor1.
3. Définissez la propriété DriveMotors de NxtDrive1 :
 - Si votre robot a le moteur de la roue gauche connecté au port de sortie C et Le moteur de la roue droite connecté au port de sortie B, le réglage par défaut de « CB » n'a pas besoin d'être modifié.
 - Si votre robot est configuré différemment, définissez la propriété DriveMotors sur un valeur de texte à deux lettres, où la première lettre est le port de sortie connecté à

le moteur de la roue gauche et la deuxième lettre est le port de sortie connecté au moteur de la roue droite.

4. Définissez la propriété SensorPort de NxtUltrasonicSensor1.
 - Si le capteur à ultrasons de votre robot est connecté au port d'entrée 4, la valeur par défaut le réglage de « 4 » n'a pas besoin d'être modifié.
 - Si votre robot est configuré différemment, définissez la propriété SensorPort sur le port d'entrée connecté au capteur à ultrasons.

COMPOSANTS VISIBLES

Créons ensuite les composants de l'interface utilisateur illustrés à la figure 12-2.



Figure 12-3. L'application dans le Concepteur de composants

Pour établir la connexion Bluetooth, vous aurez besoin de l'adresse Bluetooth unique du NXT. Malheureusement, les adresses Bluetooth sont constituées de huit nombres hexadécimaux à deux chiffres (une manière de représenter des valeurs binaires) séparés par des deux-points, ce qui les rend très difficiles à saisir. Vous ne voudrez pas saisir l'adresse sur votre téléphone à chaque fois que vous exécuterez l'application. Donc, pour éviter cela, vous utiliserez un ListPicker qui affiche une liste des robots qui ont été couplés à votre téléphone et vous permettra d'en choisir un. Vous utiliserez des boutons pour avancer et reculer, tourner à gauche et à droite, vous arrêter et

déconnexion. Vous pouvez utiliser un VerticalArrangement pour tout disposer sauf pour le ListPicker et un HorizontalArrangement pour contenir les boutons pour tourner à gauche, s'arrêter et tourner à droite.

Vous pouvez créer l'interface illustrée dans la figure 12-2 en faisant glisser les composants répertoriés dans le tableau 12-2.

Tableau 12-2. Composants visibles pour l'application de contrôle Robot NXT

Type de composant	Groupe de palettes	Comment vous l'appellerez	But
Sélecteur de liste	Interface utilisateur	ConnectListPicker	Choisissez le robot auquel vous connecter.
Disposition de l'arrangement vertical		VerticalArrangement1	Un conteneur visuel.
Bouton	Bouton Suivant de l'interface utilisateur		Avancez.
Disposition de l'arrangement horizontal		HorizontalArrangement1	Un conteneur visuel.
Bouton	Bouton gauche de l'interface utilisateur		Tourner à gauche.
Bouton	Bouton d'arrêt de l'interface utilisateur		Arrêt.
Bouton	Bouton droit de l'interface utilisateur		Tournez à droite.
Bouton	Bouton arrière de l'interface utilisateur		Reculez.
Bouton	Bouton de déconnexion de l'interface utilisateur		Déconnectez-vous du NXT.

Pour organiser la disposition visuelle comme indiqué dans la Figure 12-2, placez LeftButton, StopButton, et RightButton à l'intérieur de HorizontalArrangement1, et placez ForwardButton, HorizontalArrangement1, BackwardButton et DisconnectButton à l'intérieur Disposition verticale1.

Définissez les propriétés des composants comme suit :

1. DÉCOchez Défilable sur Screen1.
2. Définissez la largeur de ConnectListPicker et DisconnectButton sur « Fill parent ».
3. Définissez la largeur et la hauteur de VerticalArrangement1, ForwardButton, HorizontalArrangement1, LeftButton, StopButton, RightButton et BackwardButton pour « Remplir le parent ».
4. Définissez le texte de ConnectListPicker sur « Connect... ».
5. Définissez le texte de ForwardButton sur « ^ ».
6. Définissez le texte de LeftButton sur « < ».
7. Définissez le texte de StopButton sur « - ».
8. Définissez le texte de RightButton sur « > ».

9. Définissez le texte de BackwardButton sur « v ».
10. Définissez le texte de DisconnectButton sur « Disconnect ».
11. Définissez la taille de police de ConnectListPicker et DisconnectButton sur 30.
12. Définissez la taille de police de ForwardButton, LeftButton, StopButton, RightButton et Bouton Backward à 40.

Dans cette application, il est logique de masquer la majeure partie de l'interface utilisateur jusqu'à ce que le Bluetooth soit connecté au NXT. Pour ce faire, définissez la propriété Visible de VerticalArrangement1 sur masqué. Ne vous inquiétez pas, dans un instant, nous ferons en sorte que l'application révèle l'interface utilisateur après sa connexion au NXT.

Ajout de comportements aux composants

Dans cette section, vous allez programmer le comportement de l'application, notamment :

- Permettre à l'utilisateur de connecter l'application à un robot en le choisissant dans une liste.
- Permettre à l'utilisateur de déconnecter l'application d'un robot.
- Laisser l'utilisateur conduire le robot à l'aide des boutons de commande.
- Forcer le robot à s'arrêter lorsqu'il détecte un obstacle.

CONNEXION AU NXT

Le premier comportement que vous ajouterez est la connexion au NXT. Lorsque vous cliquez sur ConnectListPicker, une liste des robots couplés s'affiche. Lorsque vous choisissez un robot, l'application établira une connexion Bluetooth avec ce robot.

AFFICHAGE DE LA LISTE DES ROBOTS

Pour afficher la liste des robots, vous utiliserez ConnectListPicker. Un ListPicker ressemble à un bouton, mais lorsque vous cliquez dessus, il affiche une liste d'éléments parmi lesquels vous pouvez en choisir un.

Vous utiliserez le bloc BluetoothClient1.AddressesAndNames pour fournir une liste des adresses et des noms des appareils Bluetooth qui ont été associés à Android.

Étant donné que BluetoothClient1 est utilisé avec les composants NXT, il limite automatiquement les appareils inclus dans la propriété AddressesAndNames à ceux qui sont des robots, de sorte que vous ne verrez pas d'autres types d'appareils Bluetooth (tels que des casques) dans la liste. Le tableau 12-3 répertorie les blocs dont vous aurez besoin pour cette étape.

Tableau 12-3. Blocs pour ajouter un ListPicker à l'application

Type de bloc	Tiroir	But
ConnectListPicker.BeforePicking	ConnectListPicker	Déclenché lorsque ConnectListPicker est cliqué.
définissez ConnectListPicker.Elements sur ConnectListPicker		Définissez les choix qui apparaîtront.
BluetoothClient1.AdressesAndNames BluetoothClient1		Les adresses et les noms des robots associés à Android.

Comment fonctionnent les blocs

Lorsque vous cliquez sur ConnectListPicker , l'événement ConnectListPicker.BeforePicking est déclenché avant que la liste de choix ne soit affichée, comme le montre la figure 12-3. Pour spécifier les éléments qui seront répertoriés, définissez la propriété ConnectListPicker.Elements sur le bloc BluetoothClient1.AddressesAndNames . ConnectListPicker listera les robots qui ont été couplés avec Android.

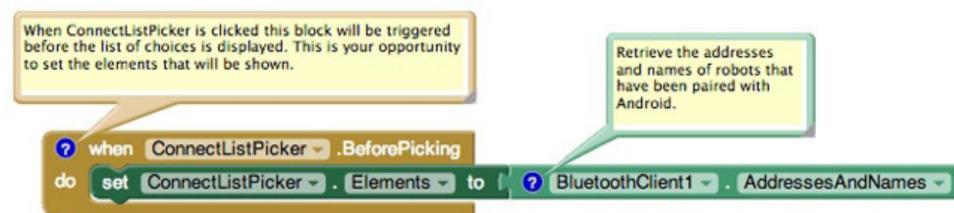


Figure 12-4. Afficher la liste des robots



Testez votre application Sur votre téléphone, cliquez sur « Connecter... » et voyez ce qui se passe. Vous devriez voir une liste de tous les robots avec lesquels votre téléphone a été associé. Si vous voyez juste un écran noir, votre téléphone n'a été associé à aucun robot. Si vous voyez des adresses et des noms d'autres appareils Bluetooth, tels qu'un casque Bluetooth, la propriété BluetoothClient de NxtDrive1 et NxtUltrasonicSensor1 n'a pas été définie correctement.

ÉTABLIR LA CONNEXION BLUETOOTH

Après avoir choisi un robot dans la liste, l'application se connectera à ce robot via Bluetooth. Si la connexion réussit, l'interface utilisateur changera. ConnectListPicker sera masqué et le reste des composants de l'interface utilisateur apparaîtra. Si le robot n'est pas allumé, la connexion échouera et un message d'erreur apparaîtra. Vous utiliserez le bloc BluetoothClient1.Connect pour établir la connexion. Le

La propriété ConnectListPicker.Selection fournit l'adresse et le nom de la personne choisie. robot. Vous utiliserez un bloc if then pour tester si la connexion a réussi. Bien ajoutez else au bloc if then , il aura donc trois zones différentes où les blocs sont connecté : si, alors et sinon. La zone if contiendra le BluetoothClient1.Connect bloc. La zone then contiendra les blocs à exécuter si la connexion est réussie. La zone else contiendra les blocs à exécuter en cas d'échec de la connexion.

Si la connexion réussit, vous utiliserez la propriété Visible pour masquer ConnectListPicker et affichez VerticalArrangement1, qui contient le reste du composants de l'interface utilisateur. Si la connexion échoue, vous utiliserez le Bloc Notifier1.ShowAlert pour afficher un message d'erreur. Le tableau 12-4 répertorie les blocs vous aurez besoin pour ce comportement.

Tableau 12-4. Blocs pour utiliser Bluetooth pour se connecter au robot

Type de bloc	Tiroir	But
ConnectListPicker.AfterPicking ConnectListPicker		Déclenché lorsqu'un robot est choisi parmi ConnectListPicker.
si donc	Contrôle	Testez si la connexion Bluetooth est réussie.
BluetoothClient1.Connect	Client Bluetooth1	Connectez-vous au robot.
ConnectListPicker.Selection	ConnectListPicker	L'adresse et le nom du robot choisi.
définir ConnectListPicker.Visible sur	ConnectListPicker	Masquer ConnectListPicker.
FAUX	Logique	Branchez-le sur l'ensemble ConnectListPicker.Visible.
ensemble ArrangementVertical1.Visible à	VerticalArrangement1	Afficher le reste de l'interface utilisateur.
vrai	Logique	Branchez-le sur l'ensemble VerticalArrangement1.Visible sur.
Notifier1.ShowAlert	Notifier1	Afficher un message d'erreur.
texte « Impossible d'établir une connexion Bluetooth » connexion."	Texte	Le message d'erreur.

Comment fonctionnent les blocs

Une fois qu'un robot est sélectionné, l' événement ConnectListPicker.AfterPicking est déclenché, comme illustré à la figure 12-4. Le bloc BluetoothClient1.Connect rend le Bluetooth connexion au robot sélectionné. Si la connexion réussit, les blocs then sont exécuté : la propriété ConnectListPicker.Visible est définie sur false pour masquer ConnectListPicker et la propriété VerticalArrangement1.Visible est définie sur true pour

show VerticalArrangement1, qui contient les boutons de la télécommande. Si la connexion échoue, les blocs else sont exécutés : le bloc Notifier1.ShowAlert affiche un message d'erreur.

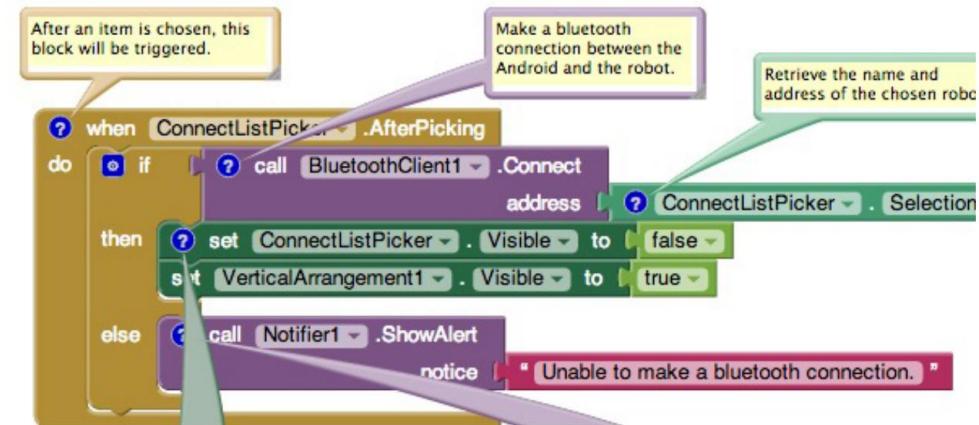


Figure 12-5. Établir la connexion Bluetooth

DÉCONNEXION DU NXT

Vous êtes probablement impatient de connecter votre Android à votre NXT, mais avant de faire cela, faisons encore une chose : ajoutons le comportement de déconnexion. De cette façon, vous êtes capable de tester à la fois la connexion et la déconnexion. Lorsque DisconnectButton est cliqué, l'application fermera la connexion Bluetooth et l'interface utilisateur changement. ConnectListPicker réapparaîtra et le reste de l'interface utilisateur les composants seront masqués. Utilisez les blocs répertoriés dans le tableau 12-5 pour construire le Bloc BluetoothClient1.Disconnect qui ferme la connexion Bluetooth. Vous serez utilisiez la propriété Visible pour afficher ConnectListPicker et masquer VerticalArrangement1, qui contient le reste des composants de l'interface utilisateur.

Tableau 12-5. Blocs de déconnexion du robot

Type de bloc	Tiroir	But
DisconnectButton.Click	Bouton Déconnecter	Déclenché lorsque DisconnectButton est cliqué.
BluetoothClient1.Déconnexion	Client Bluetooth1	Déconnectez-vous du robot.
définir ConnectListPicker.Visible à	ConnectListPicker	Afficher ConnectListPicker.
vrai	Logique	Branchez-le sur l'ensemble ConnectListPicker.Visible.

Type de bloc	Tiroir	But
ensemble ArrangementVertical1.Visible à	VerticalArrangement1	Masquer le reste de l'interface utilisateur.
FAUX	Logique	Branchez-le sur l'ensemble VerticalArrangement1.Visible sur.

Lorsque l'utilisateur clique sur DisconnectButton , l' événement DisconnectButton.Clicked est déclenché, comme le montre la figure 12-5. Le bloc BluetoothClient1.Disconnect ferme le Connexion Bluetooth. La propriété ConnectListPicker.Visible est définie sur true pour afficher ConnectListPicker et la propriété VerticalArrangement1.Visible est définie sur false pour masquer VerticalArrangement1, qui contient les boutons de la télécommande.

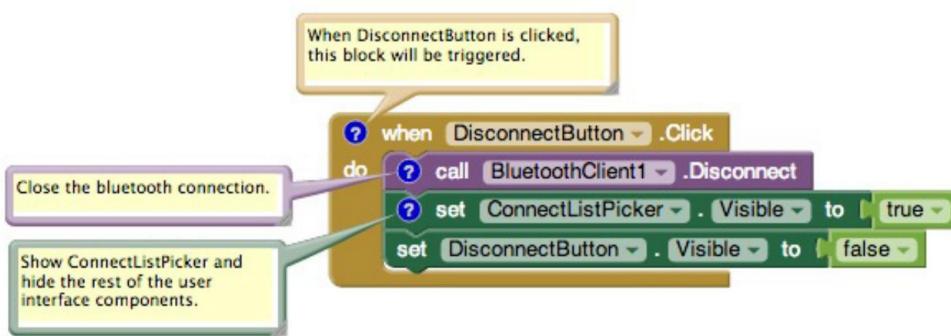


Figure 12-6. Déconnexion du robot



Testez votre application Assurez-vous que votre robot est allumé puis, sur votre téléphone, cliquez sur « Connecter... » et choisissez le robot que vous auquel vous souhaitez vous connecter. Il faudra un moment pour faire le Connexion Bluetooth. Une fois le robot connecté, vous devriez voir les boutons de contrôle du robot, ainsi que le bouton Déconnexion bouton. Cliquez sur le bouton Déconnecter. Les boutons pour le contrôle du robot devrait disparaître et le Connect bouton devrait réapparaître.

Conduire le NXT

Passons à la partie la plus amusante : ajouter un comportement pour avancer et reculer, tourner à gauche et à droite et s'arrêter. N'oubliez pas d'arrêter : si vous le faites, vous

ayez un robot incontrôlable entre vos mains ! Le composant NxtDrive fournit cinq blocs pour entraîner les moteurs du robot :

- MoveForwardIndefinite fait avancer les deux moteurs.
- MoveBackwardIndefinitely fait reculer les deux moteurs.
- TurnCounterClockwiseIndefinite fait tourner le robot vers la gauche en conduisant le moteur droit vers l'avant et le moteur gauche vers l'arrière.
- TurnClockwiseIndefinite fait tourner le robot vers la droite en conduisant vers la gauche moteur en avant et le moteur droit en arrière.
- Stop arrête les deux moteurs.

Les blocs Move... et Turn... ont chacun un paramètre appelé Power. Vous utiliserez un bloc numérique, ainsi que tous les autres éléments répertoriés dans le tableau 12-6, pour spécifier le quantité de puissance que le robot doit utiliser pour faire tourner les moteurs. La valeur peut varier de 0 à 100. Cependant, si vous spécifiez trop peu de puissance, les moteurs émettront un gémissement mais pas tourner. Dans cette application, vous utiliserez 90 (pour cent).

Tableau 12-6. Blocs pour contrôler le robot

Type de bloc	Tiroir	But
ForwardButton.Cliquez	Bouton Suivant	Déclenché lorsque ForwardButton est cliqué.
NxtDrive1.MoveForwardIndefinitely	NxtDrive1	Faites avancer le robot.
numéro (90)	Mathématiques	La quantité de puissance.
BackwardButton.Cliquez	Bouton Retour	Déclenché lorsque BackwardButton est cliqué.
NxtDrive1.MoveBackwardIndefinitely	NxtDrive1	Faites reculer le robot.
numéro (90)	Mathématiques	La quantité de puissance.
LeftButton.Cliquez	Bouton gauche	Déclenché lorsque LeftButton est cliqué.
NxtDrive1.TurnCounterClockwise Indéfiniment	NxtDrive1	Tournez le robot dans le sens inverse des aiguilles d'une montre.
numéro (90)	Mathématiques	La quantité de puissance.
Bouton Droit.Clic	Bouton de droite	Déclenché lorsque RightButton est cliqué.
NxtDrive1.TurnClockwiseIndefinitely	NxtDrive1	Tournez le robot dans le sens des aiguilles d'une montre.
numéro (90)	Mathématiques	La quantité de puissance.

214 Chapitre 12 : Robot à distance

Type de bloc	Tiroir	But
StopButton.Cliquez	Bouton Arrêt	Déclenché lorsque StopButton est cliqué.
NxtDrive1.Stop	NxtDrive1	Arrêtez le robot.

Comment fonctionnent les blocs

Lorsque l'utilisateur clique sur ForwardButton , l'événement ForwardButton.Clicked est déclenché. Le bloc NxtDrive1.MoveForwardIndefinitely illustré à la figure 12-6 est utilisé pour déplacer le robot en avant à 90% de sa puissance. Les événements restants fonctionnent de la même manière pour les autres boutons, chacun alimentant le robot vers l'arrière, la gauche et la droite.

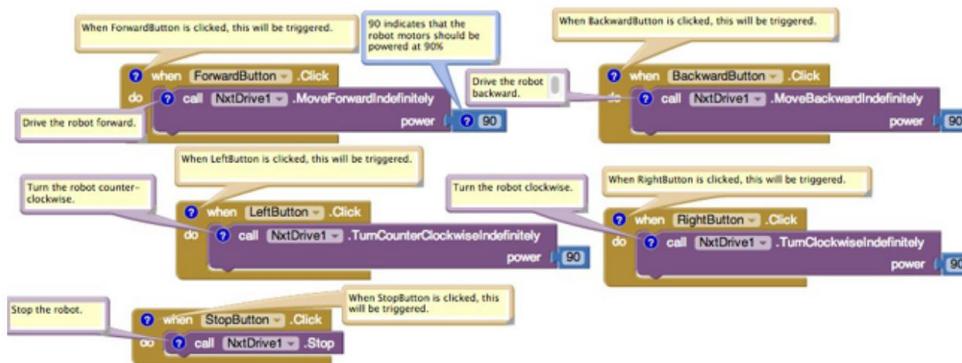


Figure 12-7. Conduire le robot

Lorsque l'utilisateur clique sur StopButton , l'événement StopButton.Clicked est déclenché. Le bloc NxtDrive1.Stop est utilisé pour arrêter le robot. Testez votre application. Suivez les instructions dans la section précédente « Testez votre application » pour vous connecter au NXT. Assurez-vous que le robot est pas sur une table où il pourrait tomber, puis testez son comportement comme suit :

1. Cliquez sur le bouton Suivant. Le robot devrait avancer.
2. Cliquez sur le bouton Précédent. Le robot devrait reculer.
3. Cliquez sur le bouton gauche. Le robot doit tourner dans le sens inverse des aiguilles d'une montre.
4. Cliquez sur le bouton droit. Le robot doit tourner dans le sens des aiguilles d'une montre.
5. Cliquez sur le bouton d'arrêt. Le robot devrait s'arrêter.

Si votre robot ne bouge pas, mais que vous entendez un gémississement, vous devrez peut-être augmenter la puissance. Vous pouvez en utiliser 100 pour une puissance maximale.

Utilisation du capteur à ultrasons pour détecter les obstacles

À l'aide du capteur à ultrasons, le robot s'arrêtera s'il rencontre un obstacle dans un rayon de 30 centimètres, comme celui illustré à la figure 12-7.

Vous pouvez utiliser le composant NxtUltrasonicSensor pour détecter les obstacles. Il possède deux propriétés nommées BottomOfRange et TopOfRange qui définissent la plage de détection en centimètres. Par défaut, la propriété BottomOfRange est définie sur 30 centimètres et TopOfRange sur 90 centimètres.

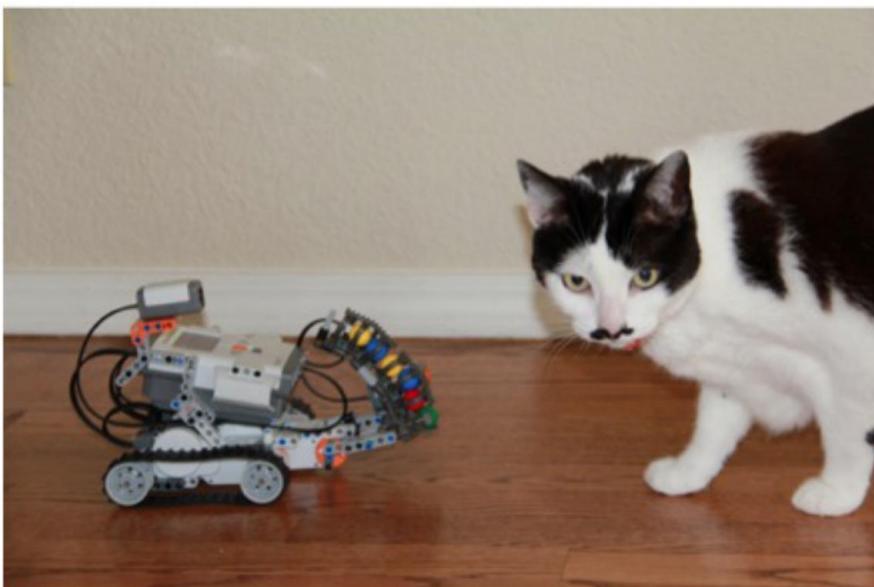


Figure 12-8. Un obstacle domestique courant pour votre robot NXT

Le composant NxtUltrasonicSensor comporte également trois événements appelés BelowRange, WithinRange et AboveRange. L' événement BelowRange sera déclenché lorsqu'un obstacle est détecté à une distance inférieure à BottomOfRange. L' événement WithinRange sera déclenché lorsqu'un obstacle est détecté à une distance entre BottomOfRange et TopOfRange. L' événement AboveRange sera déclenché lorsqu'un obstacle est détecté à une distance supérieure à TopOfRange.

Vous utiliserez le bloc d'événements NxtUltrasonicSensor1.BelowRange , présenté dans le tableau 12-7, pour détecter un obstacle dans un rayon de 30 centimètres. Si vous souhaitez détecter un obstacle à une distance différente, vous pouvez ajuster la propriété BottomOfRange . Vous utiliserez le bloc NxtDrive1.Stop pour arrêter le robot.

Tableau 12-7. Blocs d'utilisation du NxtUltrasonicSensor

Type de bloc	Tiroir	But
un obstacle NxtUltrasonicSensor1.BelowRange NxtUltrasonicSensor1 à une		Déclenché lorsque le capteur à ultrasons détecte distance inférieure à 30 centimètres.
NxtDrive1.Stop	NxtDrive1	Arrêtez le robot.

Comment fonctionnent les blocs

Lorsque le capteur à ultrasons du robot détecte un obstacle à une distance inférieure à 30 centimètres, l' événement NxtUltrasonicSensor1.BelowRange est déclenché, comme le montre la figure 12-8. Le bloc NxtDrive1.Stop arrête le robot.

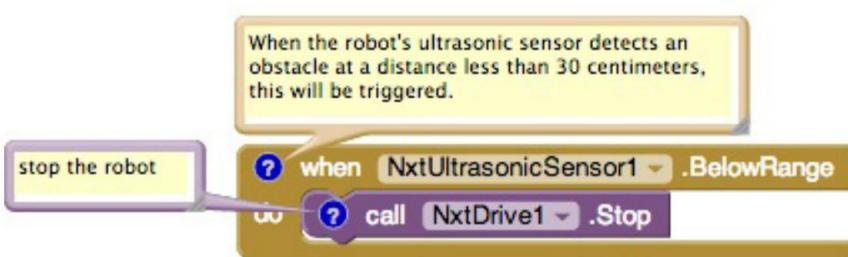


Figure 12-9. Déetecter un obstacle et arrêter le robot



Testez votre application Suivez les instructions de la section précédente « Testez votre application » pour vous connecter au NXT. À l'aide des boutons de navigation, conduisez votre robot vers un obstacle, comme un chat. Le robot doit s'arrêter lorsqu'il s'approche à moins de 30 centimètres du chat.

Si le robot ne s'arrête pas, le chat s'est peut-être éloigné du robot avant lui. dessiné dans les 30 centimètres. Vous devrez peut-être tester votre application avec un obstacle inanimé.

Variantes

Une fois que cette application fonctionne (et que vous avez passé suffisamment de temps à jouer avec votre robot NXT), vous souhaiterez peut-être essayer ce qui suit :

- Variez la quantité de puissance lorsque vous conduisez le robot.

- Vous pouvez le faire en modifiant la valeur numérique que vous branchez dans le Blocs MoveForwardIndefinitely, MoveBackwardIndefinitely, TurnCounterclockIndefinitely et TurnClockwiseIndefinitely .
- Utilisez le NxtColorSensor pour allumer une lumière rouge lorsqu'un obstacle est détecté.
 - Vous pouvez utiliser un composant NxtColorSensor et son GenerateColor propriété.
 - Vous devrez définir la propriété DetectColor sur false (ou la décocher dans le Concepteur de composants) car le capteur de couleur ne peut pas détecter et générer de la couleur en même temps.
- Utilisez un OrientationSensor pour contrôler le robot.
- Utilisez des éléments de construction LEGO pour attacher physiquement votre téléphone au robot. Créez des applications qui rendent le robot autonome.

Résumé

Voici quelques-uns des concepts que nous avons abordés dans ce didacticiel :

- Vous pouvez utiliser le composant ListPicker pour choisir parmi une liste de robots appariés.
- Le composant BluetoothClient établit la connexion avec le robot.
- Le composant Notifier affiche un message d'erreur.
- Vous pouvez utiliser la propriété Visible pour masquer ou afficher les composants de l'interface utilisateur.
- Le composant NxtDrive peut conduire, tourner et arrêter le robot.
- Vous pouvez utiliser le composant NxtUltrasonicSensor pour détecter les obstacles.

Amazon à la Librairie

Supposons que vous parcouriez des livres dans votre librairie préférée et que vous souhaitez savoir combien coûte un prix des livres sur Amazon.com. Avec l'Amazonie à

Dans l'application Librairie, vous pouvez numériser un livre ou saisir un ISBN, et l'application vous indiquera le prix le plus bas actuel du livre sur Amazon.com. Vous pouvez également rechercher des livres sur un sujet particulier.

Amazon à la librairie démontre comment vous pouvez utiliser App Inventor pour créer des applications qui communiquent avec des services Web (c'est-à-dire des interfaces de programmation d'applications ou API). Cette application obtiendra les données d'un service Web créé par l'un des auteurs de ce livre. À la fin de ce chapitre, vous serez en mesure de créer votre propre application personnalisée pour parler à Amazon. L'application dispose d'une interface utilisateur simple avec laquelle l'utilisateur peut saisir des mots-clés ou l'ISBN d'un livre (numéro de livre standard international : un code à 10 ou 13 chiffres qui identifie de manière unique un livre), puis répertorier le titre, l'ISBN et le prix le plus bas pour un nouvel exemplaire chez Amazon. Il utilise également le composant BarcodeScanner afin que l'utilisateur puisse numériser un livre pour déclencher une recherche au lieu de saisir du texte (techniquement, le scanner saisit simplement l'ISBN du livre pour vous).

Figure 13-1.



Ce que vous apprendrez

Dans cette application (illustré dans la figure 13-1), vous apprendrez :

- Comment utiliser un lecteur de codes-barres dans une application.
- Comment accéder à une source d'informations Web (API d'Amazon) via TinyWebDB composant.
- Comment traiter les données complexes renvoyées par cette source d'informations Web. En particulier, vous apprendrez à traiter une liste de livres dans laquelle chaque livre est lui-même une liste de trois éléments (titre, prix et ISBN).

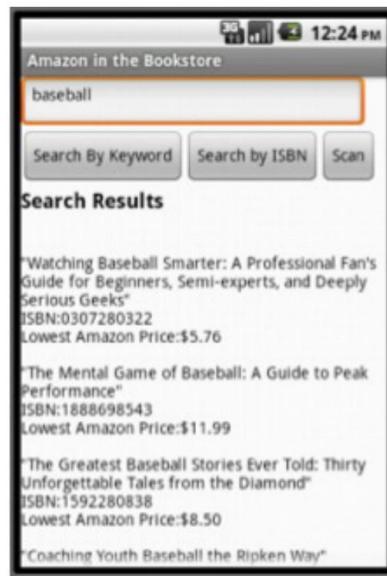


Figure 13-2. Amazon à la librairie fonctionnant dans l'émulateur

Vous serez également initié au code source que vous pouvez utiliser pour créer votre propre site Web. API de service avec le langage de programmation Python et App Engine de Google.

Qu'est-ce qu'une API ?

Avant de commencer à concevoir vos composants et à programmer l'application, examinons de plus près ce qu'est une interface de programmeur d'application (API) et comment elle fonctionne. Une API est comme un site Web, mais au lieu de communiquer avec des humains, elle communique avec d'autres programmes informatiques. Les API sont souvent appelées programmes « serveur » car elles fournissent généralement des informations aux programmes « clients » qui interagissent réellement avec les humains, comme une application App Inventor. Si vous avez déjà utilisé une application Facebook sur votre téléphone, vous utilisez un programme client qui communique avec l'application serveur API Facebook.

Dans ce chapitre, vous allez créer une application client Android qui communique avec une API Amazon. Votre application demandera des informations sur le livre et l'ISBN à l'API Amazon, et l'API renverra des listes à jour à votre application. L'application présentera ensuite les données du livre à l'utilisateur.

L'API Amazon que vous utiliserez est spécialement configurée pour être utilisée avec App Inventor. Nous Je n'entrerai pas dans les détails sanglants ici, mais il est utile de savoir qu'à la suite de cette configuration, vous pouvez utiliser le composant TinyWebDB pour communiquer avec Amazon.

La bonne nouvelle est que vous savez déjà comment faire ça ! Vous appellerez TinyWebDB.GetValue pour demander des informations, puis traiterez les informations renvoyées dans le gestionnaire d'événements TinyWebDB.GetValue , comme vous le faites lorsque vous utilisez une base de données Web. (Vous pouvez revenir aux applications MakeQuiz et TakeQuiz au chapitre 10 pour vous rafraîchir la mémoire, si nécessaire.)

Avant de créer l'application, vous devez comprendre le protocole de l'API Amazon, qui spécifie le format de votre demande et le format des données renvoyées. Tout comme différentes cultures humaines ont des protocoles différents (lorsque vous rencontrez quelqu'un, est-ce que vous serrez la main, vous inclinez ou hochez la tête ?), les ordinateurs qui communiquent entre eux ont également des protocoles. L'API Amazon que vous utiliserez ici fournit une interface Web permettant d'explorer le fonctionnement de l'API avant de commencer à l'utiliser. Bien que l'API soit conçue pour communiquer avec d'autres ordinateurs, cette interface Web vous permet de voir exactement comment cette communication se déroulera. En suivant ces étapes, vous pouvez essayer quels appels GetValue particuliers renverront via le site Web et savoir que l'interface API se comportera exactement de la même manière lorsque vous lui demanderez des données via le composant TinyWebDB dans App Inventor. Commençons :

1. Ouvrez un navigateur et accédez à <http://aiamazonapi.appspot.com/>. Vous verrez le site Web illustré à la figure 13-2.

222 Chapitre 13 : Amazon à la librairie



Figure 13-3. L'interface Web de l'API Amazon d'App Inventor

2. Sur cette page Web, vous pouvez essayer la seule fonction que vous pouvez appeler avec cette API : obtenir de la valeur. Entrez un terme (par exemple, « baseball ») dans le champ Balise, puis cliquez sur « Obtenir la valeur ». La page Web affichera une liste des cinq meilleurs livres renvoyés par Amazon, comme le montre la figure 13-3.

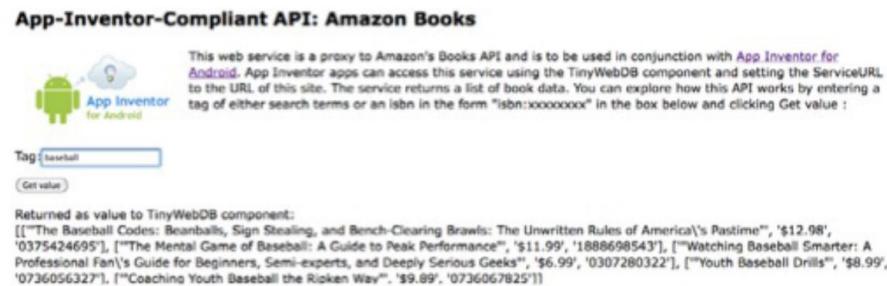


Figure 13-4. Faire un appel à l'API Amazon pour rechercher des livres liés au tag (ou mot-clé) « baseball »

La valeur renvoyée est une liste de livres, chacun entre parenthèses [comme ceci] et en fournissant le titre, le coût et l'ISBN.

Si vous regardez bien, vous verrez que chaque livre est en fait représenté comme une sous-liste d'une autre liste principale. La liste principale (sur le baseball) est placée entre parenthèses, et chaque sous-liste (ou livre) est entourée de son propre ensemble de parenthèses à l'intérieur des parenthèses principales. Ainsi, la valeur de retour de cette API est en fait une liste de listes, chaque sous-liste fournissant les informations d'un livre. Regardons cela d'un peu plus près. Chaque crochet gauche ([]) dans les données indique le début d'une liste. Le premier crochet gauche du résultat désigne le début de la liste externe (la liste des livres). À sa droite immédiate se trouve le début de la première sous-liste, le premier livre, comme démontré ici :

[« Les codes du baseball : balles de haricots, vols de pancartes et bagarres pour nettoyer les bancs : Les règles non écrites du passe-temps américain », « 12,98 \$ », « 0375424695 »]

La sous-liste comprend trois parties : un titre, le prix actuel le plus bas du livre sur Amazon et l'ISBN du livre. Lorsque vous obtenez ces informations dans votre application App Inventor, vous pourrez accéder à chaque pièce en utilisant l'élément de liste de sélection, avec l'index 1 pour le titre, l'index 2 pour le prix et l'index 3 pour l'ISBN. (Pour vous rafraîchir la mémoire sur l'utilisation d'un index et de listes, revisitez l'application MakeQuiz au chapitre 10.)

3. Au lieu de rechercher par mot-clé, vous pouvez rechercher un livre en saisissant un ISBN. Pour effectuer une telle recherche, vous saisissez une balise sous la forme « isbn : 000000000000 », où la liste de 0 représente un numéro ISBN réel (voir Figure 13-4). Les doubles crochets ([[]) dans le résultat [['« App Inventor »,' '\$21.93,' '1449397484']] indiquent qu'une liste de listes est toujours renvoyée, même s'il n'y a qu'un seul livre. Cela peut sembler un peu étrange maintenant, mais cela sera important lorsque nous accéderons aux informations de notre application.

App-Inventor-Compliant API: Amazon Books

This web service is a proxy to Amazon's Books API and is to be used in conjunction with [App Inventor for Android](#). App Inventor apps can access this service using the TinyWebDB component and setting the ServiceURL to the URL of this site. The service returns a list of book data. You can explore how this API works by entering a tag of either search terms or an isbn in the form "isbn:xxxxxx" in the box below and clicking Get value :

Tag:

Returned as value to TinyWebDB component:
[[{"App Inventor": "\$21.93", "1449397484"}]]

Figure 13-5. Interrogation de l'API Amazon par ISBN au lieu d'un mot-clé

Commencer

Connectez-vous au site Web App Inventor et démarrez un nouveau projet. Ensuite, nommez-le « AmazonBooks » et définissez le titre de l'écran sur « Amazon à la librairie ». Ensuite, connectez votre appareil ou émulateur pour des tests en direct.

Conception des composants

L'interface utilisateur de l'application Amazon Book est relativement simple : donnez-lui une zone de texte pour saisir des mots-clés ou des ISBN, deux boutons pour lancer les deux types de recherches (mot-clé ou ISBN) et un troisième bouton pour permettre à l'utilisateur de numériser un livre (nous j'y reviendrai dans un instant). Ensuite, ajoutez une étiquette d'en-tête et une autre étiquette pour répertorier les résultats renvoyés par l'API Amazon, et enfin deux composants non visibles : TinyWebDB et un BarcodeScanner. Vérifiez vos résultats par rapport à la figure 13-5.

Commencer

224 Chapitre 13 : Amazon à la librairie

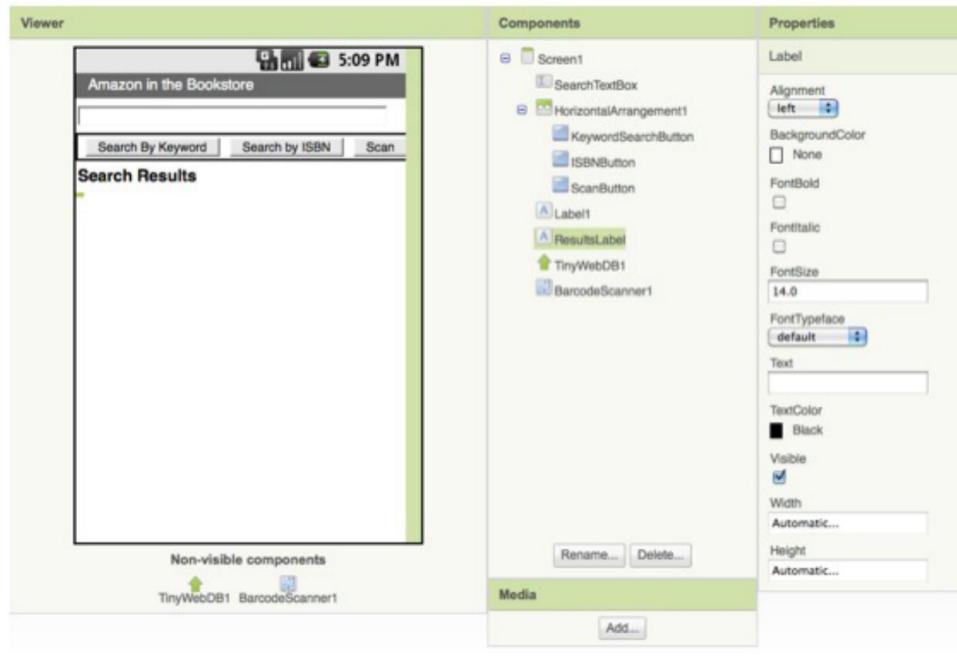


Figure 13-6. L'interface utilisateur d'Amazon dans la librairie affichée dans le concepteur

Le tableau 13-1 répertorie tous les composants dont vous aurez besoin pour créer l'interface utilisateur illustrée à la figure 13-5.

Tableau 13-1. Liste des composants pour l'application Amazon dans la librairie

Type de composant	Groupe de palettes	Comment vous l'appellerez	But
Zone de texte	Interface utilisateur	SearchTextBox	L'utilisateur saisit des mots-clés ou un ISBN ici.
Disposition de l'arrangement horizontal		HorizontalArrangement1	Disposez les boutons sur une ligne.
Bouton	Bouton de recherche	de mots-clés de l'interface utilisateur	Cliquez pour effectuer une recherche par mot-clé.
Bouton	Interface utilisateur	Bouton ISBN	Cliquez pour rechercher par ISBN.
Bouton	Stockage	Bouton de numérisation de l'interface utilisateur	Cliquez pour numériser l'ISBN d'un livre.
Étiquette		Étiquette de l'interface utilisateur1	L'en-tête « Résultats de la recherche ».
Étiquette		Étiquette de résultats de l'interface utilisateur	Où vous afficherez les résultats.
MinusculeWebDB	Stockage	MinusculeWebDB1	Parlez à Amazon.com.
Scanner de codes-barres	Capteurs	Scanner de codes-barres1	Scannez les codes-barres.

Définissez les propriétés des composants de la manière suivante :

1. Définissez l' indice de SearchTextBox sur « Entrez des mots-clés ou un ISBN ».

2. Définissez les propriétés des boutons et des étiquettes afin qu'ils apparaissent comme indiqué dans Figure 13-5.
3. Définissez la propriété ServiceURL du composant TinyWebDB sur `http://aiamazonapi.appspot.com/`.

Programmation du comportement de l'application

Pour cette application, vous allez spécifier les comportements suivants dans l'éditeur de blocs :

Recherche par mot-clé

L'utilisateur saisit certains termes et clique sur le KeywordSearchButton pour lancer une recherche Amazon. Vous appellerez `TinyWebDB1.GetValue` pour que cela se produise.

Recherche par ISBN

L'utilisateur saisit un ISBN et clique sur le bouton ISBN. Vous regrouperez le préfixe « isbn : » avec le numéro saisi et exécuterez la recherche Amazon.

Lecture de codes-barres

L'utilisateur clique sur un bouton et le scanner est lancé. Lorsque l'utilisateur scanne l'ISBN d'un livre, votre application lancera une recherche par ISBN.

Traitement de la liste des livres

Dans un premier temps, votre application affichera les données renvoyées par Amazon de manière rudimentaire. Plus tard, vous modifierez les blocs afin que l'application extraie le titre, le prix et l'ISBN de chaque livre renvoyé et les affiche de manière organisée.

RECHERCHE PAR MOT-CLÉ

Lorsque l'utilisateur clique sur KeywordSearchButton, vous souhaitez récupérer le texte de SearchTextBox et l'envoyer en tant que balise dans votre demande à l'API Amazon. Vous utiliserez le bloc `TinyWebDB.GetValue` pour demander la recherche Amazon. Lorsque les résultats reviendront d'Amazon, le gestionnaire d'événements `TinyWebDB.GetValue` sera déclenché. Pour l'instant, affichez simplement le résultat qui est renvoyé directement dans `ResultsLabel`, comme le montre la figure 13-6. Plus tard, après avoir constaté que les données sont effectivement récupérées, vous pouvez afficher les données de manière plus sophistiquée.

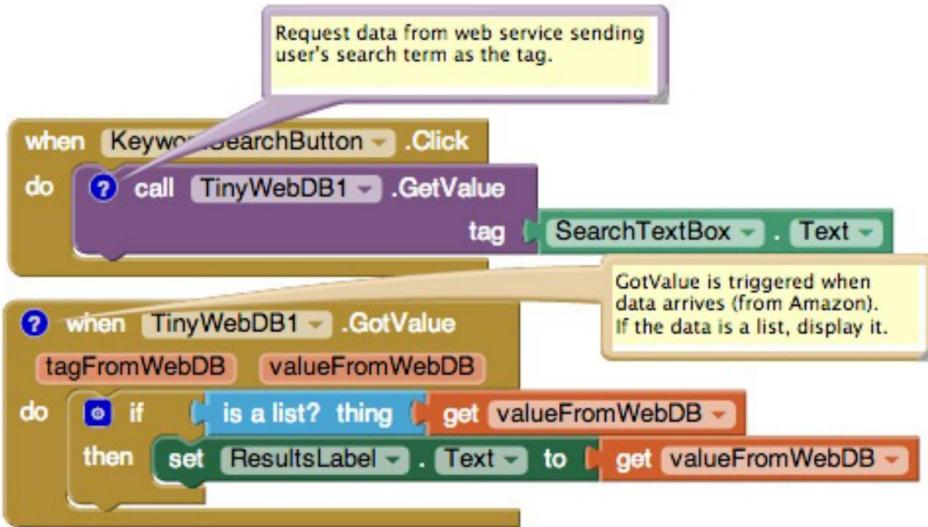


Figure 13-7. Envoyez la demande de recherche à l'API et placez les résultats dans ResultsLabel

Comment fonctionnent les blocs

Lorsque l'utilisateur clique sur KeywordSearchButton, la requête TinyWebDB1.GetValue est effectuée. La balise envoyée avec la demande correspond aux informations que l'utilisateur a saisies dans SearchTextBox. Si vous avez terminé l'application MakeQuiz (Chapitre 10), vous savez que les requêtes TinyWebDB1.GetValue ne reçoivent pas de réponse immédiate. Au lieu de cela, lorsque les données arrivent de l'API, TinyWebDB1.GotValue est déclenchée. Dans GotValue, les blocs vérifient la valeur renvoyée pour voir s'il s'agit d'une liste (ce ne sera pas le cas si l'API Amazon est hors ligne ou s'il n'y a pas de données pour les mots-clés). S'il s'agit d'une liste, les données sont placées dans ResultsLabel.



Testez votre application Saisissez un terme dans la zone de recherche et cliquez sur Rechercher par mot clé. Vous devriez obtenir une liste similaire à celle illustrée dans la figure 13-7. (Ce n'est pas très joli, mais nous y reviendrons sous peu.)

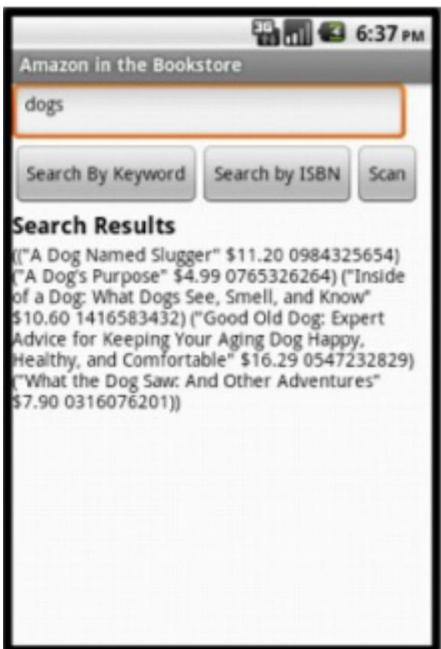


Figure 13-8. Résultat de recherche par mot-clé pour « chiens »

RECHERCHE PAR ISBN

Le code de recherche par ISBN est similaire, mais dans ce cas, l'API Amazon s'attend à ce que la balise soit sous la forme « isbn:xxxxxxxxxxxxxx » (c'est le protocole que l'API attend pour la recherche par ISBN). Vous ne voulez pas forcer l'utilisateur à connaître ce protocole ; l'utilisateur doit simplement pouvoir saisir l'ISBN dans la zone de texte et cliquer sur Rechercher par ISBN, et l'application doit ajouter le préfix « isbn : » en coulisses avec créer du texte. La figure 13-8 montre les blocs pour ce faire.

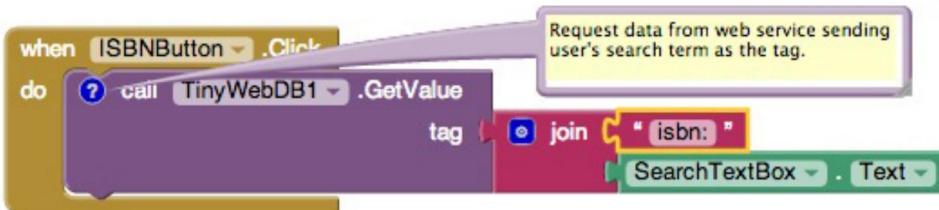


Figure 13-9. L'application ajoute « isbn : » à la recherche pour qu'elle recherche un livre particulier.

Comment fonctionnent les blocs

Le bloc de jointure concatène le préfixe « isbn : » avec les informations que l'utilisateur a saisies dans SearchTextBox et envoie le résultat sous forme de balise à TinyWebDB1.GetValue.

228 Chapitre 13 : Amazon à la librairie

Tout comme pour la recherche par mot-clé, l'API renvoie un résultat de liste pour une recherche ISBN : dans ce cas, une liste d'un seul élément dont l'ISBN correspond exactement à la saisie de l'utilisateur. Étant donné que le gestionnaire d'événements TinyWebDB.GetValue est déjà configuré pour traiter une liste de livres (même une liste contenant un seul élément), vous n'aurez pas besoin de modifier votre gestionnaire d'événements pour que cela fonctionne.



Testez votre application Saisissez un ISBN (par exemple, 9781449397487) dans le RechercheTextBox champ et cliquez sur le bouton ISBN informations sur le livre apparaissent-elles ?

NE LAISSEZ PAS VOS UTILISATEURS EN PENTE

Lorsque vous appelez un service Web (API) avec TinyWebDB1.GetValue, il peut y avoir un délai avant que les données n'arrivent et que TinyWebDB1.GetValue soit déclenché. C'est généralement une bonne idée d'informer les utilisateurs que la demande est en cours de traitement pour les rassurer sur le fait que l'application ne s'est pas bloquée. Pour cette application, vous pouvez placer un message dans ResultsLabel chaque fois que vousappelez TinyWebDB1.GetValue, comme le montre la figure 13-9.

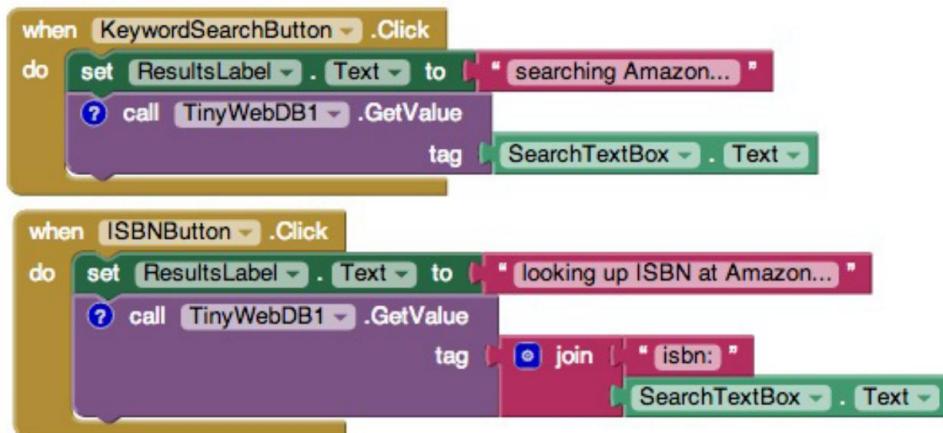


Figure 13-10. Ajout d'un message pour informer l'utilisateur de ce qui se passe

Comment fonctionnent les blocs

Pour les recherches par mot-clé et ISBN, un message « Recherche sur Amazon... » est placé dans ResultsLabel lorsque les données sont demandées. Notez que lorsque GetValue est déclenché, ce message est remplacé par les résultats réels d'Amazon.

NUMÉRISATION D'UN LIVRE

Soyons réalistes : taper sur un téléphone portable n'est pas toujours la chose la plus simple, et on a tendance à se tromper ici et là. Ce serait certainement plus facile (et entraînerait moins d'erreurs) si un utilisateur pouvait simplement lancer votre application et scanner le code-barres du livre. Il s'agit d'une autre excellente fonctionnalité intégrée du téléphone Android que vous pouvez facilement exploiter avec App Inventor.

La fonction `BarcodeScanner.DoScan` démarre le scanner. Tu voudras appeler ça lorsque le `ScanButton` est cliqué. Le gestionnaire d'événements `BarcodeScanner.AfterScan` est déclenché dès que quelque chose a été scanné. Il a un argument, `result`, qui contient les informations analysées. Dans ce cas, vous souhaitez lancer une recherche ISBN en utilisant ce résultat, comme le montre la figure 13-10.

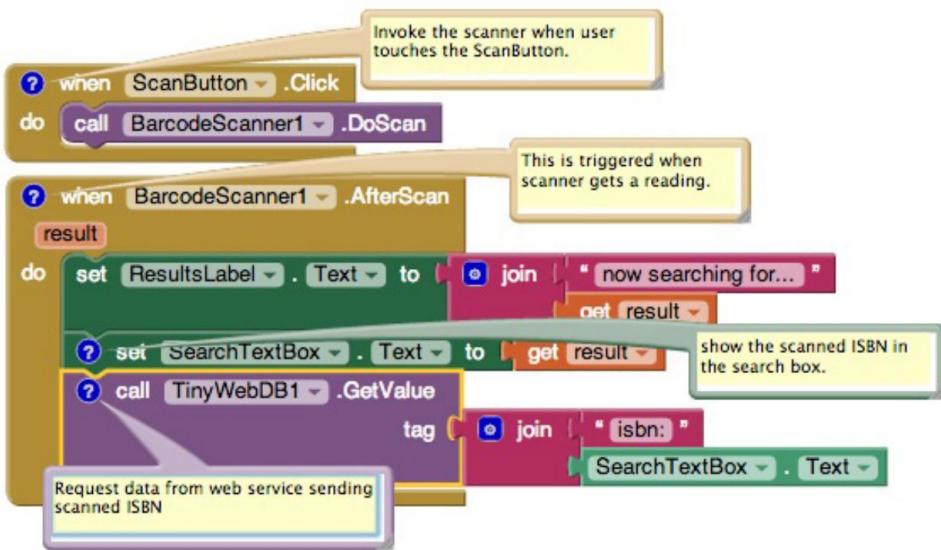


Figure 13-11. Blocs permettant de lancer une recherche ISBN après une numérisation par un utilisateur

Comment fonctionnent les blocs

Lorsque l'utilisateur clique sur le `ScanButton`, `DoScan` lance le scanner. Lorsqu'un élément a été analysé, `AfterScan` est déclenché. L'argument `result` contient le résultat de l'analyse, dans ce cas, l'ISBN d'un livre. L'utilisateur est informé qu'une demande a été effectuée, le résultat (le numéro ISBN numérisé) est placé dans `SearchTextBox` et `TinyWebDB1.GetValue` est appelé pour lancer la recherche. Encore une fois, le gestionnaire d'événements `TinyWebDB1.GotValue` traitera les informations du livre renvoyées.

230 Chapitre 13 : Amazon à la librairie



Testez votre application Cliquez sur le ScanButton et scannez le code-barres d'un livre. L'application affiche-t-elle les informations du livre ?

AMÉLIORER L'AFFICHAGE

Une application client comme celle que vous créez peut faire ce qu'elle veut avec les données qu'elle reçoit : vous pouvez comparer les informations de prix avec celles d'autres boutiques en ligne ou utiliser les informations de titre pour rechercher des livres similaires dans une autre bibliothèque. Presque toujours, vous souhaiterez que les informations de l'API soient chargées dans des variables que vous pourrez ensuite traiter davantage. Dans le gestionnaire d'événements TinyWebDB.GotValue que vous avez jusqu'à présent, vous placez simplement toutes les informations renvoyées par Amazon dans ResultsLabel . Au lieu de cela, traitons les données en 1) plaçant le titre, le prix et l'ISBN de chaque livre renvoyé dans des variables distinctes, et 2) affichant ces éléments de manière ordonnée. Si vous avez terminé certains des chapitres précédents, vous êtes probablement en train de maîtriser la définition des variables et de les utiliser dans votre affichage. Essayez donc de créer les variables dont vous pensez avoir besoin et les blocs pour afficher chaque résultat de recherche sur son support propre ligne distincte. Ensuite, comparez ce que vous avez fait avec la figure 13-11.

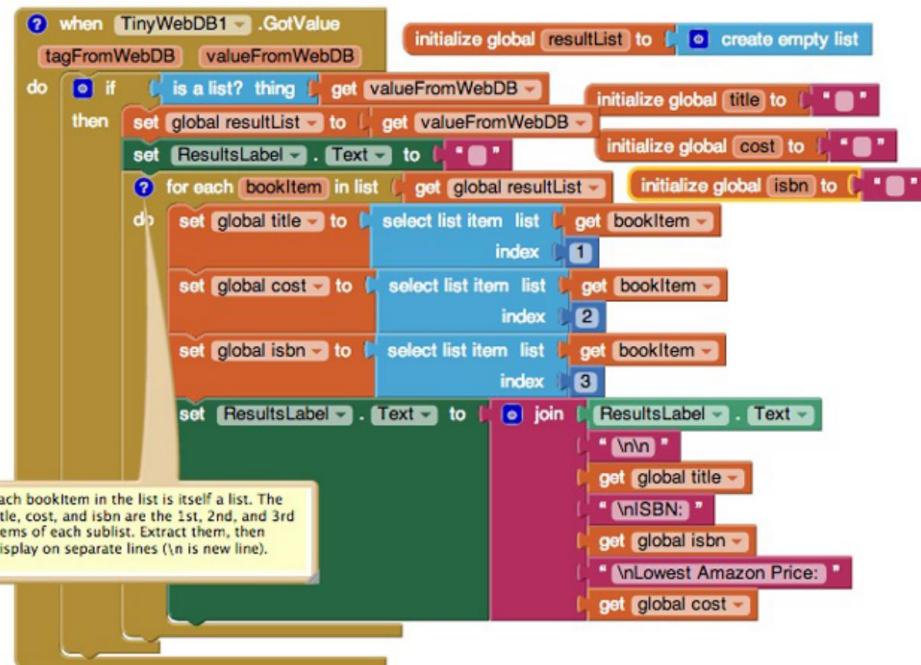


Figure 13-12. Extraire le titre, le coût et l'ISBN de chaque livre, puis les afficher sur des lignes séparées

Comment fonctionnent les blocs

Quatre variables (`resultList`, `title`, `cost` et `isbn`) sont définies pour contenir chaque élément de données tel qu'il est renvoyé par l'API. Le résultat de l'API, `valueFromWebDB`, est placé dans la variable `resultList`. Cette application aurait pu traiter l'argument `valueFromWebDB` directement, mais en général, vous le placerez dans une variable au cas où vous souhaiteriez traiter les données en dehors du gestionnaire d'événements. (Les arguments d'événement comme `valueFromWebDB` conservent leur valeur uniquement dans le gestionnaire d'événements.)

Une boucle `for each` est utilisée pour parcourir chaque élément du résultat. Rappelons que les données renvoyées par Amazon sont une liste de listes, chaque sous-liste représentant les informations d'un livre. Ainsi, l'espace réservé du `for each` est renommé `bookitem` et contient les informations actuelles du livre (une liste) à chaque itération.

Nous devons maintenant gérer le fait que la variable `bookitem` est une liste : le premier élément est le titre ; le deuxième, le prix ; et le troisième, l'ISBN. Ainsi, nous utilisons des blocs d'éléments de liste de sélection pour extraire ces éléments et les placer dans leurs variables respectives (titre, prix et isbn).

Une fois les données organisées en variables, vous pouvez les traiter comme vous le souhaitez. Cette application utilise simplement les variables dans le cadre d'un bloc de jointure qui affiche le titre, le prix et l'ISBN sur des lignes distinctes.



Testez votre application Essayez une autre recherche et vérifiez comment les informations du livre sont affichées. Cela devrait ressembler à la figure 13-12.

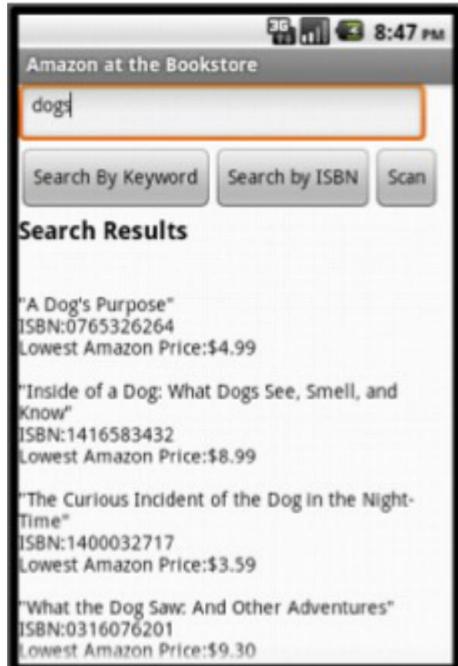
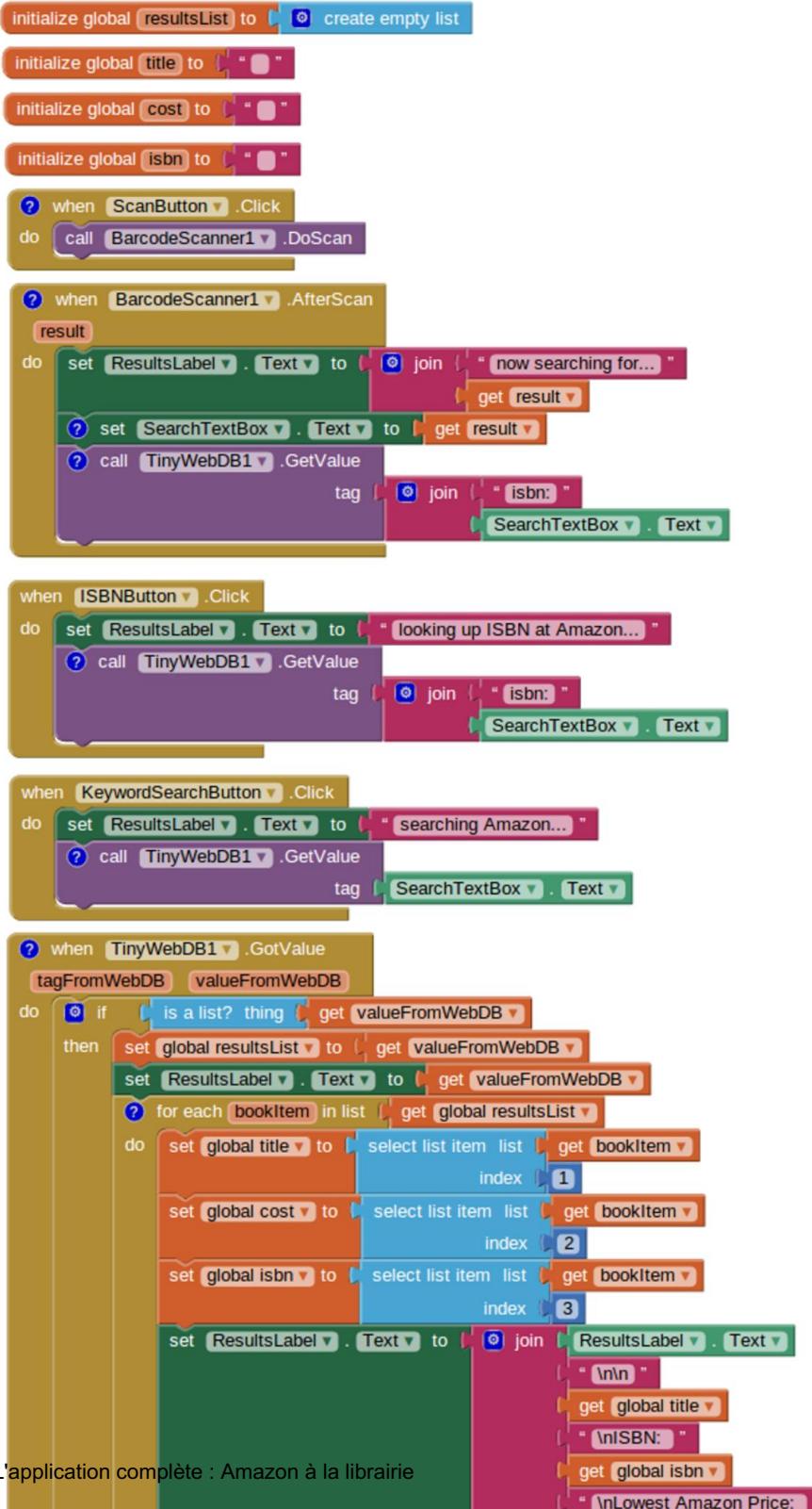


Figure 13-13. La liste de recherche affichée de manière plus sophistiquée

L'application complète : Amazon à la librairie

La figure 13-13 montre la configuration finale du bloc pour Amazon dans la librairie.



Personnalisation de l'API

L'API à laquelle vous vous êtes connecté, <http://aiamazonapi.appspot.com>, a été créée avec le langage de programmation Python et App Engine de Google. App Engine vous permet de créer et de déployer des sites Web et des services (API) hébergés sur les serveurs de Google. Vous ne payez pour App Engine que si votre site ou votre API devient populaire et attire de nombreux visiteurs.

Le service API utilisé dans ce didacticiel ne fournit qu'un accès partiel à l'API Amazon complète et renvoie un maximum de cinq livres pour toute recherche. Si vous souhaitez offrir plus de flexibilité (par exemple, lui permettre de rechercher des éléments autres que des livres), vous pouvez télécharger le code source du service depuis <http://appinventorapi.com/amazon/> et le personnaliser. Une telle personnalisation nécessite des connaissances en programmation Python, alors attention ! Mais si vous avez terminé les applications App Inventor décrites dans ce livre, vous êtes peut-être prêt à relever le défi. Pour commencer à apprendre Python, consultez la version interactive du livre *Comment penser comme un informaticien : apprendre avec Python*, puis consultez la section sur la création d'API App Inventor au chapitre 24 de ce livre.

Variantes

Une fois l'application opérationnelle, vous souhaiterez peut-être explorer certaines des variantes suivantes :

- En l'état, l'application se bloque si la recherche ne renvoie aucun livre (par exemple, lorsque l'utilisateur saisit un ISBN non valide). Modifiez les blocs pour que l'application signale lorsqu'il n'y a aucun résultat.
- Modifiez l'application pour qu'elle affiche uniquement les livres à moins de 10 \$.
- Modifiez l'application de sorte qu'après avoir numérisé un livre, son prix Amazon le plus bas soit annoncé à haute voix (utilisez le composant TextToSpeech abordé dans l'application Pas de texte pendant la conduite au chapitre 4).
- Téléchargez le <http://aiamazonapi.appspot.com> Code API et modifiez-le pour qu'il renvoie plus d'informations. Par exemple, vous pouvez lui demander de renvoyer l'URL Amazon de chaque livre, d'afficher l'URL avec chaque livre répertorié et de laisser l'utilisateur cliquer sur l'URL pour ouvrir cette page. Comme mentionné précédemment, la modification de l'API nécessite une programmation Python et une certaine connaissance de l'App Engine de Google.
Pour plus d'informations, consultez le chapitre 24.

Résumé

Voici quelques-uns des concepts que nous avons abordés avec cette application :

- Vous pouvez accéder au Web depuis une application en utilisant TinyWebDB et spécialement API construites. Vous définissez le ServiceURL du composant TinyWebDB sur l'URL de l'API, puis appelez TinyWebDB.GetValue pour demander les informations. Les données ne sont pas immédiatement renvoyées mais sont accessibles à la place dans le gestionnaire d'événements TinyWebDB.GotValue .
- La fonction BarcodeScanner.DoScan lance le scan. Lorsque l'utilisateur scanne un code-barres, l' événement BarcodeScanner.AfterScan est déclenché et les données numérisées sont placées dans le résultat de l'argument.
- Dans App Inventor, les données complexes sont représentées par des listes et des listes de listes. Si vous connaissez le format des données renvoyées par une API, vous pouvez utiliser pour chaque élément de liste et sélectionner pour extraire les informations distinctes dans des variables, puis effectuer le traitement ou configurer l'affichage comme vous le souhaitez en utilisant ces variables. .

