CAPÍTULO 8

Questionário sobre Presidentes

The Presidents Quiz é um jogo de perguntas e respostas sobre ex-líderes dos Estados Unidos. Embora este questionário seja sobre presidentes, você pode usá-lo como modelo para criar questionários ou guias de estudo sobre qualquer tópico.

Nos capítulos anteriores, você foi apresentado a alguns conceitos fundamentais de programação. Agora, você está pronto para algo mais desafiador. Você descobrirá que este capítulo requer um salto conceitual em termos de habilidades de programação e pensamento abstrato. Em particular, você usará duas variáveis de lista para armazenar os dados — neste caso, as perguntas e respostas do questionário — e usará uma variável de índice para rastrear onde o usuário está no questionário. Ao terminar, você estará armado com o conhecimento necessário para criar aplicativos de questionário e muitos outros aplicativos que requerem processamento de lista.

Este capítulo pressupõe que você esteja familiarizado com os fundamentos do App Inventor: usando o Component Designer para construir a interface do usuário e usando o Blocks Editor para especificar manipuladores de eventos e programar o comportamento do componente. Se você não estiver familiarizado com esses fundamentos, certifique-se de revisar os capítulos anteriores antes de continuar.

Você projetará o questionário para que o usuário prossiga de pergunta em pergunta clicando no botão Avançar e receba feedback sobre as respostas.

O que você aprenderá

Este aplicativo, mostrado na Figura 8-1, abrange:

- Definindo variáveis de lista para armazenar as perguntas ções e respostas em listas.
- Seqüenciar uma lista usando um índice; cada vez que o usuário clicar em Avançar, você exibirá a próxima pergunta.
- Usando comportamentos condicionais (se): realizando certas operações somente sob condições específicas Figura 8-1. The Presidents Quiz in action tions. Você usará um bloco if para lidar com o comportamento do aplicativo quando o usuário chegar ao final do questionário.

- Alternar uma imagem para mostrar uma imagem diferente para cada pergunta do questionário.

Começando

Navegue até o site do App Inventor e inicie um novo projeto. Dê ao projeto o nome "PresidentsQuiz" e defina o título da tela para "Presidents Quiz". Conecte seu dispositivo ou emulador para testes ao vivo.

Você precisará baixar as seguintes imagens para o questionário do site appinventor.org para o seu computador:

- 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

Você carregará essas imagens em seu projeto na próxima seção.

Projetando os componentes

O aplicativo Presidents Quiz possui uma interface simples para exibir as perguntas e permitir que o usuário responda. Você pode criar os componentes a partir do instantâneo do Component Designer mostrado na Figura 8-2.

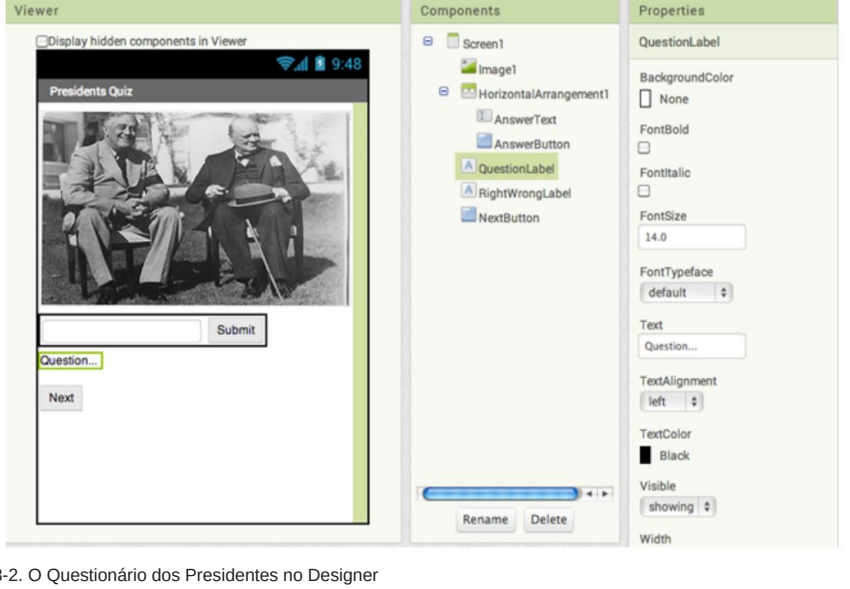

Figura 8-2. O Questionário dos Presidentes no Designer

Para criar esta interface, primeiro carregue as imagens que você baixou no projeto. Na área Mídia, clique em Adicionar e selecione um dos arquivos baixados (por exemplo, roosChurch.gif). Faça o mesmo para as outras três imagens. Em seguida, adicione os componentes listados na Tabela 8-1.

Tabela 8-1. Componentes para o aplicativo Presidents Quiz

Tipo de componente	Grupo de paletas	Qual será o nome	Propósito
Imagem	Imagem da interface do usuário	Imagem1	A imagem exibida com a pergunta.
Rótulo	Interface do usuário	QuestionLabel	Exiba a pergunta atual.
Layout de arranjo horizontal		HorizontalArrangement1	Coloque o texto da resposta e o botão em uma linha.
Caixa de texto	Interface do usuário	AnswerText	O usuário digitará sua resposta aqui.
Botão	Botão de resposta da interface do usuário		O usuário clica aqui para enviar uma resposta.
Rótulo	Interface de usuário	RightWrongLabel	Exibir "correto!" ou "incorreto!"
Botão	Interface do usuário	NextButton	O usuário clica aqui para prosseguir para a próxima pergunta.

1. Defina Image1.Picture para a imagem le roosChurch.gif, a primeira imagem que deve aparecer. Defina sua largura como "Fill parent" e sua altura como 200.
2. Defina QuestionLabel.Text como "Question..." (você inserirá a primeira pergunta no Editor de Blocos).
3. Defina AnswerText.Hint como "Digite uma resposta". Defina sua propriedade Text como em branco. Mova isso em HorizontalArrangement1.
4. Altere AnswerButton.Text para "Submit" e mova-o para HorizontalArrange Ment1.
5. Altere NextButton.Text para "Next".
6. Altere RightWrongLabel.Text para em branco.

Adicionando comportamentos aos componentes

Você precisará programar os seguintes comportamentos:

- Ao iniciar o aplicativo, aparece a primeira pergunta, incluindo sua correspondente imagem.
- Quando o usuário clica no NextButton, a segunda pergunta aparece. Ao clicar novamente, a terceira pergunta aparece e assim por diante.

- Quando o usuário chegar à última pergunta e clicar no NextButton, a primeira pergunta deverá aparecer novamente.
- Quando o usuário responder a uma pergunta, o aplicativo informará se ela está correta.

Você codificará esses comportamentos um por um, testando conforme avança.

Negando as listas de perguntas e respostas

Para começar, defina duas variáveis de lista com base nos itens da Tabela 8-2: QuestionList para conter a lista de perguntas e AnswerList para conter a lista de respostas correspondentes.
A Figura 8-3 mostra as duas listas que você criará no Blocks Editor.

Tabela 8-2. Variáveis para armazenar listas de perguntas e respostas

Tipo de bloco	Finalidade da Gaveta	
initialize global to (“QuestionList”) Variáveis	Armazena a lista de perguntas (renomeia-a QuestionList).	
inicializar global para("AnswerList")	Variáveis	Armazena a lista de respostas (renomeie-a como AnswerList).
faça uma lista	Listas	Insira os itens do QuestionList.
texto (três deles)	Texto	As questões.
faça uma lista	Listas	Insira os itens da AnswerList.
texto (três deles)	Texto	As respostas.

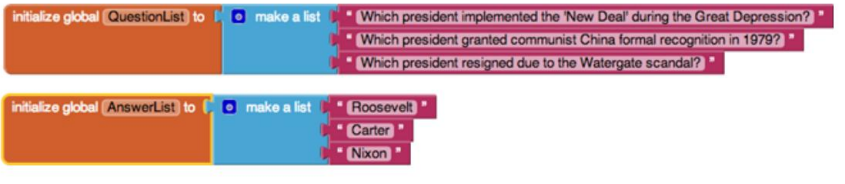

Figura 8-3. As listas para o quiz

Definindo a variável de índice

O aplicativo precisa acompanhar a pergunta atual à medida que o usuário clica no NextButton para prosseguir com o questionário. Você definirá uma variável chamada currentQuestionIndex para isso, e a variável servirá como índice tanto para QuestionList quanto para AnswerList. A Tabela 8-3 lista os blocos necessários para fazer isso, e a Figura 8-4 mostra como será essa variável.

Tabela 8-3. Criando o índice

Tipo de bloco	Finalidade da Gaveta
inicializar global para (“currentQuestionIndex”)	Variáveis Mantém o índice (posição) da pergunta/resposta atual.
número (1)	Matemática Defina o valor inicial de currentQuestionIndex como 1 (a primeira pergunta).

Figura 8-4. Iniciando a variável de índice com um valor de 1

Exibindo a primeira pergunta

Agora que você definiu as variáveis necessárias, pode especificar o comportamento interativo do aplicativo. Como em qualquer aplicativo, é importante trabalhar de forma incremental e definir um comportamento por vez. Para começar, pense apenas nas perguntas e, especificamente, em exibir a primeira pergunta da lista quando o aplicativo iniciar. Voltaremos e lidaremos com as imagens e respostas um pouco mais tarde.

Você deseja que seus blocos de código funcionem independentemente das perguntas específicas que estão na lista. Dessa forma, se você decidir alterar as perguntas ou criar um novo questionário copiando e modificando este aplicativo, você só precisará alterar as perguntas reais nas definições da lista e não precisará alterar nenhum manipulador de eventos.

Portanto, para esse primeiro comportamento, você não quer se referir diretamente à primeira pergunta: “Qual presidente implementou o 'New Deal' durante a Grande Depressão?” Em vez disso, você deseja referir-se, abstratamente, ao primeiro soquete na QuestionList (independentemente da pergunta específica ali). Dessa forma, os blocos ainda funcionarão mesmo se você modificar a pergunta naquele primeiro soquete.

Você seleciona itens específicos em uma lista com o bloco de itens da lista de seleção . O bloco pede que você especifique a lista e um índice (uma posição na lista). Se uma lista tiver três itens, você pode inserir 1, 2 ou 3 como índice.

Para esse primeiro comportamento, quando o aplicativo for iniciado, você deseja selecionar o primeiro item em QuestionList e colocá-lo em QuestionLabel. Lembre-se do Android, Where's My Car? app, se quiser que algo aconteça quando o aplicativo for iniciado, programe esse comportamento no manipulador de eventos Screen1.Initialize . Você pode usar os blocos listados na Tabela 8-4.

Tabela 8-4. Blocos para carregar a pergunta inicial ao iniciar o app

Tipo de bloco	Gaveta	Propósito
Tela1.Inicializar	Tela1	Manipulador de eventos acionado quando o aplicativo é iniciado.
defina QuestionLabel.Text para		Coloque a primeira pergunta em QuestionLabel.
selecionar item da lista	Listas	Selecione a primeira pergunta da QuestionList.
obter QuestionList global	Variáveis	A lista para selecionar perguntas.
número (1)	Matemática	Selecione a primeira pergunta usando um índice de 1.

Como funcionam os blocos

O evento Screen1.Initialize é acionado quando o aplicativo é iniciado. Conforme mostrado na Figura 8-5, o primeiro item da variável QuestionList é selecionado e colocado em QuestionLabel.Text. Assim, quando o aplicativo for iniciado, o usuário verá a primeira pergunta.

Figura 8-5. Selecionando a primeira pergunta

Teste seu aplicativo Clique em Conectar e conecte-se ao seu dispositivo ou ao emulador para testes ao vivo. Quando seu aplicativo é carregado, você vê o primeiro item da QuestionList, “Qual presidente implementou o 'New Deal' durante a Grande Depressão?”

Iterando através das perguntas

Agora, programe o comportamento do NextButton. Você já definiu o QuestionIndex atual para lembrar a pergunta atual. Quando o usuário clica no botão NextBut, o aplicativo precisa incrementar o currentQuestionIndex (ou seja, alterá-lo de 1 para 2 ou de 2 para 3 e assim por diante). Em seguida, você usará o valor resultante de currentQuestionIndex para selecionar a nova pergunta a ser exibida. Como desafio, veja se você consegue construir esses blocos sozinho. Quando terminar, compare seus resultados com a Figura 8-6.

Figura 8-6. Passando para a próxima pergunta

Como funcionam os blocos

A primeira linha de blocos incrementa a variável currentQuestionIndex. Se currentQuestionIndex tiver 1, ele será alterado para 2. Se tiver 2, será alterado para 3 e assim por diante.
Depois que a variável currentQuestionIndex foi alterada, o aplicativo a usa para selecionar a nova pergunta a ser exibida. Quando o usuário clicar em NextButton pela primeira vez, os blocos de incremento mudarão currentQuestionIndex de 1 para 2, então o aplicativo selecionará o segundo item da QuestionList, “Qual presidente concedeu reconhecimento formal à China comunista em 1979?” Na segunda vez que NextButton for clicado, o QuestionIndex atual será definido de 2 para 3, e o aplicativo selecionará a terceira pergunta da lista, “Qual presidente renunciou devido ao escândalo Watergate?”

Observação Reserve um minuto para comparar os blocos de *NextButton.Click* com os do manipulador de eventos Screen.Initialize . Nos blocos Screen.Initialize , o aplicativo usou selecionar o item da lista com um número concreto (1) para selecionar o item da lista. Nesses blocos, você está selecionando o item da lista usando uma variável como índice. O aplicativo não escolhe o primeiro item da lista, nem o segundo, nem o terceiro; ele escolhe o item currentQuestionIndexth e, portanto, um item diferente será selecionado cada vez que o usuário clicar em NextButton. Esse é um uso muito comum para um índice — incrementar seu valor para localizar e exibir ou processar itens em uma lista.

Teste seu aplicativo Teste o comportamento do NextButton para ver se o aplicativo está funcionando corretamente. Clique no botão Avançar no telefone. O telefone exibe a segunda pergunta: “Qual presidente concedeu o reconhecimento formal à China comunista em 1979?” Deveria, e a terceira pergunta deveria aparecer quando você clicar no NextButton novamente. Mas se você clicar novamente, verá um erro: “Tentativa de obter o item 4 de uma lista de comprimento 3.” O aplicativo tem um bug! Você sabe qual é o problema? Tente descobrir antes de prosseguir.

O problema com o código até agora é que ele simplesmente incrementa para a próxima pergunta a cada vez, sem nenhuma preocupação com o final do questionário. Quando currentQuestionIndex já é 3 e o usuário clica no botão Next, o aplicativo altera currentQuestionIn dex de 3 para 4. Em seguida, ele chama select list item para obter o item currentQuestionIndexth , neste caso, o quarto item. Como há apenas três itens na variável QuestionList, o dispositivo Android não sabe o que fazer. Como resultado, ele exibe um erro e força o encerramento do aplicativo. Como podemos avisar o aplicativo que chegou ao final do questionário?

O aplicativo precisa fazer uma pergunta quando o NextButton for clicado e executar diferentes blocos dependendo da resposta. Como você sabe que seu aplicativo contém três perguntas, uma maneira de fazer a pergunta seria: "A variável currentQuestionIn dex é maior que 3?" Se a resposta for sim, você deve definir currentQuestionIndex de volta para 1 e levar o usuário de volta para a primeira pergunta. Os blocos necessários para isso estão listados na Tabela 8-5.

Tabela 8-5. Blocos para verificar o valor do índice para o final da lista

Tipo de bloco	Gaveta	Propósito
se	Ao controle	Descubra se o usuário está na última pergunta.
>=	Matemática	Teste se currentQuestionIndex é maior que 3.
obter corrente global QuestãoÍndice	Arraste do bloco de inicialização	Coloque isso no lado esquerdo de =.
número 3	Matemática	Coloque isso no lado direito de = porque 3 é o número de itens na lista.
definir corrente global QuestãoÍndice para	Arraste do bloco de inicialização	Defina como 1 para reverter para a primeira pergunta.
número 1	Matemática	Defina o índice como 1.

Os blocos devem aparecer conforme ilustrado na Figura 8-7.

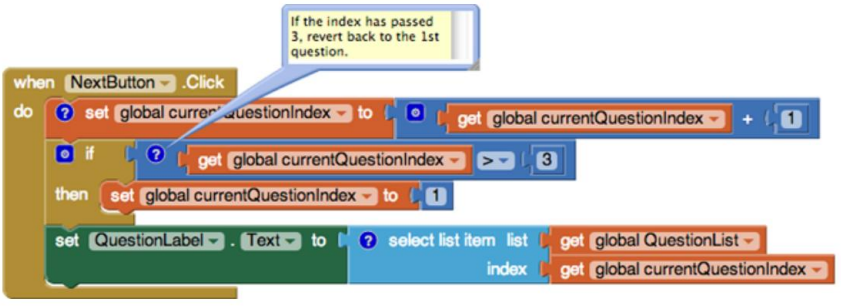

Figura 8-7. Verificando se o índice passou da última pergunta

Como funcionam os blocos

Quando o usuário clica no NextButton, o aplicativo incrementa o índice como fazia antes. Mas então, conforme mostrado na Figura 8-8, ele verifica se currentQuestionIndex é maior que 3, que é o número de perguntas. Se for maior que 3, currentQuestionIndex é definido novamente como 1 e a primeira pergunta é exibida. Se for 3 ou menos, os blocos dentro do bloco if não são executados e a pergunta atual é exibida normalmente.

Teste seu aplicativo Pegue o telefone e clique no botão Avançar. A segunda pergunta, “Qual presidente concedeu o reconhecimento formal à China comunista em 1979?” deve aparecer no QuestionLabel no telefone, como antes. Ao clicar novamente, a terceira pergunta deve aparecer no telefone. Agora, para o comportamento que você está realmente testando: se você clicar novamente, verá a primeira pergunta (“Qual presidente implementou o 'New Deal' durante a Grande Depressão?”) aparecer no telefone.

Tornando o questionário fácil de modificar

Se seus blocos para o NextButton funcionarem, dê um tapinha nas costas; você está no caminho certo para se tornar um programador! Mas e se você adicionar uma nova pergunta (e resposta) ao questionário? Seus blocos ainda funcionariam? Para explorar isso, primeiro adicione uma quarta pergunta à QuestionList e uma quarta resposta à AnswerList, conforme mostrado na Figura 8-8.

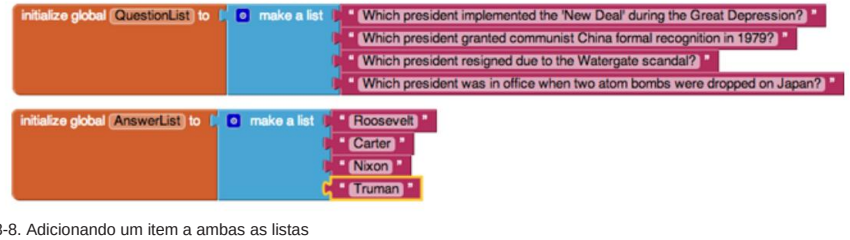

Figura 8-8. Adicionando um item a ambas as listas

Teste seu aplicativo Clique no botão Avançar várias vezes. Você notará que a quarta pergunta nunca aparece, não importa quantas vezes você clique em Avançar. Você sabe qual é o problema? Antes de continuar a leitura, veja se consegue corrigir os blocos para que a quarta pergunta apareça.

A questão aqui é que o teste para determinar se o usuário está na última questão é muito específico; ele pergunta se a variável currentQuestionIndex é maior que 3. Você poderia apenas alterar o número 3 para 4 e o aplicativo funcionaria corretamente novamente. O problema com essa solução, no entanto, é que cada vez que você modificar as perguntas e listas de respostas, você também deve se lembrar de fazer essa alteração no teste if.

Essas dependências em um programa de computador geralmente levam a bugs, especialmente quando um aplicativo cresce em complexidade. Uma estratégia muito melhor é projetar os blocos para que funcionem, não importa quantas perguntas existam. Essa generalidade torna mais fácil se você, como programador, quiser personalizar seu questionário para algum outro tópico. Também é essencial se a lista com a qual você está trabalhando muda dinamicamente — por exemplo, pense em um aplicativo de questionário que permita ao usuário adicionar novas perguntas (você criará isso no Capítulo 10). Para um programa ser mais geral, ele não pode se referir a números concretos como 3, porque isso só funciona para questionários de três questões.

Então, ao invés de perguntar se o valor de currentQuestionIndex é maior que o número específico 3, pergunte se ele é maior que o número de itens em QuestionList. Se o aplicativo fizer essa pergunta mais geral, ele funcionará mesmo quando você adicionar ou remover itens da QuestionList. Vamos modificar o manipulador de eventos NextButton.Click para substituir o teste anterior que se refere diretamente a 3. Você precisará dos blocos listados na Tabela 8-6.

Tabela 8-6. Blocos para verificar o comprimento da lista

Tipo de bloco	Gaveta	Propósito
comprimento da lista	Listas	Pergunte quantos itens estão na QuestionList.
get global QuestionList Arraste do bloco de inicialização	Coloque isso	no soquete “lista” do comprimento da lista.

Como funcionam os blocos

O teste if agora compara o currentQuestionIndex com o tamanho da Question List, conforme mostrado na Figura 8-11. Portanto, se currentQuestionIndex for 5 e o comprimento de QuestionList for 4, então currentQuestionIndex será definido de volta para 1. Observe que, como os blocos não se referem mais a 3 ou a qualquer número específico, o comportamento funcionará, não importa quantos os itens estão na lista.

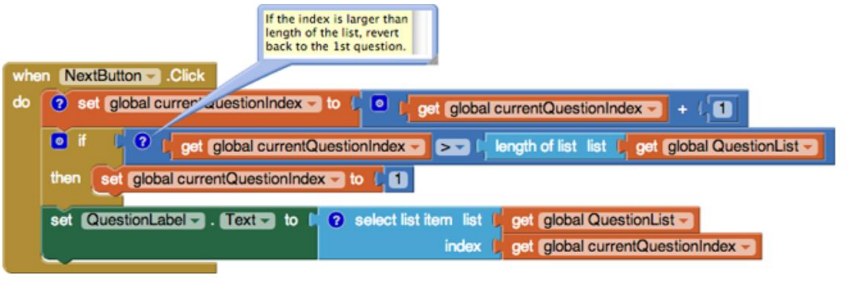

Figura 8-9. Verificando se o índice é maior que o comprimento da lista (em vez de 3)

Teste seu aplicativo Quando você clica no botão Avançar, o aplicativo passa pelas quatro perguntas, passando para a primeira após a quarta?

Mudando a imagem para cada pergunta Agora

que você programou todos os comportamentos para percorrer as perguntas (e tornou seu código mais inteligente e flexível tornando-o mais abstrato), sua próxima tarefa é fazer com que as imagens funcionem corretamente, também. No momento, o aplicativo mostra a mesma imagem, independentemente da pergunta que está sendo feita. Você pode alterar isso para que uma imagem pertencente a cada pergunta apareça quando o usuário clicar no NextButton. Anteriormente, você adicionou quatro fotos como mídia para o projeto. Agora, você criará uma terceira lista, PictureList, com os nomes dos arquivos das imagens como seus itens. Você também modificará o manipulador de eventos NextButton.Click para alternar a imagem a cada vez, assim como você alterna o texto da pergunta. (Se você já está pensando em usar o currentQuestionIndex aqui, você está no caminho certo!) Primeiro, crie uma PictureList e inicialize-a com os nomes dos arquivos de imagem. Certifique-se de que os nomes sejam exatamente iguais aos nomes de arquivo que você carregou na seção Mídia do projeto. A Figura 8-10 mostra como devem ser os blocos para PictureList .

Figura 8-10. A PictureList com nomes de arquivos de imagem como itens

Em seguida, modifique o manipulador de eventos NextButton.Click para que ele altere a imagem que aparece para cada pergunta. A propriedade Image.Picture é usada para alterar a imagem exibida. Para modificar NextButton.Click, você precisará dos blocos listados na Tabela 8-7.

Tabela 8-7. Blocos para adicionar a imagem que acompanha a pergunta

Tipo de bloco	Gaveta	Propósito
definir Image1.Imagem para	Image1	Defina isto para alterar a imagem.
selecionar item da lista	Listas	Selecione a imagem correspondente à pergunta atual.
obter PictureList global	Arraste para fora do bloco de inicialização	Selecione um nome de arquivo nesta lista.
obter índice global de questões atuais	Arraste para fora do bloco de inicialização	Selecione o item currentQuestionIndex .

Como funcionam os blocos

O currentQuestionIndex serve como índice tanto para QuestionList quanto para PictureList. Contanto que você tenha configurado suas listas adequadamente de modo que a primeira pergunta corresponda à primeira imagem, a segunda à segunda e assim por diante, o índice único pode servir a ambas as listas, conforme mostrado na Figura 8-11 . Por exemplo, a primeira foto, roosChurch.gif, é uma foto do presidente Franklin Delano Roosevelt (sentado com o primeiro-ministro britânico Winston Churchill), e “Roosevelt” é a resposta para a primeira pergunta.

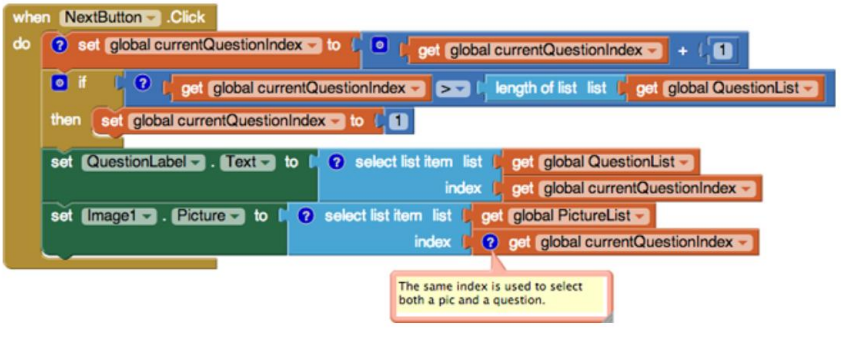

Figura 8-11. Selecionando a imagem currentQuestionIndexth a cada vez

Teste seu aplicativo Clique em Avançar algumas vezes. Uma imagem diferente aparece cada vez que você clica no NextButton?

Verificando as respostas do usuário Até

agora, você criou um aplicativo que simplesmente percorre as perguntas e respostas (emparelhado com uma imagem da resposta). É um ótimo exemplo de aplicativos que usam listas, mas para

Para ser um verdadeiro aplicativo de quiz, ele precisa fornecer feedback aos usuários sobre se suas respostas estão corretas. Vamos adicionar os blocos para fazer exatamente isso. Nossa interface é configurada para que o usuário digite uma resposta em AnswerText e clique no AnswerButton. O aplicativo deve comparar a entrada do usuário com a resposta à pergunta atual, usando um bloco if else para verificar. O RightWrongLabel deve então ser modificado para relatar se a resposta está correta. Existem alguns blocos necessários para programar esse comportamento, todos listados na Tabela 8-8.

Tabela 8-8. Blocos para indicar se uma resposta está correta

Tipo de bloco	Gaveta	Propósito
AnswerButton.Click	Botão de resposta	Acionado quando o usuário clica no AnswerButton.
caso contrário	Ao controle	Se a resposta estiver correta, faça uma coisa; caso contrário, faça outro.
=	Matemática	Pergunte se a resposta está correta.
RespostaTexto.Texto	RespostaTexto	Contém a resposta do usuário.
selecionar item da lista	Listas	Selecione a resposta atual da AnswerList.
obter AnswerList global	Arraste para fora do bloco de inicialização	A lista para selecionar.
obter índice global de questões atuais	Arraste para fora do bloco de inicialização	O número da pergunta (e resposta) atual.
definir RightWrongLa bel.Text para	Rótulo CertoErrado	Relate a resposta aqui.
texto (“correto!”)	Texto	Exiba isso se a resposta estiver certa.
definir RightWrongLa bel.Text para	Rótulo CertoErrado	Relate a resposta aqui.
texto (“incorreto!”)	Texto	Exiba isso se a resposta estiver errada.

Como funcionam os blocos

A Figura 8-12 demonstra como o teste if else pergunta se a resposta fornecida pelo usuário (AnswerText.Text) é igual ao item currentQuestionIndexth na lista de respostas. Se currentQuestionIndex for 1, o aplicativo comparará a resposta do usuário com o primeiro item em AnswerList, “Roosevelt”. Se currentQuestionIndex for 2, o aplicativo comparará a resposta do usuário com a segunda resposta da lista, “Carter”, e assim por diante. Se o resultado do teste for positivo, então a ramificação é executada e o RightWrongLabel é definido como “correto!” Se o teste for falso, a ramificação else é executada e o RightWrongLabel é definido como “incorreto!”

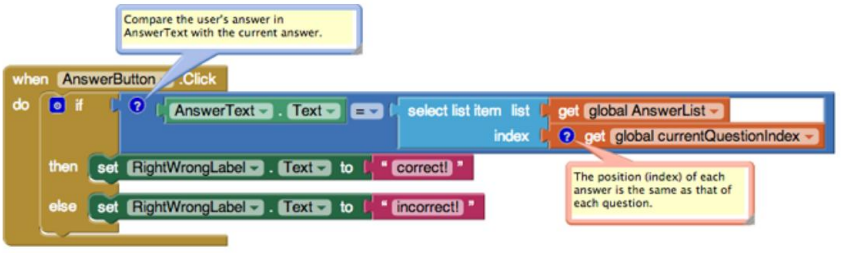

Figura 8-12. Verificando a resposta

Teste seu aplicativo Tente responder a uma das perguntas. Ele deve relatar se você respondeu à pergunta exatamente como especificado na lista de respostas. Teste com uma resposta correta e incorreta. Você provavelmente notará que, para uma resposta ser marcada como correta, ela deve ser uma correspondência exata e específica para o que você inseriu na AnswerList. Certifique-se também de testar se as coisas funcionam em perguntas sucessivas.

O aplicativo deve funcionar, mas você pode perceber que, ao clicar no NextButton, a mensagem "correto!" ou "incorreto!" o texto e a resposta anterior ainda estão lá, conforme mostrado na Figura 8-13, mesmo que você esteja olhando para a próxima pergunta. É bastante inócuo, mas os usuários do seu aplicativo definitivamente perceberão esses problemas de interface do usuário. Para apagar Right WrongLabel e AnswerText, você colocará os blocos listados na Tabela 8-9 no manipulador de eventos NextButton.Click .

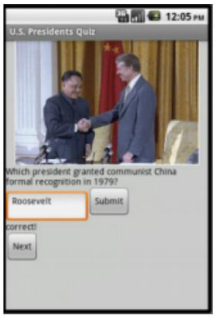

Figura 8-13. O questionário em ação com a resposta anterior aparecendo quando não deveria

Tabela 8-9. Blocos para limpar o RightWrongLabel

Tipo de bloco	Gaveta	Propósito
definir RightWrongLa bel.Text para	RightWrongLabel Este é o rótulo a ser apagado.	
texto ("")	Texto	Quando o usuário clicar em NextButton, limpe o feedback da resposta anterior.
definir AnswerText.Text como AnswerText		A resposta do usuário da pergunta anterior.
texto ("")	Texto	Quando o usuário clicar no NextButton, limpe a resposta anterior.

Como funcionam os blocos

Conforme mostrado na Figura 8.14, ao clicar no NextButton, o usuário passa para a próxima pergunta, de modo que as duas primeiras linhas do manipulador de eventos limpem RightWrongLabel e AnswerText .

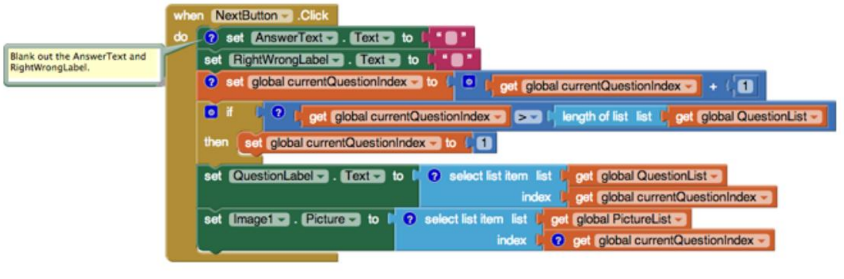

Figura 8-14. A PictureList com nomes de arquivos de imagem como itens

Teste seu aplicativo Responda a uma pergunta e clique em Enviar e, em seguida, clique no botão Avançar. Sua resposta anterior e seu feedback desapareceram?

O aplicativo completo: o questionário dos presidentes

A Figura 8-15 mostra a configuração do bloco final para o Presidents Quiz.

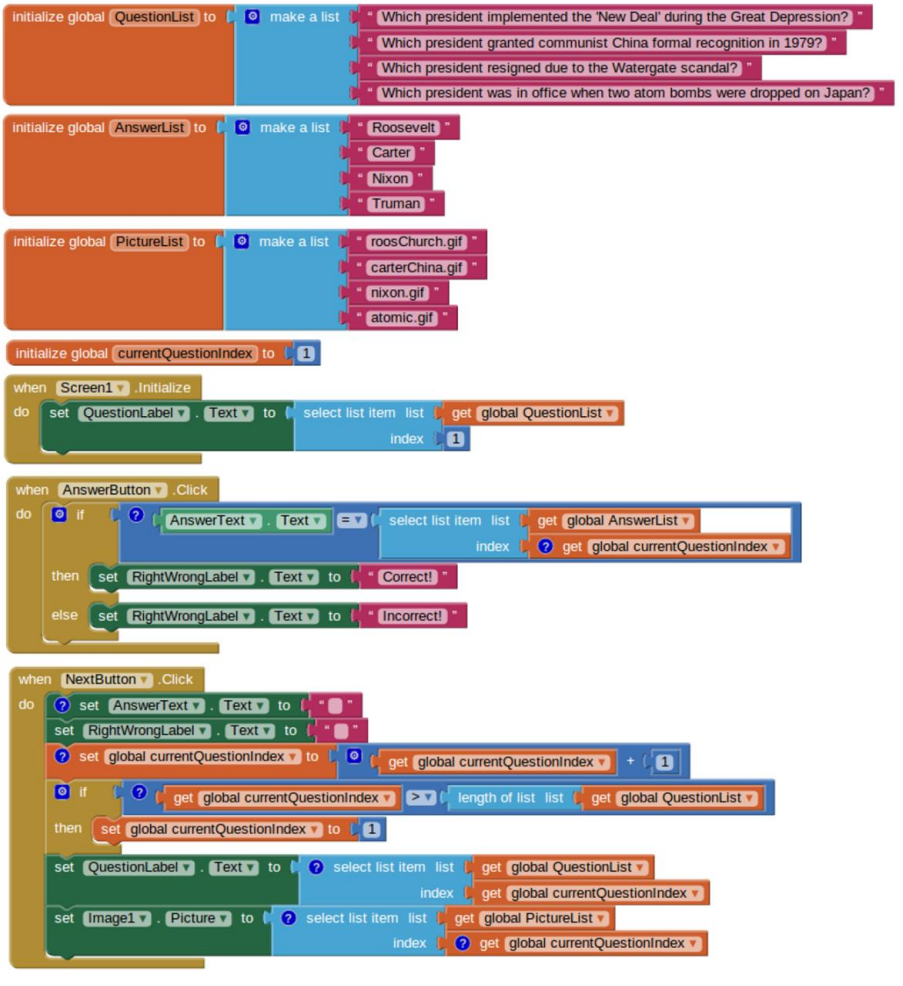

Figura 8-15. Os blocos para o Questionário dos Presidentes

variações

Ao fazer este teste funcionar, você pode querer explorar algumas variações, como as seguintes:

- Em vez de apenas mostrar imagens para cada pergunta, tente reproduzir um clipe de som ou um pequeno vídeo. Com som, você pode transformar seu questionário em um aplicativo Name That Tune.
- O questionário é muito rígido em termos do que aceita como resposta válida. Existem várias maneiras de melhorar isso, aproveitando os blocos de processamento de texto na gaveta de texto. Uma maneira é usar o bloco de letras maiúsculas na gaveta Texto para converter a resposta do usuário e a resposta real em letras maiúsculas antes de comparar. Outro

é usar o bloco text.contains para ver se a resposta do usuário está contida na resposta real. Outra opção é fornecer várias respostas para cada pergunta e verificar iterando (foreach) através delas para ver se alguma corresponde.

- Outra forma de melhorar a verificação das respostas é transformar o quiz para que seja de múltipla escolha. Você precisará de uma lista adicional para conter as opções de resposta para cada pergunta. As respostas possíveis serão uma lista de listas, com cada sublista contendo as opções de resposta para uma pergunta específica. Use o componente ListPicker para dar ao usuário a capacidade de escolher uma resposta. Você pode ler mais sobre listas no Capítulo 19.

Resumo

Aqui estão algumas das ideias que abordamos neste tutorial:

- Os aplicativos mais sofisticados têm dados (geralmente armazenados em listas) e comportamento - seus manipuladores de eventos.
- Use um bloco ifelse para verificar as condições. Para mais informações sobre condicionais, consulte o Capítulo 18.
- Os blocos nos manipuladores de eventos devem se referir apenas abstratamente aos itens da lista e ao tamanho da lista para que o aplicativo funcione mesmo se os dados na lista forem alterados.
- Variáveis de índice rastreiam a posição atual de um item dentro de uma lista. Ao incrementá-los, use um bloco if para lidar com o comportamento do aplicativo quando o usuário chegar ao final da lista.