Tarefa 1: Memorizar

Objetivo

A meta dessa tarefa é recriar a demonstração dada nas primeiras duas palestras e fazer algumas pequenas melhorias. É importante que você entenda o que você está fazendo com cada passo da recriação da demonstração da palestra, então você está preparado para fazer tais melhorias.

Trata-se principalmente de experimentar a criação de um projeto no *Xcode* e digitar o código a partir do zero. **Não copie / cole nenhum código de nenhum lugar.** Digite e veja o que o *Xcode* faz.

Esteja certo de revisar a seção de Dicas a seguir.

Também, confira as últimas novidades na seção de Avaliação para ter certeza de que você entendeu o que você está fazendo para ser avaliado nesta tarefa.

Prazo de Entrega

Essa tarefa deve ser entregue antes de você começar a assistir a Palestra 3.

Materiais

Para recriar a demonstração, você certamente precisará assistir as duas primeiras palestras.

Você necessitará instalar o programa *Xcode 11.4* (gratuito), usando a *App Store* no seu *Mac* (versões anteriores do *Xcode* não funcionarão, embora você possa tentar a versão 11.3.1 e ver se funciona). É altamente recomendável que você faça isso imediatamente, caso você tenha qualquer problema em adquirir o *Xcode* para usar, você tem tempo para pedir ajuda no *Piazza*. O *Xcode* 11.4 requer o *Catalina* (*macOS* 10.15.4).

Se você está utilizando uma versão mais recente do *Xcode*, talvez algumas melhorias na *SwiftUI* façam com que alguma das palestras não faça mais sentido. Você só terá que adaptá-las.

Tarefas necessárias

1. Faça o jogo Memorizar funcionar como demonstrado nas Palestras 1 e 2. Digite todo o código. **Não copie / cole o código de nenhum lugar.**

2. Atualmente, os cartões aparecem em ordem previsível (as partidas são sempre lado-a-lado, fazendo o jogo ser muito fácil). Após isso, embaralhe os cartões.

3. Nossos cartões são ordenados atualmente numa única coluna (iremos consertar isso na próxima semana). Isso faz com que os nossos cartões serem realmente pequenos e finos (especialmente em retrato), que não parece ser muito bom. Force para cada cartão ter uma relação de largura/altura para 2:3 (isso resultará em um espaço vazio acima e / ou abaixo dos seus cartões, dos quais são finos).

4. O seu jogo tenha uma inicialização com um número aleatório de pares de cartões entre 2 pares e 5 pares.

5. Quando o seu jogo aleatoriamente mostrar 5 pares, a fonte que estamos utilizando para o emoji será muito larga (em retrato) e começará a ser cortada. Teremos que ajustar a fonte nos 5 pares (apenas) para usar uma fonte menor que ***.largeTitle***. Continue a usar ***.largeTitle*** quando estiverem 4 ou menos pares no jogo.

6. Sua UI deverá funcionar em modo de retrato ou paisagem em qualquer dispositivo *IOS*. No modo paisagem, sues cartões devem ser largos (mas ainda com a relação 2:3. Isso provavelmente não requer nenhum trabalho na sua parte (isso é parte do poder do *SwiftUI*), mas certifique-se de experimentar funcionando em diferentes simulações no Xcode para ter certeza.

Dicas

1. Economia é valiosa na codificação. A maneira mias fácil de garantir uma linha de código sem erros é não escrever essa linha de código de forma alguma. Os tópicos 2, 3 e 4 (e possivelmente também o 5) das Tarefas Necessárias, podem ser feitos com a adição ou alteração de **uma única linha de código** (isso é uma dica, não uma tarefa necessária, então você não precisa fazê-la necessariamente).

2. Não implementamos a parte “reativa"do *SwiftUI* ainda(por exemplo, quando o View automaticamente atualiza toda vez que o Model mudar), então você terá que reiniciar a sua aplicação no simulador (e observando no painel de Pré-visualização) enquanto testa o seu código.

3. Embaralhar os cartões pode ser mais fácil do que você pensa. Não deixe de se familiarizar com a documentação para usar o *Array*.

4. Certifique-se de fazer o embaralhamento no local apropriado na sua *MVVM*. O embaralhamento é pertencente ao UI (por exemplo, *View* ou *ViewModel*) ou ao *Model*?

5. Outra área da documentação que você vai querer de familiarizar muito é a documentação para o protocolo *View*. Há muita coisa lá (*View* é muito poderoso), e é claro que você não é obrigado a entender todas as funções do *View* (ainda), mas pelo menos escanear / pesquisar através dele lhe dará uma ideia do que há lá, e provavelmente será útil para resolver o seu problema de resolução.

6. A função ***Int.random(in: ClosedRange<Int>)*** pode gerar um número inteiro em qualquer faixa, por exemplo, ***let random = Int.random(in: 15…75)*** geraria um número inteiro entre 15 e 75, inclusive.

7. Com certeza, você precisará ver a documentação para a *Font*. Você está convidado a usar uma das outras fontes embutidas listadas ali (ou seja, você não precisa chamar nenhuma das fontes como sistema ou personalizado). Basta escolher a fonte que lhe der o tamanho desejado, já que uma fonte não afeta o aspecto de um emoji (além do tamanho).

8. O tópico 5 das Tarefas Necessárias também pode ser implementado em uma única linha de código. Mas você provavelmente precisará usar o operador “ternário"do *Swift* (por exemplo, ***let x = truthTest ? Foo : bar***) para fazer isso. É muito comum o uso de operadores ternários para afetar os argumentos de modificadores de *Views*.

9. Uma coisa que não estamos abordando nesta tarefa é que nossa fonte emoji parece estar muito pequena em modo paisagem. Nós vamos consertar isso de uma maneira muito mais poderosa na próxima semana.

Coisas para aprender

Aqui está uma lista parcial de conceitos que esta tarefa se destina a permitir que você adquira prática com ou de outra forma demonstre seu conhecimento.

1. *Xcode* 11.4
2. *Swift* 5.2
3. *MVVM*
4. Procurar conceitos na documentação (*Array, Font* and *View*)
5. *Int.random(in:)*
6. Executar a sua aplicação em diferentes simuladores.

Avaliação

Em todas as tarefas deste trimestre, o objetivo é escrever um código de qualidade que seja executado sem alertas ou erros, e depois testar a aplicação resultante e iterar até que funcione corretamente.

Aqui estão as razões mais comuns pelas quais as tarefas são avaliadas com nota baixa:

- Projeto dos não construídos;
- Um ou mais itens na seção Tarefas Necessárias não foram satisfeitos;
- Um conceito fundamental não foi compreendido;
- O projeto não é executado sem alertas;
- -O código é visualmente descuidado e difícil de ler (por exemplo, o recuo não é consistente, etc.);
- Sua solução é difícil (ou impossível) para alguém que lê o código entender devido à falta de comentários, nomes de variáveis/métodos pobres, estruturas de solução ruins, métodos longos, etc;

Muitas vezes, os alunos perguntam: “Quantos comentários devo do meu código eu devo fazer?” **A resposta é que seu código deve ser fácil e completamente compreensível por qualquer pessoa que o leia.** Você pode assumir que o leitor conhece a API do IOS e sabe como funciona o código do jogo Memorize das palestras 1 e 2, mas não deve assumir que eles já conhecem sua (ou qualquer) solução para a tarefa.

Crédito Extra

Tentamos fazer com que o Crédito Extra seja uma oportunidade para expandir o que você aprendeu esta semana. Tentar, ao menos, alguns deles a cada semana é altamente recomendado para aproveitar ao máximo este curso. Quanto você ganha no Crédito Extra depende do escopo do item em questão.

Se você optar por lidar com um item de Crédito Extra, marque-o em seu código com comentários para que seu graduador possa encontrá-lo.

Esta semana há apenas um item de Crédito Extra, desculpe! Se pensarmos em mais alguns esta semana, os colocaremos nos fóruns de classe.

1. Faça com que os emoji em seus cartões sejam escolhidos aleatoriamente de um conjunto maior de emoji (pelo menos uma dúzia). Em outros, não use sempre os mesmos cinco emoji em todos os jogos.

Traduzido por Henrique Matheus Alves Pereira - henrique.map@outlook.com

https://cs193p.sites.stanford.edu/sites/g/files/sbiybj16636/files/media/file/a1.pdf