

# Use a Cabeça C#

Quarta Edição

NÃO SERIA UM SONHO SE  
EXISTISSE UM LIVRO C# MAIS  
DIVERTIDO DO QUE MEMORIZAR UM  
DICIONÁRIO? PROVAVELMENTE É SÓ  
UMA FANTASIA...



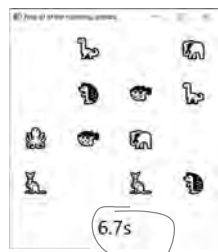
Andrew Stellman

Jennifer Greene



**ALTA BOOKS**  
GRUPO EDITORIAL  
Rio de Janeiro, 2023

## Sumário



Vamos aumentar o interesse pelo jogo! O tempo transcorrido desde o início do jogo aparecerá na parte inferior da janela, aumentando constantemente e só parando após o último animal ser combinado.

Intro	xxvii
1. Comece a criar com C#: <i>Crie algo incrível... rápido!</i>	39
2. Aprofunde-se no C#: Declarações, classes e código	87
<i>Unity Lab #1: Explore C# com o Unity</i>	125
3. Objetos... oriente-se!: <i>Entendendo o código</i>	141
4. Tipos e referências: <i>Obtendo referência</i>	193
<i>Unity Lab #2: Escreva Código C# para o Unity</i>	251
5. Encapsulamento: <i>Mantenha sua privacidade... privada</i>	265
6. Herança: <i>Árvore genealógica do objeto</i>	311
<i>Unity Lab #3: Instâncias GameObject</i>	381
7. Interfaces, coerções e "is": <i>Classes cumprindo suas promessas</i>	393
8. Enums e coleções: <i>Organizando seus dados</i>	443
<i>Unity Lab #4: Interfaces do usuário</i>	453
9. LINQ e lambdas: <i>Controle seus dados</i>	505
10. Lendo e gravando arquivos: <i>Salve o último byte para mim!</i>	567
<i>Unity Lab #5: Raycast</i>	615
11. Captain Amazing: <i>The Death Of The Object</i>	625
12. Tratamento de exceção: <i>Apagar incêndio é cansativo</i>	661
<i>Unity Lab #6: Navegação da Cena</i>	689
Exercício para Download: <i>Batalha final de combinação de animais</i>	699
i. Projetos Blazor do ASP.NET Core	
Guia do Aluno Visual Studio para Mac	701
ii. Guia do Código Kata para Avançados e/ou Impatientes	763



## Introdução

**Sua mente no C#.** Você está sentado tentando aprender alguma coisa, mas sua mente fica insistindo em lhe dizer que esta aprendizagem não é importante. Sua mente diz: "Melhor deixar espaço para coisas mais importantes, por exemplo, quais animais selvagens evitar e se praticar arco e flecha pelado é uma ideia ruim." Como você engana sua mente para ela pensar que sua vida realmente depende de aprender o C#?

A quem se destina este livro?	xxviii
Sabemos o que você pensa.	xxix
Sabemos o que seu <i>cérebro</i> pensa.	xxix
Metacognição: pensando sobre pensar	xxx
Veja o que NÓS fizemos	xxxii
LEIA-ME	xxxiv
Equipe de revisão técnica	xxxvi
Agradecimentos	xxxvii



# 1

comece a criar com c#

## Crie algo incrível... rápido!

### Deseja criar apps incríveis... agora mesmo?

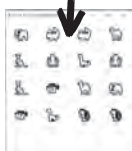
Com C#, você tem uma **linguagem de programação moderna** e uma **ferramenta valiosa** ao seu alcance. E com o **Visual Studio** você tem um **ótimo ambiente de desenvolvimento** com recursos muito intuitivos que facilitam bastante a codificação. O Visual Studio não é só uma ótima ferramenta para escrever código, também é uma **ferramenta de aprendizagem realmente valiosa** para explorar a linguagem C#. Ficou interessado? Vire a página e vamos codificar.



ELABORAR PROJETO



CRIAR JANELA



ESCREVER CÓDIGO C#



LIDAR COM CLIQUES DO MOUSE



ADICIONAR RELÓGIO

Por que você deve aprender C#	40
Visual Studio é uma ferramenta para escrever código e para explorar a linguagem C#	41
Crie seu primeiro projeto no Visual Studio	42
Crie um jogo!	44
Como criar seu jogo	45
Crie um projeto WPF no Visual Studio	46
Use XAML para criar sua janela	50
Crie a janela do jogo	51
Tamanho e título da janela com as propriedades XAML	52
Adicione linhas e colunas à grade do XAML	54
Linhas e colunas com tamanho igual	55
Adicione um controle TextBlock à grade	56
Tudo pronto para começar a escrever o código do jogo	59
Gere um método para configurar o jogo	60
Termine o método SetUpGame	62
Execute o programa	64
Adicione seu novo projeto ao controle de versão	68
A próxima etapa é lidar com os cliques do mouse	71
Faça os TextBlocks responderem aos cliques do mouse	72
Adicione o código TextBlock_MouseDown	75
Faça o resto dos TextBlocks chamar o mesmo manipulador de eventos MouseDown	76
Termine o jogo adicionando um cronômetro	77
Adicione um cronômetro ao código do jogo	78
Use o depurador para resolver a exceção	80
Adicione o resto do código e termine o jogo	84
Atualize o código no controle de versão	85
Ainda melhor se...	86

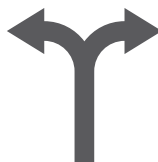
## 2

aprofunde-se no C#

**Declarações, classes e código****Você não é só um usuário de IDE. É um desenvolvedor.**

Muito trabalho é feito usando o IDE, mas só até certo ponto. O Visual Studio é uma das ferramentas de desenvolvimento de software mais avançadas já criadas, mas um **IDE poderoso** é apenas o começo. É hora de se **aprofundar no código do C#**: sua estrutura, seu funcionamento e seu controle... porque não há limites para o que seus apps podem fazer.

Vejam de perto os arquivos de um aplicativo de console	88
Duas classes podem estar no mesmo namespace (e arquivo!)	90
Declarações são os blocos de construção dos apps	93
Programas usam variáveis para trabalhar com dados	94
Gere um novo método para trabalhar com variáveis	96
Adicione ao método um código que usa operadores	97
Use o depurador para ver as variáveis mudarem	98
Use operadores para trabalhar com variáveis	100
As declarações "if" tomam decisões	101
Loops realizam uma ação repetidamente	102
Use snippets de código para escrever loops	105
Os controles orientam a mecânica das IUs	109
Crie um app WPF para experimentar os controles	110
Adicione um controle TextBox ao app	113
Adicione o código C# para atualizar o TextBlock	115
Adicione um manipulador de eventos que permita apenas a entrada de números	117
Adicione rolagem à linha inferior da grade	121
Adicione código C# para os outros controles funcionarem	122

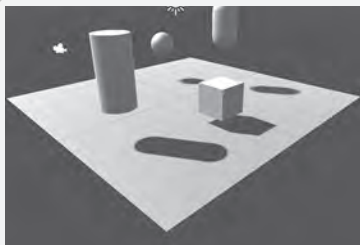


# Unity Lab 1

## Explore C# com o Unity

Bem-vindo ao seu primeiro **Use a Cabeça C# - Unity Lab**. Escrever código é uma habilidade e, como qualquer outra, melhorar requer **prática e experimentação**. O Unity será uma ferramenta muito valiosa nesse sentido. Neste lab, você começará a praticar o que aprendeu sobre o C# nos Capítulos 1 e 2.

Unity, uma ferramenta avançada para design de jogos	126
Baixe o Unity Hub	127
Use o Unity Hub para criar um novo projeto	128
Assuma controle do layout do Unity	129
Sua cena em um ambiente 3D	130
Os jogos Unity são feitos de GameObjects	131
Use Move Gizmo para mover seus GameObjects	132
Inspector mostra os componentes do GameObject	133
Adicione material ao GameObject Sphere	134
Gire a esfera	137
Seja criativo!	140



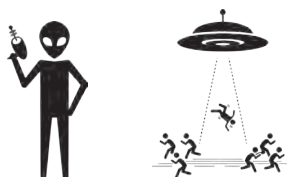
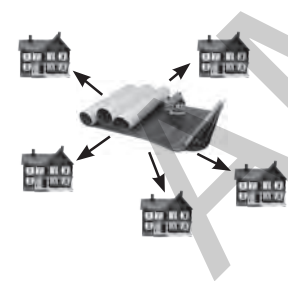
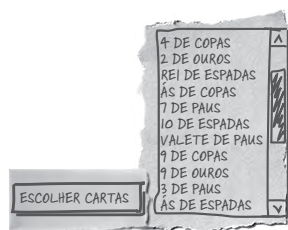
## 3

## objetos... oriente-se!

## Entendendo o código

## Cada programa que você cria resolve um problema.

Quando você cria um programa, sempre é uma boa ideia começar a pensar em qual problema ele deve resolver. Por isso os **objetos** são tão úteis. Eles permitem estruturar seu código com base no problema sendo resolvido para que você passe o seu tempo *refletindo sobre o problema* no qual precisa trabalhar, em vez de ficar parado na mecânica da escrita do código. Quando os objetos são usados corretamente, e você realmente considera como são elaborados, o resultado é um código *fácil* de escrever, ler e alterar.



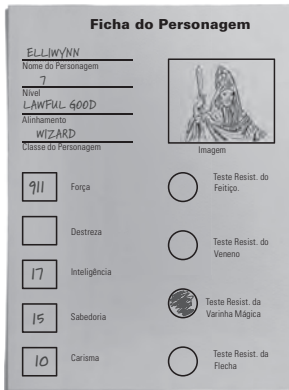
Se o código é útil, é reutilizado	142
Certos métodos requerem parâmetros e retornam valor	143
Vamos criar um programa que escolhe algumas cartas	144
Crie o aplicativo de console PickRandomCards	145
Termine o método PickSomeCards	146
A classe CardPicker terminada	148
Ana trabalha em seu próximo jogo	151
O jogo de Ana está evoluindo...	152
Crie um protótipo no papel para um jogo clássico	154
Crie uma versão WPF do app para escolher cartas	156
StackPanel é um contêiner que empilha outros controles	157
Reutilize a classe CardPicker no novo app WPF	158
Use Grid e StackPanel para o layout da janela principal	159
Layout da janela do app para desktop Card Picker	160
Os protótipos de Ana parecem ótimos...	163
Ana pode usar objetos para resolver o problema	164
Você usa uma classe para criar um objeto	165
Ao criar um novo objeto a partir de uma classe, ele é chamado de instância dessa classe	166
Uma solução melhor para Ana... trazida por objetos	167
Uma instância usa campos para controlar as coisas	171
Obrigado pela memória	174
O que passa na mente do programa	175
Às vezes o código pode ser difícil de ler	176
Use nomes claros de classe e de método	178
Crie uma classe para trabalhar com Guy	184
Há um modo mais fácil de inicializar objetos com C#	186
Janela C# Interativo para rodar o código C#	192

# 4

## tipos e referências

### Obtendo referência

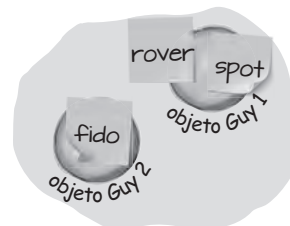
**O que seria dos seus apps sem dados?** Pense um pouco. Sem dados, seus programas são... bem, é muito difícil imaginar escrever código sem dados. Você precisa de **informações** dos usuários e as utiliza a fim de pesquisar ou produzir novas informações para retornar a eles. Na verdade, quase tudo o que você faz na programação envolve **trabalhar com dados**, de um modo ou de outro. Neste capítulo, você entenderá os prós e os contras dos **tipos de dados** e das **referências** do C#, verá como trabalhar com dados em seu programa e até aprenderá outras coisas sobre **objetos** (*adivinha... objetos são dados também!*).



Criar uma referência é como escrever um nome em uma nota adesiva e colocá-la no objeto. Você está usando-a para rotular um objeto para se referir a ele mais tarde.



Owen poderia ter nossa ajuda!	194
As fichas do personagem armazenam diferentes tipos de dados no papel	195
O tipo de uma variável determina quais dados ela pode armazenar	196
C# tem vários tipos para armazenar inteiros	197
Vamos falar sobre strings	199
Literal é um valor escrito diretamente no código	200
Variável lembra um copo de dados para viagem	203
Outros tipos com tamanhos diferentes também	204
10kg de dados em uma embalagem de 5kg	205
A coerção permite copiar valores que o C# não pode converter automaticamente em outro tipo	206
C# faz algumas conversões automaticamente	209
Quando você chama um método, os argumentos precisam ser compatíveis com os tipos dos parâmetros	210
Owen sempre melhora seu jogo...	212
Vamos ajudar Owen a experimentar as pontuações da habilidade	214
Use o compilador C# para encontrar linhas com problemas	216
Use variáveis de referência para acessar os objetos	224
Referências são como notas adesivas para os objetos	225
Se não houver mais nenhuma referência, seu objeto será descartado no lixo	226
Múltiplas referências e seus efeitos colaterais	228
Duas referências significam DUAS variáveis que podem mudar os dados do mesmo objeto	235
Objetos usam referências para se comunicar	236
Arrays mantêm muitos valores	238
Arrays podem conter variáveis de referência	239
Null significa uma referência que aponta para nada	241
Um test drive aleatório	245
Bem-vindo à lanchonete Sandubas Preço Bom é Aqui de Sloppy Joe!	246

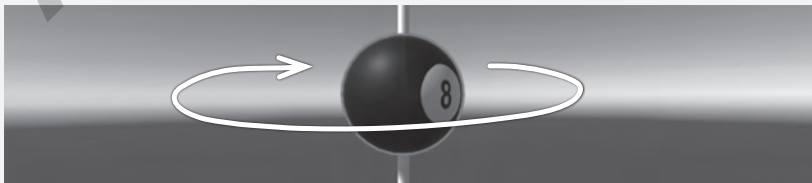


# Unity Lab 2

## Escreva Código C# para o Unity

Unity não é só um motor e editor multiplataforma poderoso para criar jogos e simulações em 2D e 3D. É também uma **ótima maneira de praticar a escrita do código C#**. Neste Lab você praticará mais a escrita do código C# para um projeto Unity.

Os scripts C# adicionam comportamento aos GameObjects	252
Adicione um script C# ao GameObject	253
Escreva código C# para girar a esfera	254
Ponto de interrupção para depurar seu jogo	256
Use o depurador para entender Time.deltaTime	257
Adicione um cilindro para mostrar o eixo Y	258
Adicione campos à classe para o ângulo de rotação e a velocidade	259
Use Debug.DrawRay para explorar como os vetores em 3D funcionam	260
Rode o jogo e veja o raio na exibição Scene	261
Gire a bola em um ponto na cena	262
Use o Unity para ver melhor a rotação e os vetores	263
Seja criativo!	264





# 5

## encapsulamento

### Mantenha sua privacidade... privada

#### Já desejou ter um pouco mais de privacidade?

Às vezes seus objetos sentem o mesmo. Assim como você não quer um estranho lendo seu diário ou folheando seu extrato bancário, os bons objetos não permitem que **outros** objetos metam o nariz em seus campos. Neste capítulo, você aprenderá sobre o poder do **encapsulamento**, um modo de programar que o ajuda a tornar o código flexível, fácil de usar e difícil de abusar. Você **tornará privados os dados dos objetos** e adicionará **propriedades** para proteger como esses dados são acessados.



SwordDamage
Roll
MagicMultiplier
FlamingDamage
Damage
CalculateDamage
SetMagic
SetFlaming

Ajudando Owen com os danos	266
Crie um aplicativo de console para calcular os danos	267
XAML para uma versão WPF da calculadora de danos	269
Code-behind para a calculadora de danos WPF	270
Conversa sobre jogos de tabuleiro (ou quem sabe... discussão sobre rolar dados?)	271
Vamos tentar corrigir o bug	272
Debug.WriteLine para escrever informações de diagnóstico	273
É fácil usar mal seus objetos sem querer	276
Encapsulamento significa manter privados os dados em uma classe	277
Use o encapsulamento para controlar o acesso aos métodos e aos campos da classe	278
Mas o campo RealName está REALMENTE protegido?	279
Campos e métodos privados só podem ser acessados em instâncias da mesma classe	280
Por que encapsulamento? O objeto como uma caixa-preta...	285
Usaremos o encapsulamento para melhorar a classe SwordDamage	289
O encapsulamento mantém seus dados seguros	290
Aplicativo de console para testar PaintballGun	291
As propriedades facilitam o encapsulamento	292
Modifique o método Main para usar a propriedade Balls	293
Propriedades autoimplementadas simplificam o código	294
Use um setter privado para criar uma propriedade somente de leitura	295
E se quisermos mudar o tamanho do pente?	296
Construtor com parâmetros para inicializar propriedades	297
Especifique argumentos ao usar a palavra-chave "new"	298



RealName: "Herb Jones"  
 Alias: "Dash Martin"  
 Password: "the crow flies at midnight"

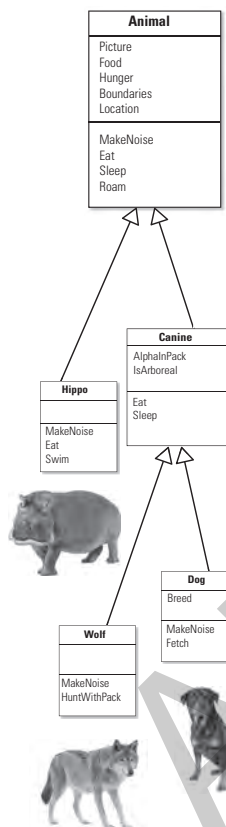
## 6

## herança

## Árvore genealógica do objeto

Algumas vezes você **SÓ** quer ser como seus pais.

Alguma vez encontrou uma classe que faz **quase** exatamente o que deseja que a **sua** classe faça? Já se pegou pensando que, se pudesse só **mudar algumas coisas**, essa classe seria perfeita? Com a **herança**, é possível **estender** uma classe existente para que a nova obtenha seu comportamento, com a **flexibilidade** de fazer alterações nesse comportamento para conseguir ajustá-lo como deseja. A herança é um dos conceitos e técnicas mais poderosos na linguagem C#: com ela você pode **evitar código duplicado**, **modelar o mundo real** com mais precisão e permanecer com apps **mais fáceis de manter** e **menos propensos a erros**.



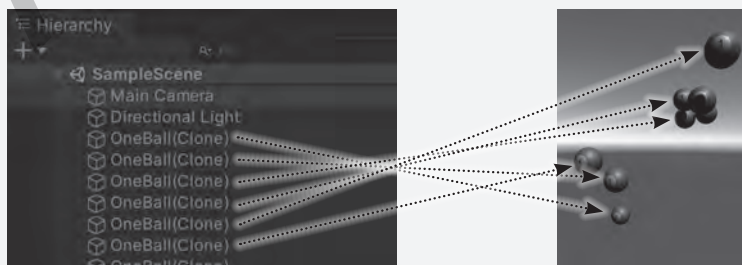
Calcule o dano para MAIS armas	312
Declaração switch para combinar várias candidatas	313
Mais uma coisa... podemos calcular o dano para uma adaga?	
Um bastão? Um cajado? E...	315
Quando suas classes usam a herança, você só precisa escrever o código uma vez	316
Crie um modelo de classe genérico e fique mais específico	317
Como você planejará um simulador de zoo?	318
Animais diferentes têm comportamentos diferentes	320
Toda subclasse estende sua classe básica	323
Se você pode usar uma classe básica, pode usar uma de suas subclasses	324
Dois pontos para estender uma classe básica	328
Sabemos que a herança adiciona campos, propriedades e métodos da classe básica à subclasse...	329
Uma subclasse pode anular os métodos para alterar ou substituir os membros herdados	330
Alguns membros só são implementados em uma subclasse	335
Use o depurador para saber como funciona a anulação	336
Crie um app para explorar virtual e override	338
Uma subclasse pode ocultar métodos na classe básica	340
Use override e virtual para herdar o comportamento	342
Uma subclasse pode acessar sua classe básica com a palavra-chave base	344
Quando uma classe básica tem um construtor, a subclasse precisa chamá-lo	345
Subclasse e classe básica com construtores diferentes	346
É hora de terminar o trabalho de Owen	347
Quando as classes se sobrepõem só um pouco, há um importante princípio chamado separação de conceitos	348
Crie um sistema de gerenciamento de colmeias	354
Modelo de classe do sistema de gerenciamento de colmeias	355
Classe Queen: como gerenciar as operárias	356
IU: adicione o XAML para a janela principal	357
Feedback orienta o jogo Beehive Management	366
O Beehive Management System é baseado em turnos...	
agora vamos convertê-lo em tempo real	368
Algumas classes nunca devem ser instanciadas	370
Uma classe abstrata é incompleta de propósito	372
Como dissemos, certas classes nunca devem ser instanciadas	374
Um método abstrato não tem corpo	375
Propriedades abstratas são como métodos abstratos	376

# Unity Lab 3

## Instâncias GameObject

C# é uma linguagem orientada a objetos e, como estes Use a Cabeça C# Unity Labs são todos **sobre praticar a escrita do código C#**, faz sentido que os laboratórios foquem a criação de objetos.

Vamos criar um jogo no Unity!	382
Crie um novo material na pasta Materials	383
Crie uma bola de bilhar em um ponto aleatório na cena	384
Use o depurador para entender Random.value	385
Torne o GameObject um prefab	386
Crie um script para controlar o jogo	387
Anexe o script à Main Camera	388
Pressione Play para executar o código	389
Use Inspector para trabalhar com as instâncias GameObject	390
Física para evitar a sobreposição das bolas	391
Seja criativo!	392



## 7

## interfaces, coerção e “is”

**Classes cumprindo suas promessas****Ações falam mais alto do que palavras.**

Às vezes você precisa agrupar seus objetos com base em **coisas que eles podem fazer**, não nas classes que eles herdam. É aí que entram as **interfaces**, elas lhe permitem trabalhar com qualquer classe que pode fazer o trabalho. Mas **com grandes poderes vêm grandes responsabilidades**, e qualquer classe que implementa uma interface deve se certificar de cumprir todas as suas obrigações... ou o compilador quebrará suas pernas, sabe?

DEFENDA  
A COLMEIA A TODO  
CUSTO.

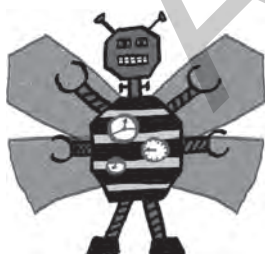


objeto Queen

SIM,  
SENHORA!



objeto HiveDefender



A colmeia está sendo atacada!	394
Usando a coerção para chamar o método DefendHive...	395
Uma interface define métodos e propriedades que uma classe deve implementar...	396
As interfaces permitem que classes não relacionadas façam o mesmo trabalho	397
Praticando com interfaces	398
Você não pode instanciar uma interface, mas pode referenciá-la	404
Referências da interface são referências do objeto comuns	407
RoboBee 4000 pode fazer o trabalho de uma operária sem usar o valioso mel	408
A propriedade Job de IWorker é uma correção	412
Use “is” para verificar o tipo do objeto	413
Use “is” para acessar métodos na subclasse	414
E se quiséssemos que animais diferentes nadassem ou caçassem em grupos?	416
Use interfaces para trabalhar com classes que fazem o mesmo trabalho	417
Navegue com segurança sua hierarquia de classes com “is”	418
O C# tem outra ferramenta para uma conversão do tipo segura: a palavra-chave “as”	419
Use upcast e downcast para subir e descer na hierarquia	420
Exemplo rápido de upcast	421
Upcast torna CoffeeMaker uma Appliance	422
Downcast retorna Appliance para CoffeeMaker	423
Upcast e downcast funcionam com interfaces também	424
Interfaces podem herdar de outras interfaces	426
Interfaces podem ter membros estáticos	433
Implementações-padrão fornecem corpos aos métodos das interfaces	434
Método ScareAdults com uma implementação-padrão	435
A vinculação de dados atualiza os controles WPF automaticamente	437
Modifique o Sistema de Gerenciamento de Colmeias para usar a vinculação de dados	438
Polimorfismo significa que um objeto pode ter muitas formas diferentes	441



# 8

## enums e coleções

### Organizando seus dados

**Desgraça pouca é bobagem.**

No mundo real, você não recebe dados organizados em pequenas partes. Não, os dados são recebidos em **grandes quantidades, pilhas e grupos**. Serão necessárias ferramentas poderosas para organizar todos eles, e é aí que entram os **enums** e as **coleções**. Enums são tipos que permitem definir valores válidos para ordenar seus dados. Coleções são objetos especiais que armazenam muitos valores, permitindo **armazenar, ordenar e gerenciar** todos os dados que seus programas precisam analisar. Assim, reserve um tempo para pensar sobre como escrever programas que trabalham com dados e deixe que as coleções se preocupem com como controlá-los.



Carta Duque de touros raramente jogada

Strings nem sempre funcionam para armazenar categorias de dados	444
Enums trabalham com um conjunto de valores válidos	445
Enums permitem representar números com nomes	446
Poderíamos usar um array para criar um baralho...	449
Pode ser chato trabalhar com arrays	450
Listas facilitam armazenar qualquer coleção...	451
Listas são mais flexíveis que arrays	452
Criaremos um app para armazenar sapatos	455
Coleções genéricas armazenam qualquer tipo	458
Inicializadores de coleção se parecem com inicializadores de objeto	464
Vamos criar uma lista de patos	465
Listas são fáceis, mas ORDENAR pode ser complicado	466
IComparable<Duck> ajusta List a ordenar Ducks	467
Use IComparer para informar a List como ordenar	468
Crie uma instância do objeto de comparação	469
Comparadores fazem comparações complexas	470
Anular o método ToString permite ao objeto se descrever	473
Atualize os loops foreach para que Ducks e Cards se escrevam no console	474
Use Dictionary para armazenar chaves e valores	480
Resumo das funcionalidades do dicionário	481
Crie um programa que usa um dicionário	482
E ainda MAIS tipos de coleção...	483
Uma fila é FIFO — primeiro a entrar, primeiro a sair	484
Uma pilha é LIFO — último a entrar, primeiro a sair	485
Exercício para download: Two Decks	490

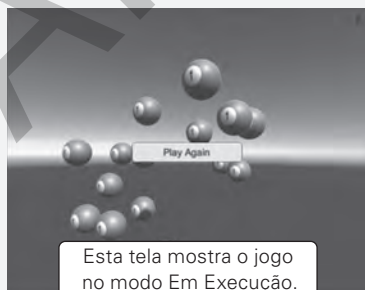


# Unity Lab 4

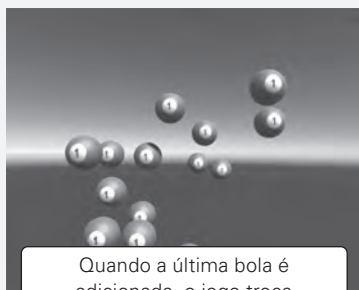
## Interfaces do Usuário

No último Unity Lab você começou a montar um jogo, usando um prefab para criar instâncias `GameObject` que aparecem em pontos aleatórios no espaço em 3D do jogo e voam em círculos. Este Unity Lab continua do ponto em que o último parou, permitindo que você aplique o que aprendeu sobre interfaces no C# e muito mais.

Adicione uma pontuação que aumenta quando o jogador clica em uma bola	492
Adicione dois modos diferentes ao jogo	493
Adicione um modo ao jogo	494
Adicione uma IU ao jogo	496
Configure Text para exibir a pontuação na IU	497
Adicione um botão que chama um método para iniciar o jogo	498
Faça os botões Play Again e Score Text funcionarem	499
Termine o código do jogo	500
Seja criativo!	504



Esta tela mostra o jogo no modo Em Execução. Bolas são adicionadas e o jogador pode clicar nelas para pontuar.



Quando a última bola é adicionada, o jogo troca para o modo Game Over. O botão Play Again aparece e nenhuma outra bola é adicionada.

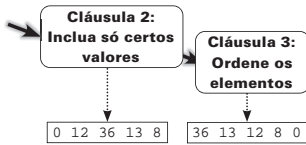
# 9

## LINQ e lambdas

### Controle seus dados

**Você está pronto para um novo mundo de desenvolvimento de aplicativo.**

Usar o WinForms para criar aplicativos para Desktop é uma ótima forma de aprender conceitos do C#, mas há  *muito mais*  que você pode fazer com os seus programas. Neste capítulo, você usará **XAML** para criar seus aplicativos para Windows e aprenderá a **construir** páginas que servirão para qualquer aparelho, a **integrar** seus dados nas páginas com **vinculação de dados** e a **usar** o Visual Studio para resolver o mistério das páginas XAML ao explorar objetos criados com o seu código XAML.



Jimmy é superfã do Captain Amazing...	506
...mas a coleção dele está por todo lado	507
Use LINQ para consultar as coleções	508
LINQ trabalha com qualquer IEnumerable<T>	510
Sintaxe da consulta de LINQ	513
O LINQ trabalha com objetos	515
Use uma consulta LINQ para terminar o app de Jimmy	516
Var permite que C# descubra os tipos da variável	518
O LINQ é versátil	524
Consultas LINQ não executadas até acessar os resultados	525
Consulta de grupo para separar a sequência em grupos	526
Consultas join para combinar dados de duas sequências	529
Use a palavra-chave new para criar tipos anônimos	530
Testes unitários para assegurar um código funcional	538
Projeto de teste unitário para o app da coleção de Jimmy	540
Escreva seu primeiro teste unitário	541
Escreva um teste unitário para o método GetReviews	543
Testes unitários para casos extremos e dados estranhos	544
Use o operador => para criar expressões lambda	546
Test drive com lambda	547
Refatore clown com lambdas	548
Use o operador ?: para lambdas fazerem escolhas	551
Expressões lambda e LINQ	552
Consultas LINQ escritas como métodos LINQ encadeados	553
Use o operador => para criar expressões switch	555
Explore a classe Enumerable	559
Crie uma sequência enumerável à mão	560
Use yield return para criar suas próprias sequências	561
Use yield return para refatorar ManualSportSequence	562
Exercício para download: Go Fish	566

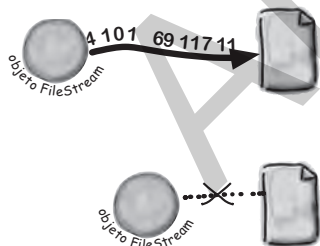
# 10 lendo e gravando arquivos

## Salve o último byte para mim!

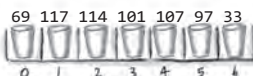
Às vezes compensa ser um pouco persistente.

Até agora, todos os seus programas tiveram vida curta. Eles inicializaram, rodaram por um tempo e finalizaram. Mas isso nem sempre é suficiente, sobretudo ao lidar com informações importantes. Você precisa conseguir **salvar seu trabalho**. Neste capítulo, veremos como **gravar dados em um arquivo** e, então, **ler essas informações de volta** no arquivo. Você aprenderá sobre **fluxo** e também verá os mistérios dos **hexadecimais** e dos **binários**.

	.NET usa fluxos para ler e gravar dados	568
	Fluxos diferentes leem e gravam coisas diferentes	569
	FileStream lê e grava bytes em um arquivo	570
	Gravar texto no arquivo em três etapas simples	571
	Swindler inicia outro plano diabólico	572
	StreamReader para ler um arquivo	575
	Os dados podem passar por mais de um fluxo	576
	Use as classes File e Directory estáticas para trabalhar com arquivos e diretórios	580
	IDisposable fecha os objetos corretamente	583
	MemoryStream para enviar dados para a memória	585
	O que acontece com um objeto quando ele é serializado?	591
	O que é exatamente o estado de um objeto? O que precisa ser salvo?	592
	Use JsonSerializer para serializar objetos	594
	JSON inclui apenas dados, não tipos C# específicos	597
	A seguir: nós nos aprofundaremos nos dados	599
	Strings do C# codificadas com Unicode	601
	Visual Studio trabalha bem com o Unicode	603
	.NET usa o Unicode para armazenar caracteres e texto	604
	C# pode usar arrays de bytes para mover dados	606
	Use BinaryReader para ler dados de volta	608
	Use BinaryWriter para gravar dados binários	607
	Um dump hex permite ver os bytes nos arquivos	610
	Use Stream.Read para ler bytes em um fluxo	612
	Modifique dumper hex para usar argumentos da linha de comando	613
	Exercício para download: Hide and Seek	614



Eureka! →





# Unity Lab 5

## Raycast

Ao configurar uma cena no Unity, você cria um mundo virtual em 3D para os personagens no jogo se moverem. Mas, na maioria dos jogos, grande parte das coisas não é controlada diretamente pelo jogador. Então, como esses objetos encontram seu caminho na cena? Neste Lab, olharemos como o C# pode ajudar.

Crie um novo projeto Unity e comece a preparar a cena	616
Configure a câmera	617
Crie um GameObject para o jogador	618
Sistema de navegação do Unity	619
Configure NavMesh	620
Jogador se move automaticamente na área do jogo	621



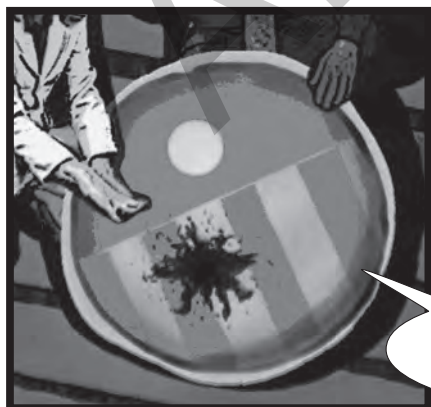
# CAPTAIN AMAZING

## THE DEATH OF THE OBJECT

Use a Cabeça C#

Quatro  
dólaresCapítulo  
11

The life and death of an object	625
Use a classe GC (com cuidado) para forçar a coleta	629
Sua última chance de FAZER algo... o finalizador do objeto	630
Quando EXATAMENTE um finalizador é executado?	631
Finalizadores não podem depender de outros objetos	633
Struct lembra um objeto...	637
...mas não é um objeto	637
Valores são copiados; referências são atribuídas	638
Structs são tipos de valor; objetos são tipos de referência	639
Pilha versus heap: mais sobre memória	641
Use parâmetros out para fazer um método retornar mais de um valor	644
Passe por referência usando o modificador ref	645
Use parâmetros opcionais para definir valores-padrão	646
Referência nula não se refere a nenhum objeto	647
Tipos de referência não nula ajudam a evitar NREs	648
Operador de coalescência nula ??	649
Tipos de valor nullable podem ser nulos... e lidados com segurança	650
"Captain" Amazing... nem tanto	651
Métodos de extensão adicionam novo comportamento às classes EXISTENTES	655
Estendendo um tipo fundamental: string	657



SÓ... PRECISA DE...  
— SUSPIRO —  
UMA... ÚLTIMA... COISA...

# 12 tratamento de exceção

## Apagar incêndio é cansativo

Programadores não são bombeiros.

Você se esforçou muito examinando manuais técnicos e alguns livros *Use a Cabeça* interessantes e chegou no auge da sua profissão. Mas ainda recebe ligações de emergência do trabalho no meio da noite porque **seu programa trava** ou **não se comporta como deveria**. Nada o tira mais da rotina de programação do que ter que corrigir um bug estranho... mas, com o **tratamento de exceção**, é possível escrever um código para **lidar com os problemas** que surgem. Melhor ainda, você pode até se planejar para esses problemas e **manter as coisas funcionando** quando elas acontecem.

Seu dumper hex lê um nome de arquivo na linha de comando 662

Quando o programa gera uma exceção, a CLR gera um objeto Exception 666

Todos os objetos Exception herdam de System.Exception 667

Existem arquivos que não podem ser despejados 670

E quando um método que você quer chamar é arriscado? 671

Lide com exceções usando try e catch 672

Use o depurador para seguir o fluxo try/catch 673

Se você tem um código que SEMPRE precisa ser executado, use um bloco finally 674

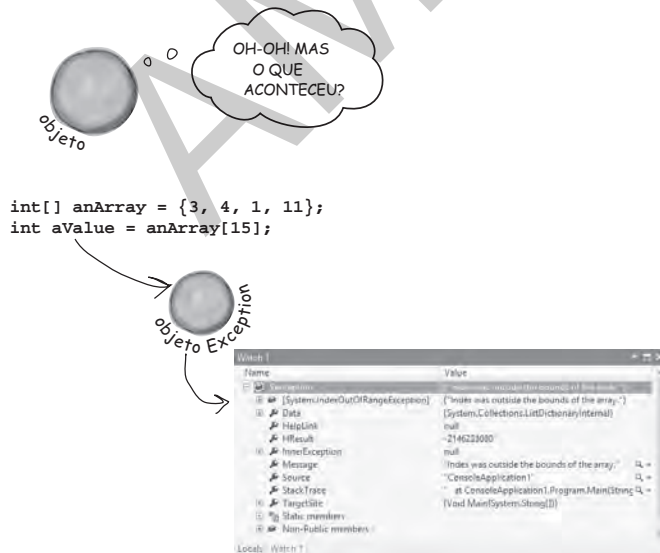
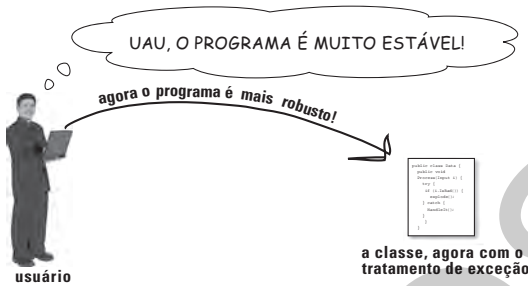
Exceções catch genéricas lidam com System.Exception 675

Use a exceção certa para a situação 680

Filtros de exceção ajudam a criar tratamentos precisos 684

O pior bloco catch de TODOS: genérico mais comentários 686

Soluções temporárias são boas (temporariamente) 687

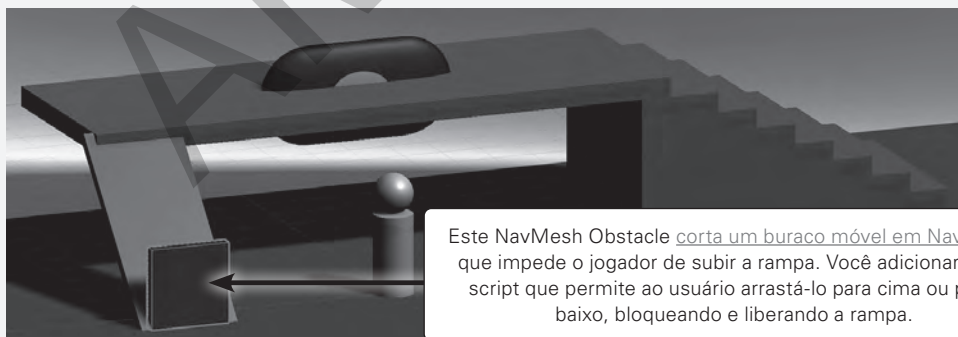


# Unity Lab 6

## Navegação da Cena

No último Unity Lab, você criou uma cena com um piso (plano) e um jogador (uma esfera aninhada em um cilindro), usou NavMesh, um NavMesh Agent, e usou o raycast para fazer o jogador seguir os cliques do mouse na cena. Neste lab, você adicionará à cena com a ajuda do C#.

Do ponto em que paramos no último Unity Lab	690
Adicione uma plataforma à cena	691
Prepare a plataforma para ser acessível	692
Inclua os degraus e a rampa em NavMesh	693
Corrija problemas de altura em NavMesh	695
Adicione um Obstáculo NavMesh	696
Adicione um script para subir e descer o obstáculo	697
Seja criativo!	698



Este NavMesh Obstacle [corta um buraco móvel em NavMesh](#) que impede o jogador de subir a rampa. Você adicionará um script que permite ao usuário arrastá-lo para cima ou para baixo, bloqueando e liberando a rampa.

## apêndice i: projetos Blazor do ASP.NET Core

### Guia do Aluno Visual Studio para Mac

# i



Por que você deve aprender o C#	702
Visual Studio é uma ferramenta para escrever código e para explorar a linguagem C#	703
Crie seu primeiro projeto no Visual Studio para Mac	704
Use o IDE Visual Studio para explorar seu app	706
Crie um jogo!	708
Como criar seu jogo	709
Crie um App Blazor WebAssembly no Visual Studio	710
Rode seu app Web Blazor em um navegador	712
Tudo pronto para começar a escrever o código do jogo	714
O Visual Studio lhe ajuda a escrever o código C#	716
Termine de criar a lista de emojis e exiba-a no app	718
Embaralhe os animais para ficarem em ordem aleatória	720
Você está rodando o jogo no depurador	722
Adicione seu novo projeto ao controle de versão	726
Código C# para lidar com os cliques do mouse	727
Adicione manipuladores de evento aos botões	728
Teste o manipulador de evento	730
Use o depurador para resolver o problema	731
Continue depurando o manipulador de evento	732
Rastreie o bug que causa o problema...	734
Adicione código para reiniciar o jogo quando o jogador vence	736
Termine o jogo adicionando um cronômetro	739
Adicione um cronômetro ao código do jogo	740
Limpe o menu de navegação	742
Ainda melhor se...	743
Os controles orientam a mecânica das IUs	744
Crie um novo projeto App Blazor WebAssembly	745
Crie uma página com um controle deslizante	746
Adicione uma entrada de texto ao app	748
Adicione seletores de cor e data ao app	751
A seguir: versão Blazor do app para selecionar cartas	752
Layout da página com linhas e colunas	754
A barra deslizante usa a vinculação de dados para atualizar uma variável	755
Bem-vindo à lanchonete Sandubas Preço Bom É Aqui de Sloppy Joe!	758
Acesse o material online dos Capítulos 5 e 6	762

# ii

## apêndice ii: Código Kata

### Guia do Código Kata para Avançados e/ou Impacientes



como usar este livro

## Introdução

NEM ACREDITO  
QUE COLOCARAM **ISTO**  
EM UM LIVRO DE  
PROGRAMAÇÃO C#!



Nesta seção, respondemos à pergunta que não quer calar:  
“Por que puseram ISTO em um livro de programação C#?”

## A quem se destina este livro?

Se responder “sim” a todas estas perguntas:

- 1 Você quer **aprender** o **C#** (e conhecer um pouco sobre desenvolvimento de jogos e Unity no processo)?
- 2 Você gosta de xeretar? Aprende mais fazendo do que apenas lendo?
- 3 Você prefere **jantares com conversas estimulantes a palestras acadêmicas cansativas e frias**?


Este livro é para você.

## Quem provavelmente deve fugir do livro?

Se responder “sim” a qualquer uma destas perguntas:

- 1 Você se interessa mais pela teoria do que pela prática?
- 2 A ideia de fazer projetos e escrever código deixa-o entediado ou um pouco nervoso?
- 3 Você tem **medo de testar algo diferente**? Acha que um livro sobre um assunto sério, como desenvolvimento, precisa ser sério o tempo todo?

Pense em outro livro primeiro.



### Caminho de aprendizagem do código Kata

Você é um **desenvolvedor avançado** com experiência em várias linguagens e deseja *acelerar* no C# e no Unity?

É um **aluno impaciente** que fica à vontade indo direto para o código?

Se respondeu **SIM!** às duas perguntas, concluímos que um caminho de aprendizagem do **código kata** é para você. Procure o apêndice **Código Kata** no fim deste livro para saber mais.

### PRECISO APRENDER OUTRA LINGUAGEM DE PROGRAMAÇÃO PARA USAR ESTE LIVRO?

**Muitas pessoas aprendem o C# como uma segunda linguagem (terceira ou décima quarta), mas você não precisa ter escrito muito código para começar.**

Se você escreveu programas (mesmo pequenos!) em *qualquer* linguagem de programação, fez uma aula de programação básica na escola ou online, fez shell script ou usou uma linguagem de consulta do banco de dados, **com certeza** tem conhecimento para este livro e se sentirá em casa.

E se tem **menos experiência**, mas ainda quer aprender o C#? Milhares de iniciantes, sobretudo os que criaram páginas Web antes ou usaram as funções do Excel, usaram este livro para aprender o C#. Mas, se você é totalmente novo, recomendamos considerar o livro *Use a Cabeça Aprenda a Programar* de Eric Freeman.

Se ainda está em dúvidas se o *Use a Cabeça C#* é ou não adequado para você, tente fazer os quatro primeiros capítulos. Pode baixá-los no site da Alta Books ou acessar <https://github.com/head-first-csharp/fourth-edition> [conteúdo em inglês]. Se se sentir à vontade depois disso, então escolheu o livro certo! Se ficar com a cabeça girando, deve considerar ler o *Use a Cabeça Aprenda a Programar*; depois dele, estará 100% pronto para este livro.

## Sabemos o que você pensa.

“Como *isto* pode ser um livro de programação C# sério?”

“Por que todos esses gráficos?”

“Será que eu posso realmente *aprender* desta forma?”

## Sabemos o que seu cérebro pensa.

Seu cérebro busca novidades. Está sempre procurando, vasculhando, *esperando* algo diferente. Ele foi feito assim e isso o ajuda a continuar vivo.

Então, o que seu cérebro faz com todas as coisas rotineiras, comuns e normais que encontra? Tudo o que *pode* fazer para impedi-las de interferir no trabalho *real*, ou seja, registrar coisas *importantes*. Ele não se preocupa em gravar coisas chatas. Elas nunca passam pelo filtro “isto obviamente não é importante”.

Como seu cérebro *sabe* o que é importante? Imagine que um dia você está fazendo uma trilha e um tigre pula na sua frente: o que acontece dentro da sua cabeça e do seu corpo?

Os neurônios disparam. As emoções saem de controle. *A química do corpo intensifica-se.*

E é assim que seu cérebro sabe...

### **Isto deve ser importante! Não se esqueça!**

Mas imagine que você está em casa ou na biblioteca. É um lugar seguro, agradável e sem tigres. Você está estudando, preparando-se para uma prova ou tentando aprender algum assunto técnico complicado que seu chefe acha que levará uma semana, dez dias no máximo.

Só há um problema. Seu cérebro está tentando fazer um grande favor. Está tentando ter certeza de que este conteúdo *obviamente* inútil não fique empacando outros recursos escassos. Recursos que são mais bem utilizados guardando coisas realmente *importantes*. Como tigres. Como o perigo que o fogo representa. Como o fato de que você nunca deveria ter colocado as fotos daquela “festa” em sua página do Facebook.

E não há forma simples de dizer: “Ei, cérebro, muito obrigado, mas não importa o quanto este livro é chato e o quão pouco estou registrando-o na escala Richter emocional agora; eu *realmente* quero que você guarde essas coisas”.

Seu cérebro pensa que ISTO é importante.



ÓTIMO. SÓ MAIS  
700 PÁGINAS MAÇANTES,  
SEM GRAÇA E CHATAS.

Seu cérebro pensa  
que ISTO não vale  
a pena gravar.





## Imaginamos o leitor "Use a Cabeça" como um aprendiz.

O que é preciso para *aprender* algo? Primeiro, você tem que *captar* e, depois, não *esquecer*. Não é apenas enfiar fatos em sua cabeça. Com base nas últimas pesquisas em Ciência Cognitiva, Neurobiologia e Psicologia Educacional, *aprender* requer muito mais do que um texto em uma página. Nós sabemos o que estimula o seu cérebro.

### Alguns princípios de aprendizagem da série Use a Cabeça:

**Torne-o visual.** Imagens são muito mais fáceis de memorizar do que apenas palavras, e tornam a aprendizagem muito mais eficiente (até 89% de melhoria em estudos de memória e de transferência de informações) e as coisas mais compreensíveis.

objeto Dog



Todos os elementos no array são referências. O array em si é um objeto.

**Coloque as palavras dentro ou perto dos gráficos** aos quais se relacionam, em vez de no final ou em outra página, e os leitores terão uma probabilidade quase *dobrada* de resolver problemas relativos ao conteúdo.

### Utilize um estilo conversacional e

**personalizado.** Em estudos recentes, alunos foram até 40% melhores em testes pós-aprendizagem se o conteúdo era dirigido diretamente ao leitor, usando a primeira pessoa e um estilo conversacional, em vez de um tom formal. Conte histórias no lugar de ensinar. Use uma linguagem casual. Não se leve tão a sério. Em que  *você* prestaria mais atenção: no colega em um jantar estimulante ou em uma palestra?

FAÇA TODAS AS MINHAS REFEIÇÕES NO SLOPPY JOE'S!



**Faça a pessoa pensar mais a fundo.** A menos que você exercite ativamente seus neurônios, pouca coisa acontece em sua cabeça. Um leitor tem que ser motivado, engajado, ficar curioso e inspirado a resolver problemas, chegar a conclusões e gerar um novo conhecimento. E, para tanto, é preciso ter desafios, exercícios, perguntas provocadoras e atividades que envolvam os dois lados do cérebro e os diversos sentidos.

**Chame a atenção do leitor, e a mantenha.** Todos nós já passamos pela experiência de "eu realmente quero aprender isto, mas não consigo passar da página um sem dormir". Seu cérebro presta atenção em coisas extraordinárias, interessantes, estranhas, que prendem a atenção e que são imprevisíveis. Aprender sobre um assunto técnico novo e difícil não precisa ser chato. Seu cérebro aprenderá muito mais facilmente se não for.

**Toque suas emoções.** Hoje, nós sabemos que a habilidade de lembrar algo depende muito de seu conteúdo emocional. Você se lembra de coisas com as quais se importa. Lembra quando *sente* algo. Não, não estamos falando a respeito daquelas histórias de cortar o coração sobre um menino e seu cachorro, mas sobre emoções, tais como surpresa, curiosidade, diversão, "mas que...?", e o sentimento de "agora eu peguei!" que surge quando você resolve um problema, aprende algo que todo mundo pensa que é difícil ou quando percebe que aprendeu *algo novo e incrível*, sentindo-se bem quando usa esse conhecimento.



Até emoções de medo podem ajudar a fixar as ideias na mente.

