PROGRAMAÇÃO DE COMPUTADORES V - TCC- 00.323

Modulo 8: pequenos detalhes, grandes diferenças

Aura - Erick
aconci@ic.uff.br, erickr@id.uff.br

Roteiro

- Conversão de tipo
- Operador ternário condicional - ? ; ;
- Comandos do Preprocessador
- Macros
- **tabela AISCII**
- Strings = vetores de caracteres
- Parâmetros da função **main ()**
- **Uso de arquivos**
- **Cores no computador!**

Conversão de tipo

Quando tipos diferentes são misturadas, os dados são **convertidos automaticamente** para o de maior representatividade, ou para o definido pelo usuário.

Exemplo: float a=3;

Mas uma requisição para o **tipo que o programador quer** pode ser feita através do operador ***cast***:

```
int a;
.....
a= (int) 3.5 % 2;
```

Operador condicional - ? ; ;

- Sua forma é:

Condição ? Expressão 1 ; Expressão 1 ;

- Exemplo: a>b ? a ; b ;
- Equivale a : if (a>b) a else b
- E pode ser usado direto por exemplo em atribuições

Max= a>b ? a ; b ;

O que equivale a:

if (a>b) Max= a

else Max= b

Comandos do Preprocessador

- Comandos do **pré processador** são executados antes da compilação.
- Eles **substituem** comandos complicados por comandos **mais simples, definem e testam constantes e incluem arquivos de definições.**
- Como tudo isso acontece antes da compilação qualquer comando em tempo de execução não pode usá-los.

- Exemplo:
- #include “arquivo.txt” (“ inicia a busca no diretório local)
- **#include** <stdio.h> (<> =>procura no diretório padrão da linguagem C)

Comandos do Preprocessador : **MACROS**

- **# define** Plaprox 3.14159
- Faz com que toda a ocorrencia de “Plaprox” no programa seja substituida por 3.14159...

- Esse comando por receber parâmetros (**macros**)
- # define Max(a,b) (a>b ? a ; b ;)

- De modo que onde aparecer
- S=4.5; r=3.5 ; T=Max(S,r) se terá na verdade:
- T= S>r ? S ; r ;

Tabela ASCII

- Se só podemos armazenar números no computados como de podem ter um texto, uma mensagem, um documento?
  - Para caracteres e símbolos se atribui a cada um código específico.

- Em geral se usa a tabela ASCII e para vê-lo você pode escrever:

```
char  verCodLetra=`a´;
printf (“%5d  %5c  \n”, verCodLetra , verCodLetra);
```

ASCII

- (*American Standard Code for Information Interchange* = "Código Padrão Americano para o Intercâmbio de Informação") é um código binário que codifica um conjunto de 128 sinais: **95 sinais gráficos** (letras do alfabeto latino, sinais de pontuação e sinais matemáticos) e **33 sinais de controle**. **Cada código binário possui 8 bits** , sendo 7 bits para o propósito de codificação e 1 bit de paridade (detecção de erro).
- A codificação ASCII é usada para representar textos em computadores e outros dispositivos que trabalham com texto. Desenvolvida no anos 1960, e outras codificações de caracteres a herdaram como base.
- Os sinais não-imprimíveis, conhecidos como **caracteres de controle**, são amplamente utilizados em dispositivos de comunicação e afetam o processamento do texto.

Tabela AISCII

- Exemplo de parte....

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
32		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
48	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
96	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

- Obs. Símbolos, números , Maiúsculas, minúsculas separados em grupos

Strings = vetores de caracteres

- Strings em C são vetores de caracteres com o `\0`**no final** (**caracter nulo** !)

- Consequencia:
  - **Strings em C** devem ter um espaço a mais do que o texto que aparecerá escrito!
  - Esse é usado nas diversas funções que manipulam **strings**: Por isso :
  - as constantes de **string** precisam estar entre “ “
  - E os **caracteres** individuais entre ` `

Parâmetros da função **main ()**

- a função **main () pode ter Zero ou 2 parâmetros ou argumentos : argc e argv**

- **Argc -> números de argumentos a ser passado quando esse é executado!**

- **Argv - > vetor de caracteres que armazena os nomes passado como parâmetro**

Argumentos - argc e argv

- A função main como todas as funções podem ter **argumentos**.
- Como a função main é sempre a primeira a ser executada, os parâmetros que ela recebe são fornecidos pela **linha de comando.**
- Ou seja junto ao **nome do programa**!
- Assim ela pode ter **zero** argumentos ou (2) **dois argumentos** especiais : **argc e argv**.
- O primeiro argumento, **argc**, é uma variável inteira que indica quantos argumentos foram fornecidos para a função. Observar que argc vale sempre pelo menos 1, porque o nome do programa é sempre o primeiro argumento fornecido ao programa. A partir do segundo argumento em diante é que aparecem os outros

Exemplo com zero já vimos muitos:

```
#include <stdio.h>
#define PI 3.14159
float volume_cilindro (float raio, float altura)
    { float volume;
    volume = PI * raio* raio* altura;
    return volume; }
int main(void) {
    float raio, altura, volume;
    printf(“Entre com o valor do raio: “);
    scanf(“%f”, &raio);
    printf(“Entre com o valor da altura: “);
    scanf(“%f”, &altura);
    volume = volume_cilindro(raio, altura);
    printf(“Volume do cilindro = “);
    printf(“%f”, volume);
    return 0; }
```

Exemplo com 2 :

- Isso é como:

int main (int argc, char *argv[])

- Ou
- void main (int argc, char *argv[])
- Etc...

main pode ter a seguinte forma:

```
#include <stdio.h>
int main (int argc, char *argv[])
{
    int a;
    for (i=0,i< argc; i++)
            printf("%s \n", argv[i]));
    return 0;
}
```

ExemploBobo para ver argumentos

Se o programa tiver seu executável chamado de **ExemploBobo**

E for invocado com a linha de comando:

ExemploBobo para ver argumentos

voce verá:

Funções de **entrada e saída em arquivos**

- Motivação:
- quando um programa precisa processar um volume de dados muito grande e.g.: 500 dados
- Imagine ter que digitar 499 dados novamente por causa de 1 simples erro de digitação?

- A estratégia adequada para se trabalhar com grande volume de dados é fazer uso de **arquivos de dados**.

- Para que programas possam **ler e salvar dados em um arquivo**, é necessário ter acesso a um conjunto de funções que permitam realizar essas operações:
- Como o conjunto de funções presentes na biblioteca **stdio.h.**

Funções de **entrada e saída em arquivos**

- Para que possamos ler ou escrever em um arquivo é necessário que antes o arquivo esteja aberto.
- Cada arquivo é identificado pelo seu nome, que normalmente é o nome do arquivo mais a extensão: exemplo.txt, cap8.ppt, saida.txt
- Para abrir um arquivo é necessário definir se o mesmo será utilizado para leitura ou escrita:
  - Para abrirmos um arquivo para leitura é necessário que o arquivo já exista.
  - Quando abrimos um arquivo para escrita criamos um novo arquivo onde dados de saída serão escritos, ou sobrescrevemos um arquivo existente.

Funções de **entrada e saída em arquivos**

Para abrir um arquivo, a biblioteca padrão oferece a função **fopen**.

- Esta função serve tanto para abrir um arquivo para leitura como para escrita.
- A função recebe dois parâmetros, o nome do arquivo que se deseja abrir e o modo de abertura:
  - » Leitura: "r" (read)
  - » Escrita: "w" (write).
- Esta função retorna um ponteiro para o tipo FILE definido na biblioteca. Um ponteiro é um tipo de variável que serve para armazenar endereços de memória.

Funções de **entrada e saída em arquivos**

Declaração de uma variável do tipo ponteiro para **FILE** (no exemplo, o nome dado à variável é **fp**):

- `FILE *fp;`

Inicialização:

- **Abrindo o arquivo para leitura de dados:**
  - » `fp = fopen("entrada.txt", "r");`
- **Abrindo o arquivo para escrita de dados:**
  - » `fp = fopen("saida.txt", "w");`

Funções de **entrada e saída em arquivos**

Se a abertura do arquivo não for bem sucedida, a função retorna o valor definido pela constante simbólica NULL:

```
...
fp = fopen("entrada.txt", "r");
if (fp == NULL) {
  printf("Erro na abertura do arquivo.\n");
  exit(1); /* aborta programa */
}
...
```

O valor retornado pela função fopen deve ser passado como parâmetro para as demais funções para identificar em qual arquivo se deseja fazer a operação de entrada ou saída.

Após realizar as operações de entrada e saída no arquivo, este deve ser **fechado** com o uso da função `fclose`.

```
...
fclose(fp);
...
```

Funções de **entrada por arquivo**

A principal função da biblioteca padrão para leitura de arquivos chama-se **fscanf**, e é análoga à função scanf para captura de dados via teclado.

```
int a;
float b;
FILE* fp = fopen("entrada.txt", "r");
. . .
fscanf(fp, "%d %f", &a, &b);
. . .
fclose(fp);
```

A função fscanf tem como valor de retorno o número de parâmetros que foram lidos com sucesso (também vale para a função scanf).

Funções de **entrada por arquivo**

- Vamos considerar a existência de um arquivo que armazena notas que os alunos obtiveram em uma disciplina.
- Vamos considerar que a primeira linha contém um número inteiro que informa a quantidade de notas armazenadas a seguir, estando uma nota por linha.
- Arquivo notas.txt:

```
6
7.5
8.4
9.1
4.0
5.7
4.3
```

Funções de **entrada por arquivo**

- O nosso programa vai ler as notas do arquivo e escrever na tela a média dos alunos.

```
#include <stdio.h>

int main (void) {
  int i, n;
  float nota, soma = 0.0;
  FILE *fp ;

  /* abertura do arquivo para leitura */
  fp = fopen ("notas.txt", "r");
  /* teste para verificar se houve algum erro */
  if (fp == NULL) {
    printf("Erro na abertura do arquivo.\n");
    return 1; /* aborta programa ( retorna da função main) */
  }
```

Funções de **entrada por arquivo**

(continuação)

```
  /* leitura da quantidade de notas no arquivo */
  fscanf( fp , "%d", &n);
  /* laço para leitura de cada nota */
  for (i=0; i<n ; i++) {
    fscanf(fp, "%f ", &nota);
    soma = soma + nota ;
  }

  /* fechamento do arquivo */
  fclose(fp);

  /* cálculo da média e impressão na tela */
  printf("Media = %.2f \n", soma / n);
  return 0;
}
```

Funções de **entrada por arquivo**

- Quando escrevemos programas para processar dados de entrada de um arquivo, procuramos manter o formato dos dados presentes no arquivo o mais flexível possível.
- O formato do arquivo de notas do exemplo anterior não é muito flexível pois se formos adicionar uma nota no arquivo teremos também que atualizar a primeira linha do arquivo, que representa a quantidade de notas.
- Para tornar o formato mais flexível, podemos pensar em eliminar a informação da primeira linha, isto é, podemos listar apenas as notas, e o programa deve ser capaz de exibir a média de todas as notas.

```
7.5
8.4
9.1
4.0
5.7
4.3
```

Neste caso utilizaremos o comando **while** para processar as notas do arquivo enquanto o final do arquivo não é alcançado.
Para cada nota lida incrementamos o valor do contador.

```c
#include <stdio.h>

int main (void) {
  int n;
  float nota, soma = 0.0;
  FILE *fp ;

  /* abertura do arquivo */
  fp = fopen ("notas.txt", "r");

  n = 0;

  /* laço para leitura de cada nota */
  while (fscanf(fp, "%f ", &nota) == 1) {
    soma = soma + nota;
    n++;
  }

  /* fechamento do arquivo */
  fclose(fp);

  /* cálculo da média e impressão na tela */
  if (n > 0) {
    printf("Media = %.2f \n", soma / n);
  }
  return 0;
}
```

Funções de **saida por arquivo**

- A principal função de saída em arquivo da biblioteca padrão chama-se **fprintf**, sendo análoga à função printf para saída na tela.
  - A diferenca é que a função fprintf espera um primeiro parâmetro adicional que identifica o arquivo.

```
int a ;
float b ;
FILE* fp = fopen("saida.txt", "w");
. . .
fprintf(fp, "Valores : %d %f \n", a, b);
. . .
fclose(fp);
```

  - Obviamente os valores de a e b precisam ser definidos antes da chamada à função fprintf.

Funções de **escrita em arquivo**

- Vamos considerar que numa disciplina cada aluno faz 3 provas, obtendo três notas que são consideradas para o cálculo da nota final.
- Considere que as notas obtidas para cada aluno estão armazenadas no arquivo “entrada.txt”

```
7.5   8.5   7.8
8.4   9.2   6.8
9.1  10.0   9.5
4.0   5.2   4.6
5.7   3.4   4.3
4.3   6.0   5.8
```

Funções de **escrita em arquivo**

- Vamos então desenvolver um programa que processe as notas do arquivo “entrada.txt”.
- Para cada aluno, o programa deve calcular a sua nota final (***nf = (p1+p2+p3)/3***) e verificar se o aluno foi aprovado ou reprovado.
- O programa deve gerar um arquivo de saída com o nome “saida.txt”. Neste arquivo de saída, para cada linha do arquivo de entrada, deve-se escrever a linha correspondente com a nota final do aluno e sua situação: A se o aluno foi aprovado (media >=5.0) ou R se o aluno foi reprovado.

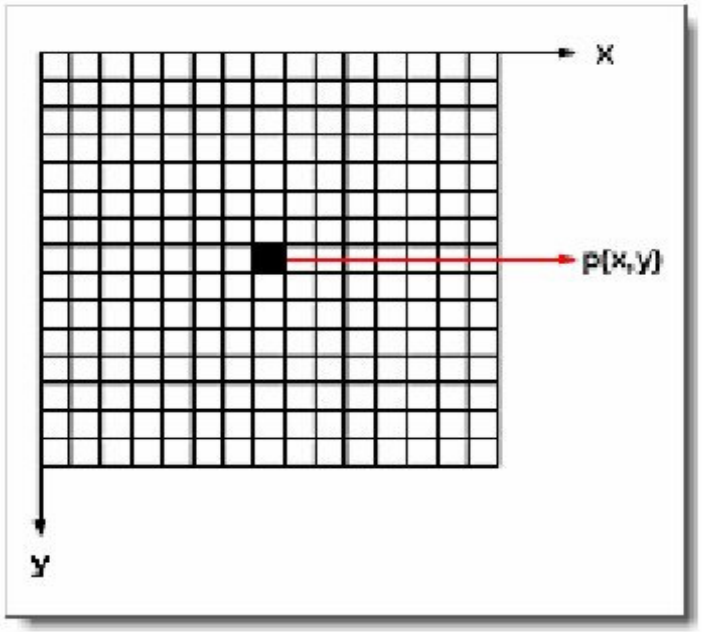
Funções de **escrita em arquivo**

```
#include <stdio.h>

int main (void) {
  float p1, p2, p3, media;
  FILE *ent, *sai;
  ent = fopen("entrada.txt", "r");
  sai = fopen("saida.txt", "w");
  while (fscanf(ent, "%f %f %f", &p1, &p2, &p3) == 3) {
    media = (p1 + p2 + p3) / 3;
    fprintf(sai, "%.1f ", media);
    if (media >= 5.0)
      fprintf(sai, "A\n");
    else
      fprintf(sai, "R\n");
}
  fclose(ent);
  fclose(sai);
  return 0;
}
```

Desenhos e cores no computador

Uma imagem digital é descrita por uma **matriz N x M** de valores de ***pixels*** (*p (x,y)*) que é como são chamados inteiros positivos, que indica a **intensidade de cor** , t , em cada posição [x,y] da imagem.



Imagens reais - > Digitais

Assim para que sejam representadas no meio **digital**, a imagem contínua tem que ser convertido numa série de valores discretos (descontínuos).

Esses valores são números (dígitos) que representam amostras (samples em inglês)

Amostragem

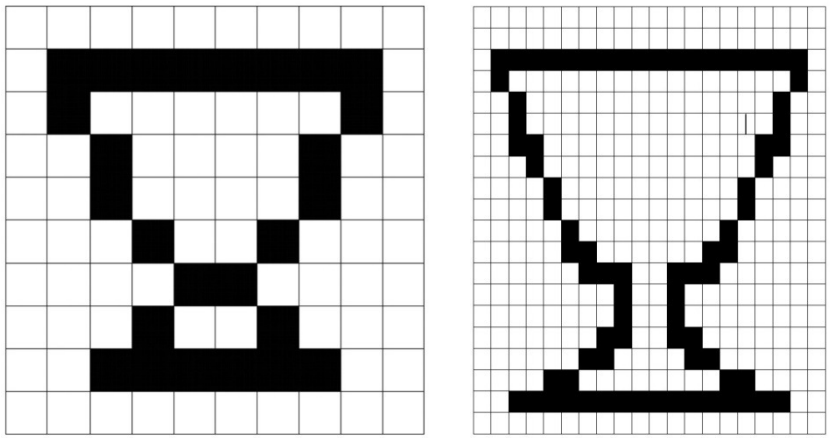
A conversão do sinal analógico para o digital é realizada por uma sequência de amostras da variação de voltagem do sinal original.

Cada amostra é arredondada para o número mais próximo da escala usada e depois convertida em um **número digital** e esse em **binário** (formado por "uns" e "zeros") para ser armazenado.

Se a imagem for binária

- Os pixeis podem estar apenas ligados ou desligados:

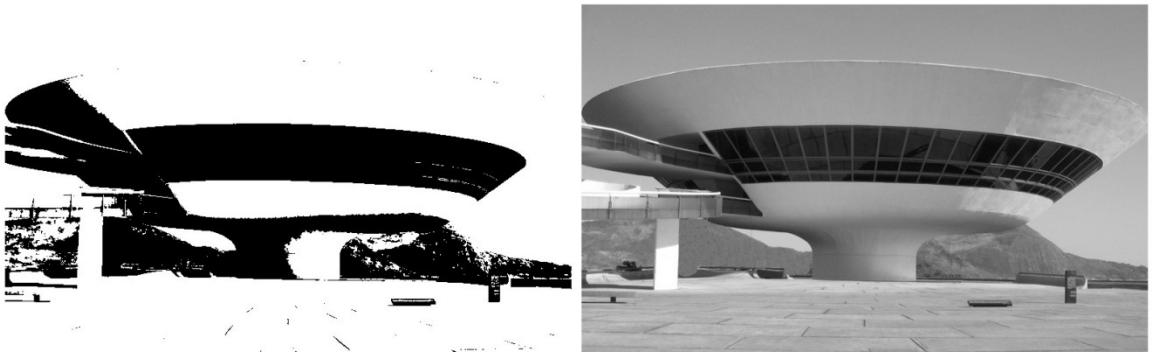
O tamanho N x M é chamado a resolução da imagem ou desenho



Mesmo objeto em duas resoluções, mas exibido no seu tamanho original.

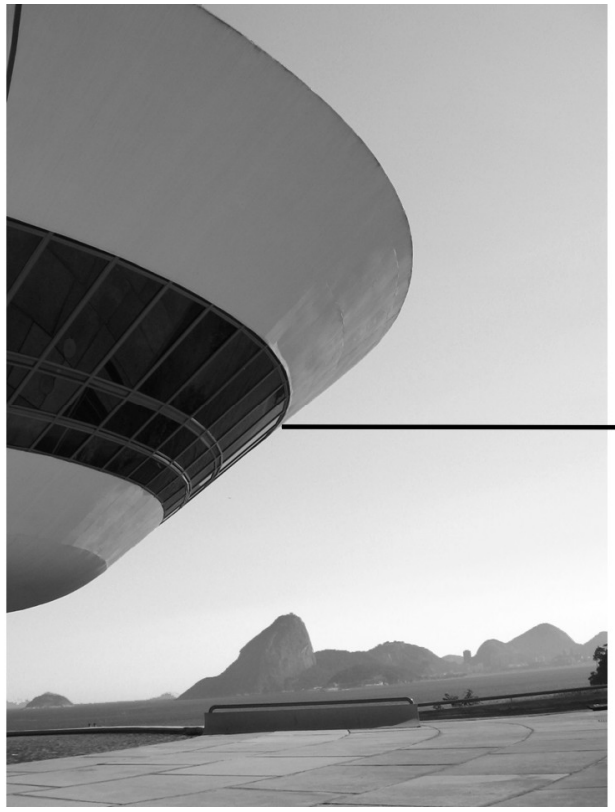
Imagens Monocromáticas

são as que não tem cores!



Exemplos de imagens monocromáticas

Um *pixel* naimagem nomocromática é caracterizado pelo valor do tom e pela sua localização na imagem.



47	52	64	132	153
51	58	121	149	142
49	99	143	144	164
94	135	161	170	199
138	165	180	212	213

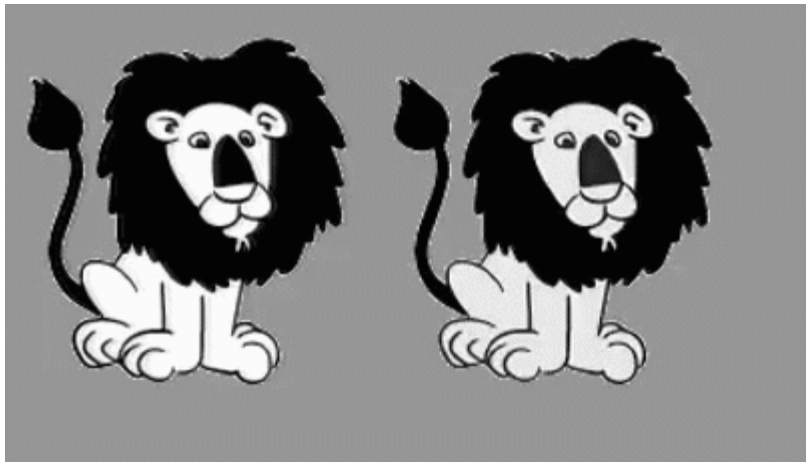
Representação matricial de uma região da imagem.

O número de tons entre os valores limites, branco e preto, que se pode representar em tons, depende de quantos bits são alocados na matriz de imagem para armazenar o tom de cada *pixel*.

Número de elementos na Escala de cinza	**Tons de cinza limites**	**Números de Bits necessários para representação do *pixels***
2^1 2 valores	0,1	1
2^3 8 valores	0 a 7	3
2^4 16 valores	0 a 15	4
2^8 256 valores	0 a 255	8

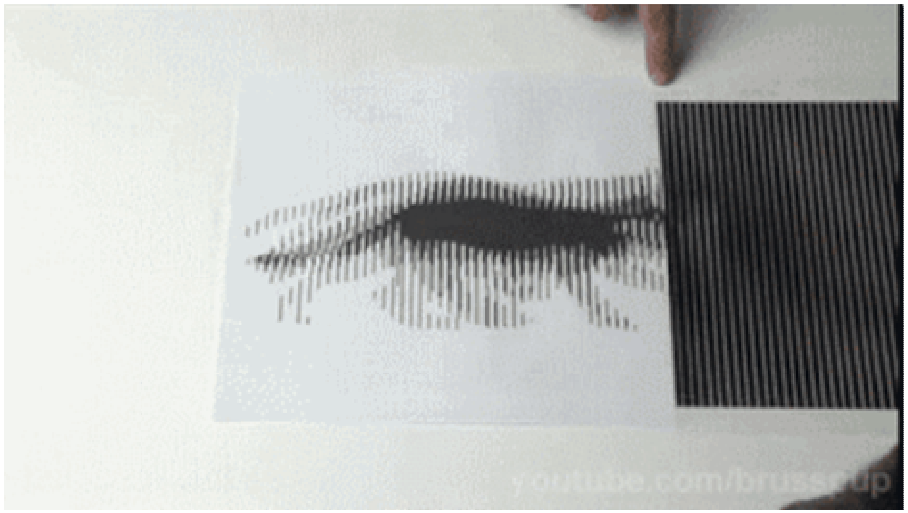
Uma animação

- É um conjunto de quadros parados que se movem!



O mesmo corre em um filme

- Só que ai se usam imagens reais capturadas e digitalizadas



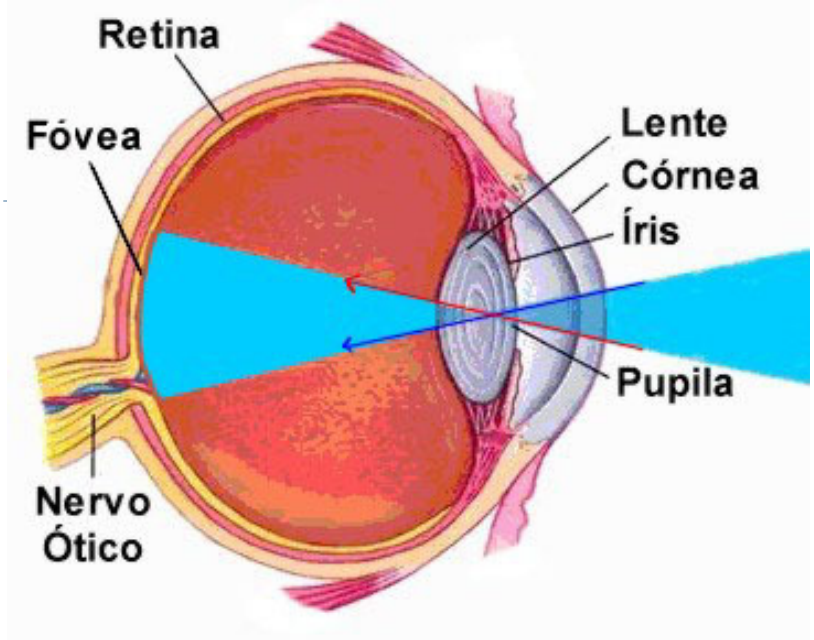
Já a cor é mais que isso

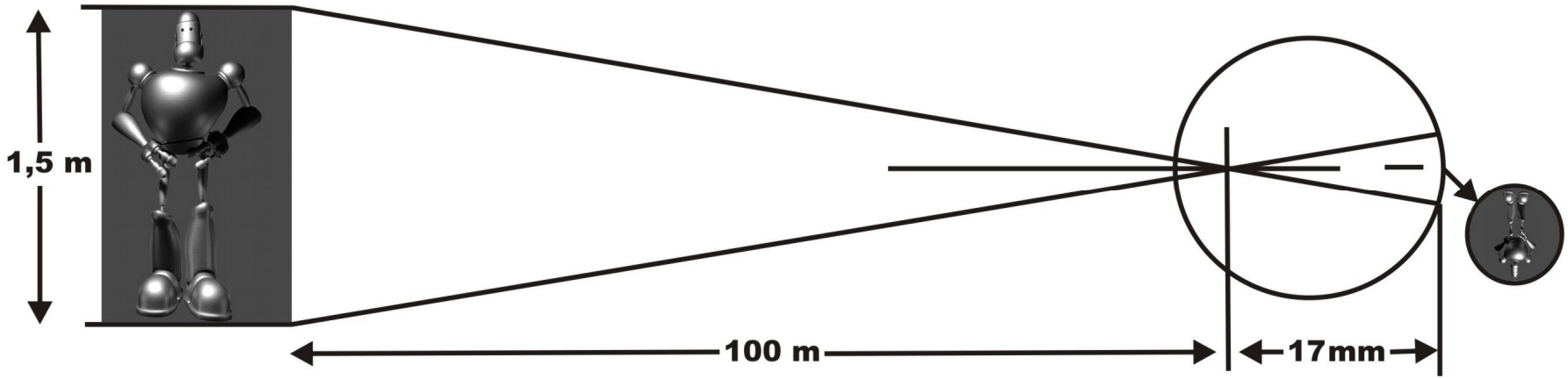
- A **cor** e a **iluminação** são analisadas por partes diferentes do celebro.
- Estas partes estão **fisicamente separadas** e são anatomicamente tão distintas quanto são a visão e a audição.

Imagem colorida

- A capacidade de **interpretar formas** tridimensionais e a **organização espacial** independem da cor, mas sim da iluminação das formas que representam.

Sistema de Visão Humana

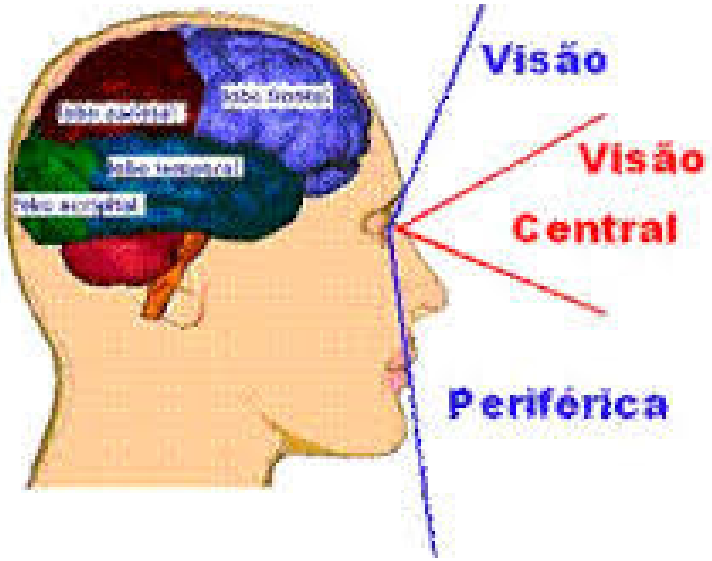




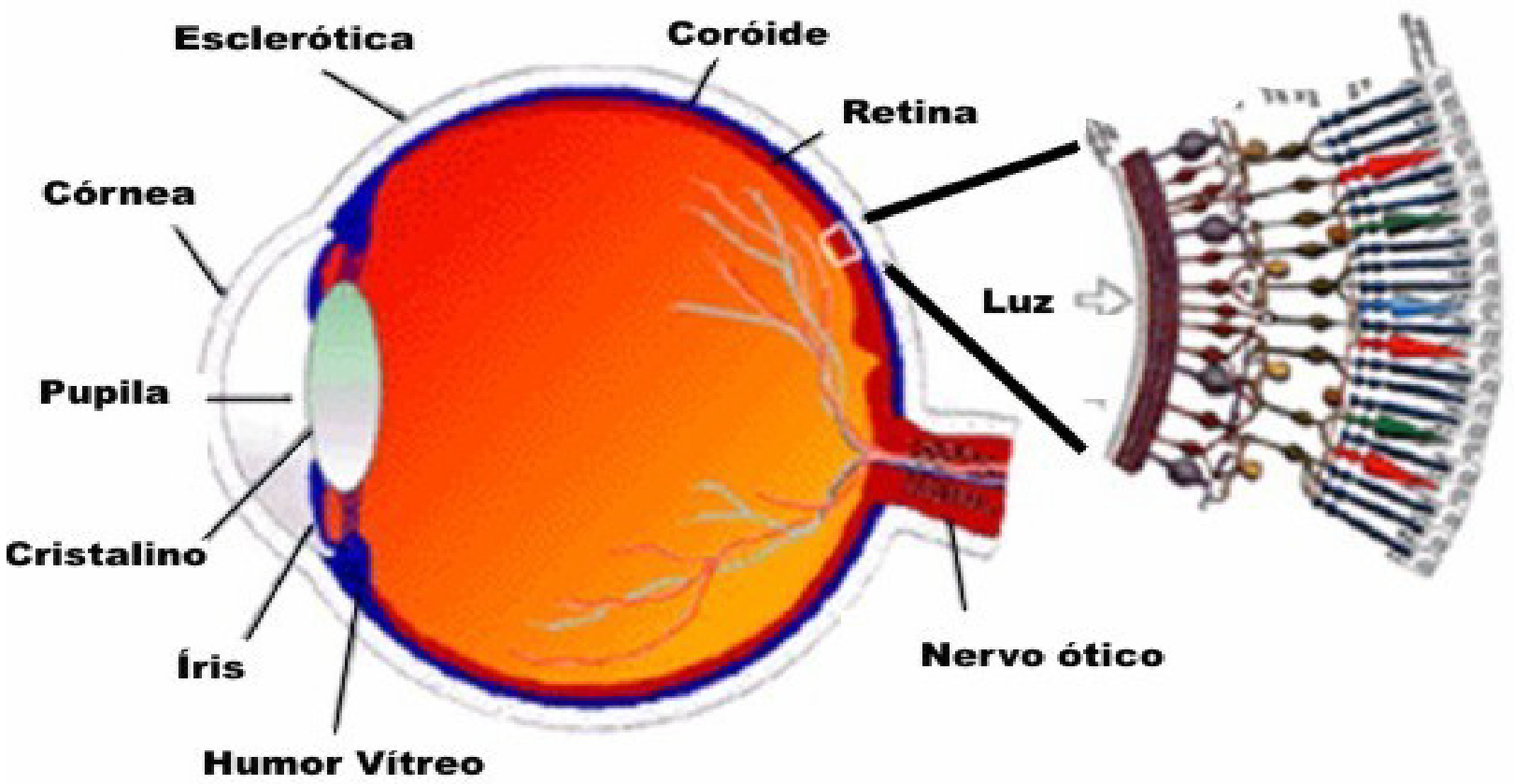
Relações de tamanho

Cores -> visão central

- Intensidade - > Visão periférica

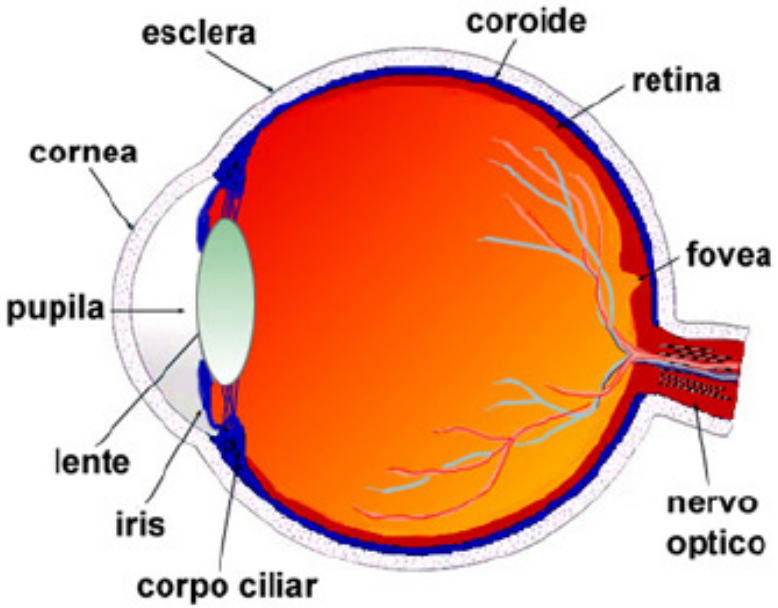


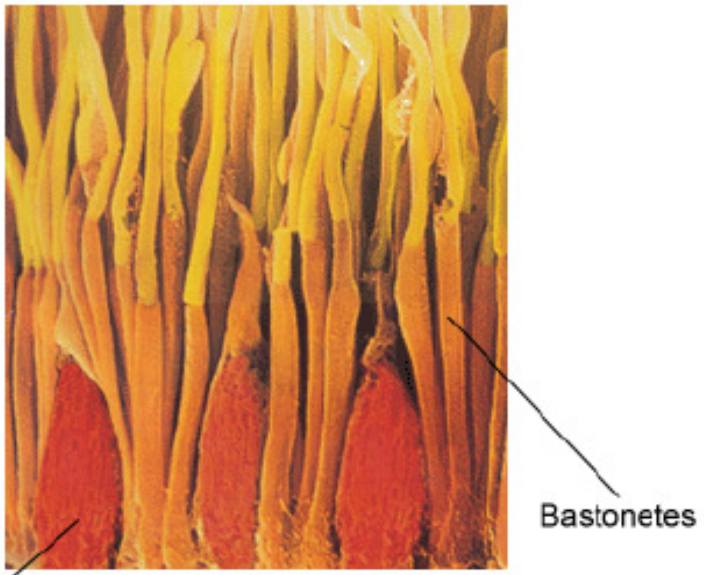
Células Cones e Bastonetes



Olho humano e células da retina

Os cones são cerca de 7 milhões.





cerca de 125 milhões

Bastonetes

- Visão monocromáica:

A substância química responsável pela sensibilidade dos bastonetes à luz é a **rodopsina**, quando a luz incide sobre uma molécula de rodopsina, esta gera um sinal elétrico que é transmitido às células nervosas presentes na retina.

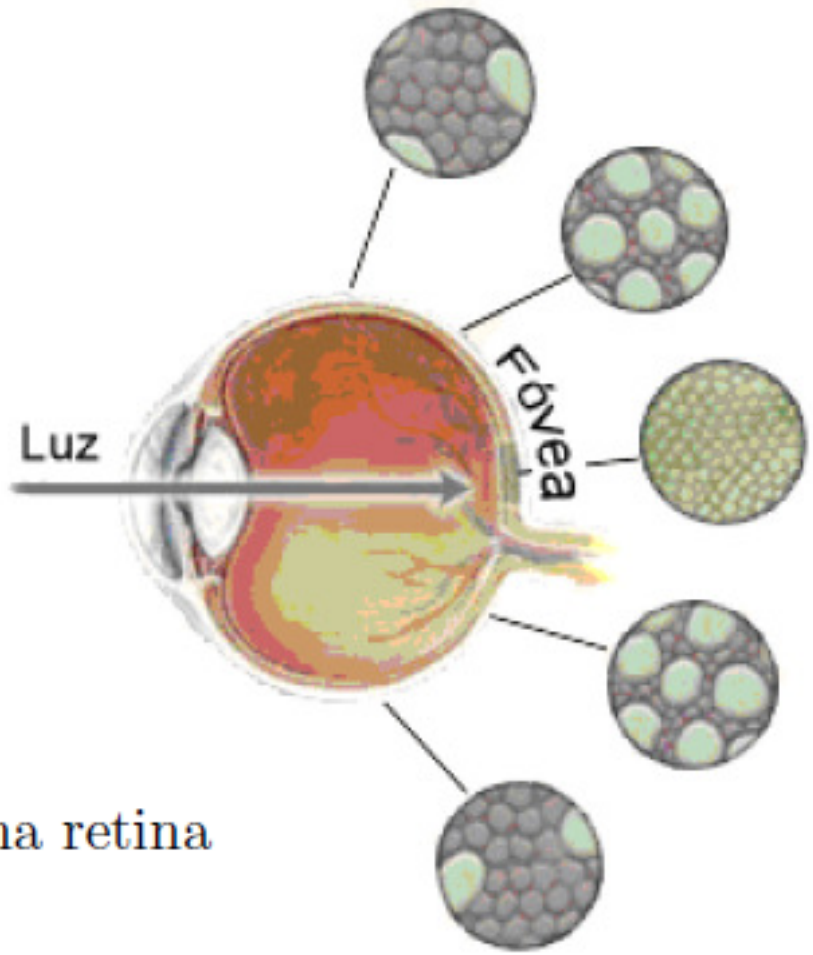
cones

- 3 tipos:

i. L-Cones - Curva de resposta com pico em 445nm

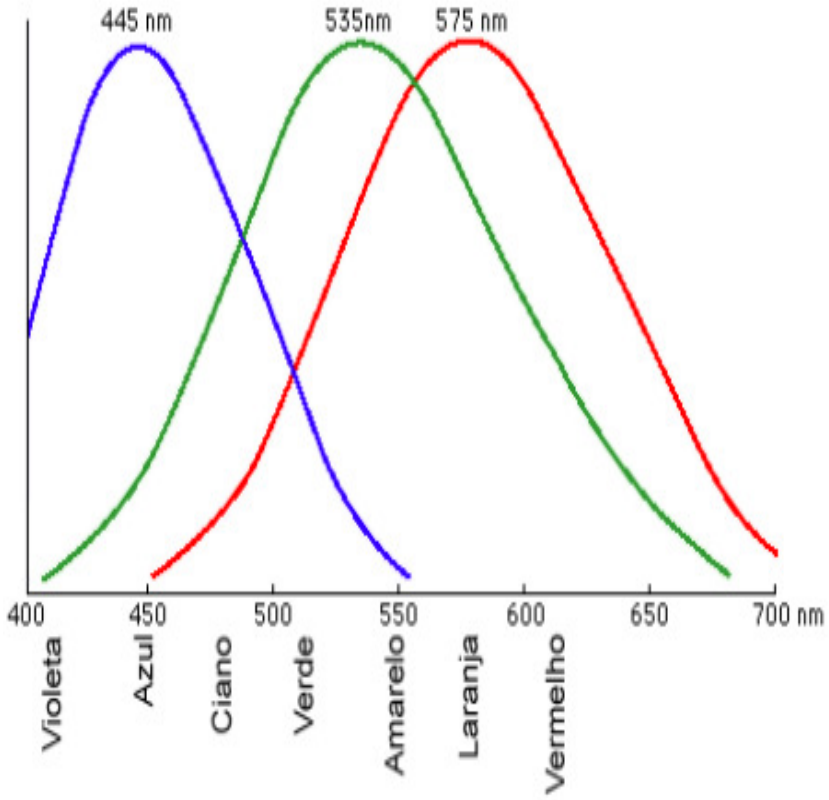
ii. M-Cones - Curva de resposta com pico em 535nm

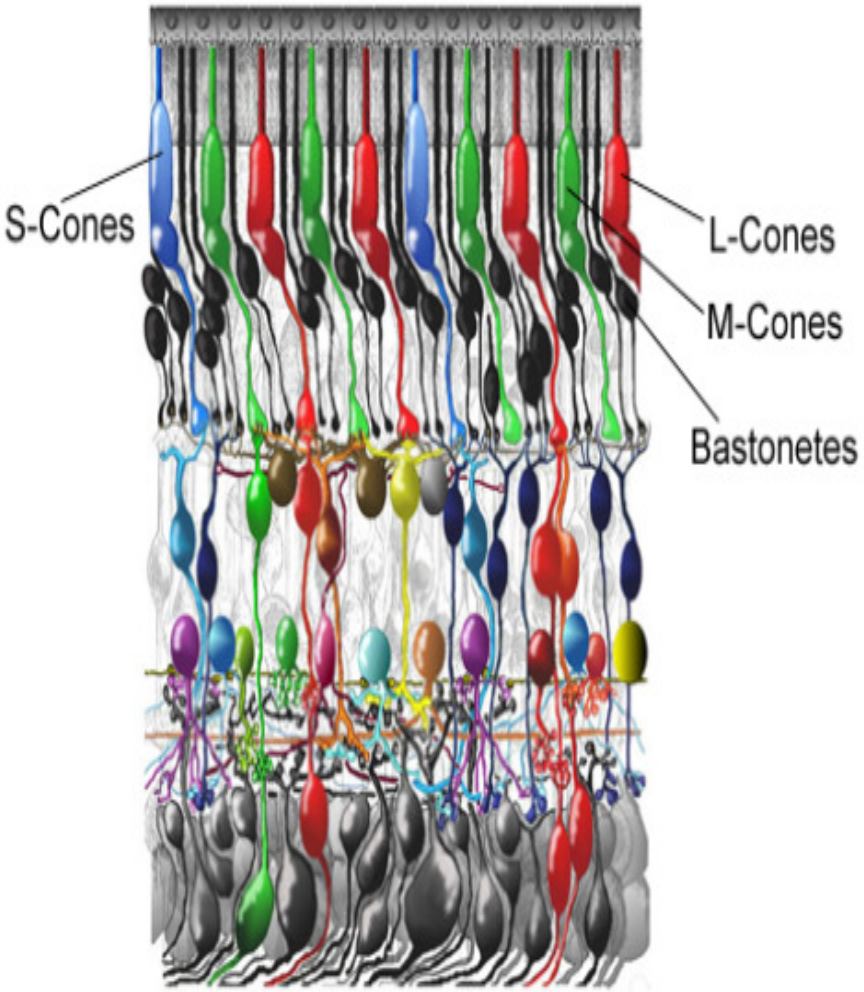
iii. S-Cones - Curva de resposta com pico em 575nm



Distribuição dos cones e bastonetes na retina

Curvas de respostas dos 3 tipos de cones





Teoria Tricromática

Apenas três tipos de receptores da retina são necessários operando com sensibilidades a **diferentes comprimentos de onda**: **três cores primárias**.

Os três cones existentes na retina são sensíveis respectivamente ao vermelho (*R*), ao verde (*G*) e ao azul (*B*), chamadas *cores primárias de luz*.

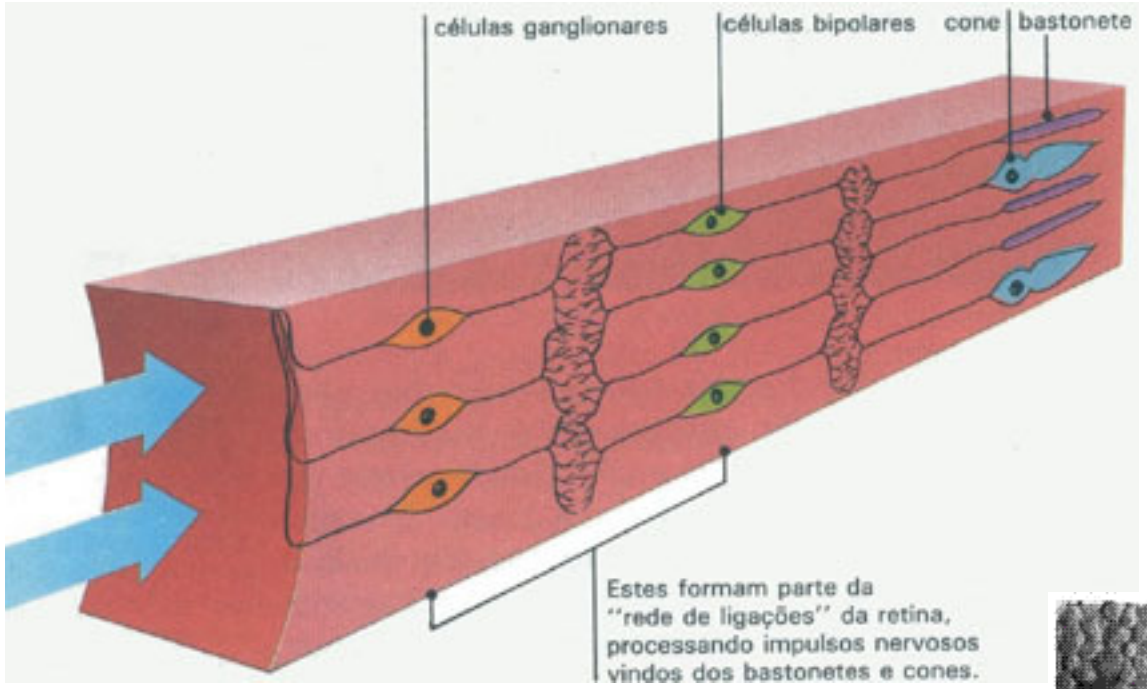
Teoria de Yong

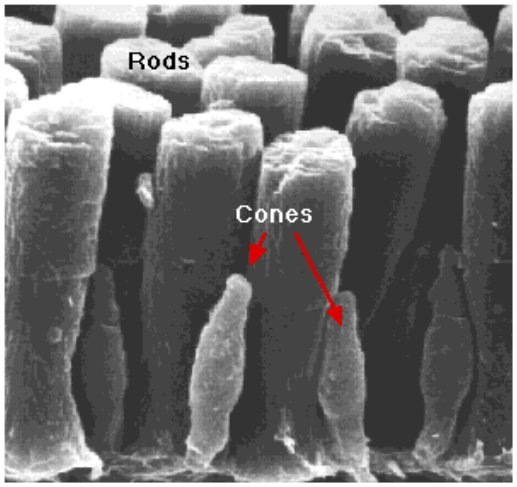
- Young, no século XIX, mostrou experimentalmente que a retina tem **3 tipos distintos de foto pigmentos**, sensíveis `as 3 cores primarias: **vermelho**, **verde** e o **azul**.

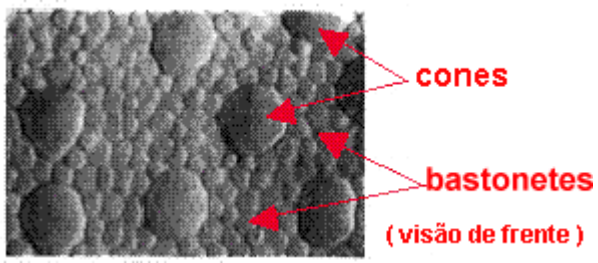
- Ele concluiu ainda que esta decomposição da luz em 3 cores não é uma característica da luz , mas sim **uma característica do sistema visual humano**

Sistema de Visão Humana

Esquema x real

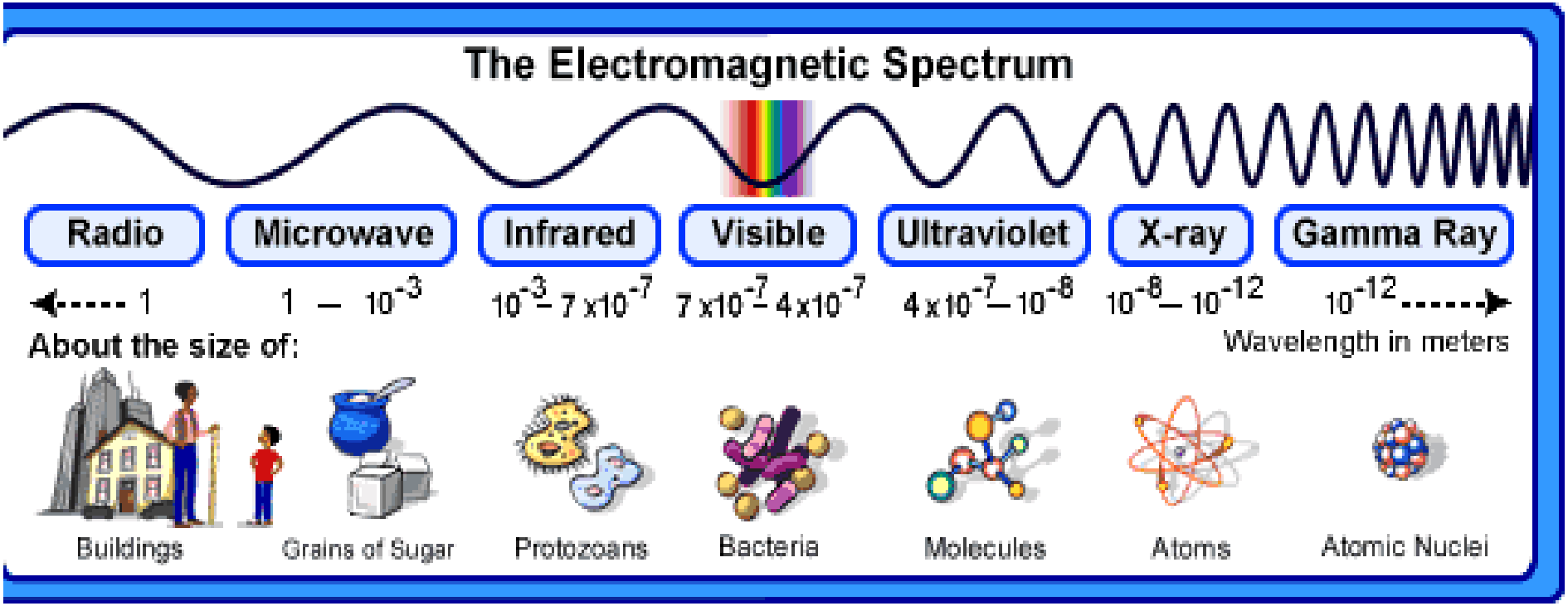






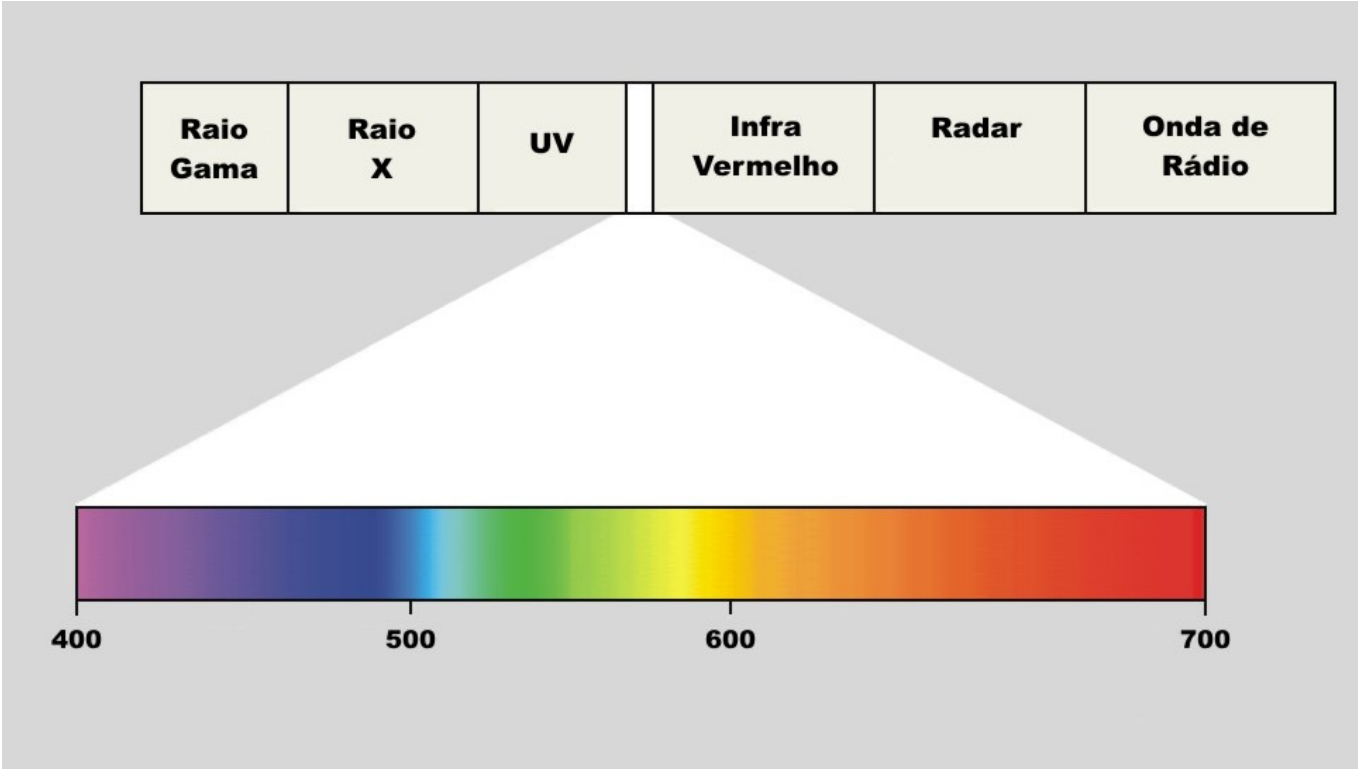
Espectro eletromagnético

- E comprimentos de onda



Características ópticas da luz

Radiação Eletromagnética



Espectro eletromagnético e comprimentos de onda

(em nano metros – nm) .

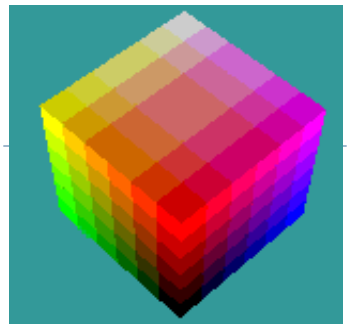
O espaço de cor *RGB uma cor é representada pela combinação de e cores*

- $C = r\,\mathbf{R} + g\,\mathbf{G} + b\,\mathbf{B}$

onde $\mathbf{R}$, $\mathbf{G}$ e $\mathbf{B}$ são as cores primarias e *r*, *g* e *b* os coeficientes da mistura

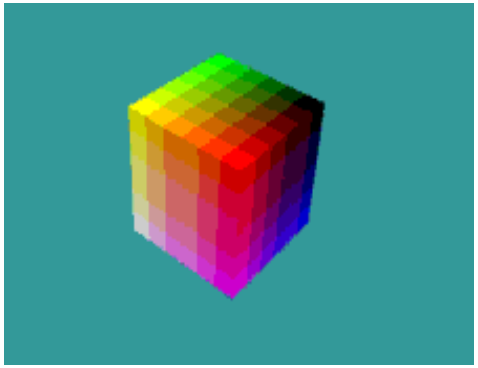
Em geral define-se em três como o número de cores primarias em um espaço, devido ao fato do olho humano possuírem **três tipos de fotorreceptores**.

Cores aditiva obtidas pela combinação de luzes RGB

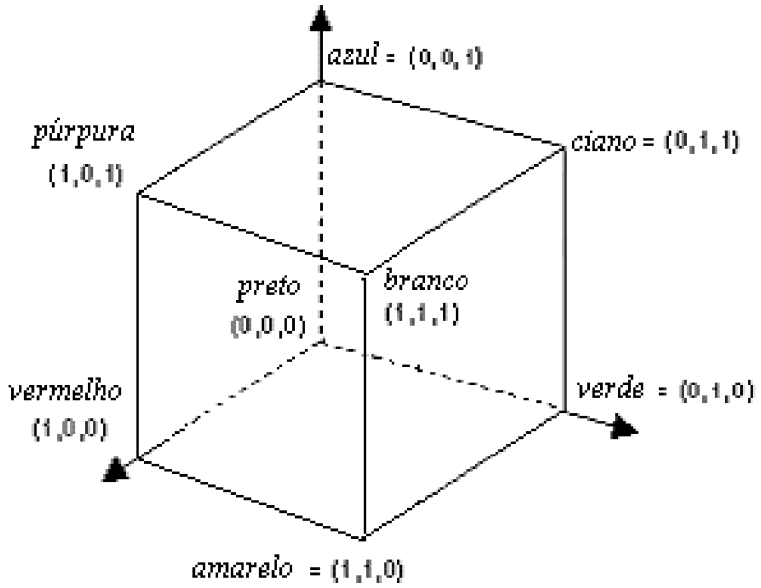


Modelos de cor

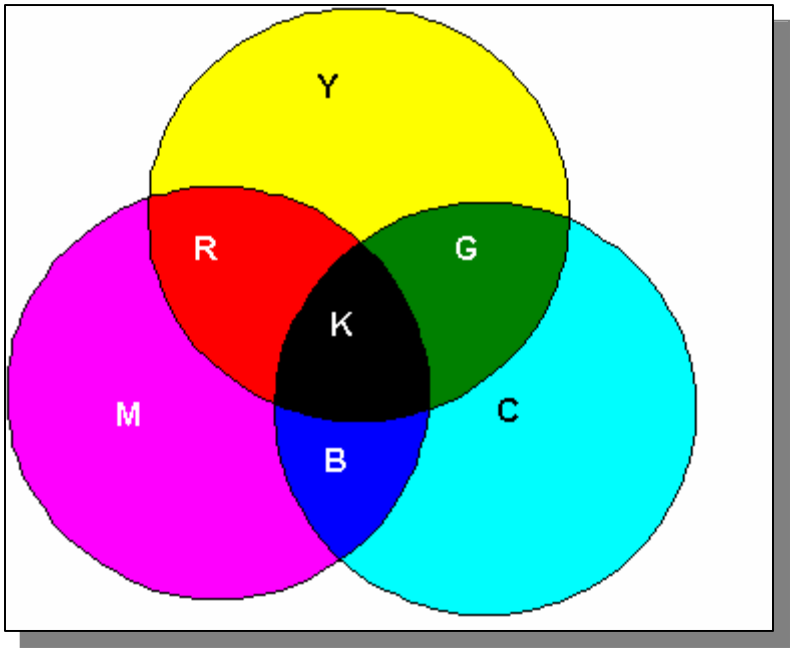
Sistema RGB



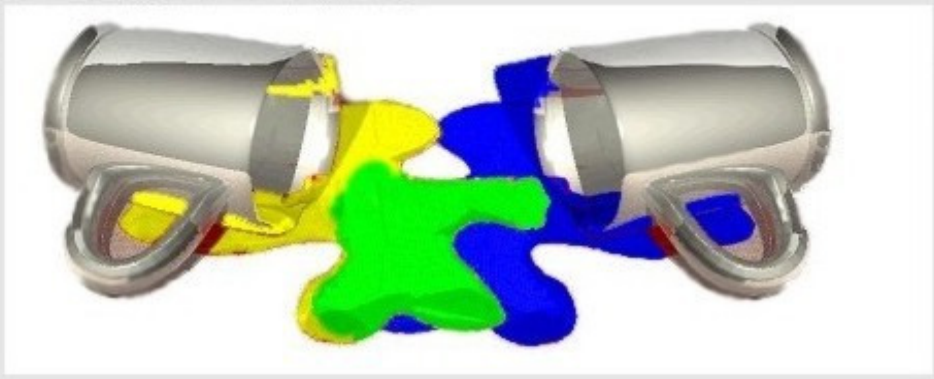
- Normalizado entre 0 e 1



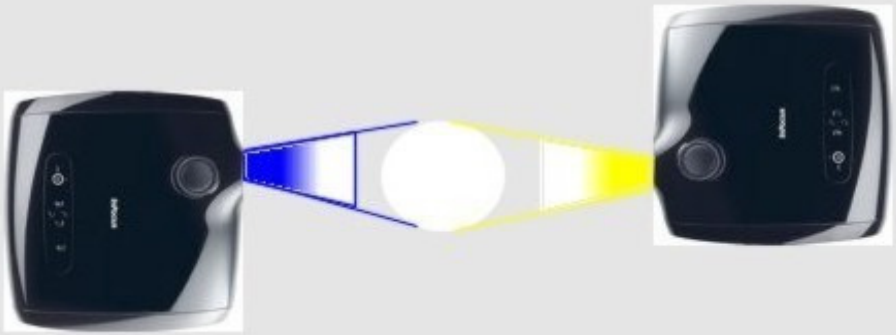
Sistemas de cores subtrativos
CMY



Modelos de cor: noção de primárias, secundárias e terciárias



Tinta amarela misturada com tinta azul cria uma tinta verde.



Luz amarela com luz azul cria uma luz branca.

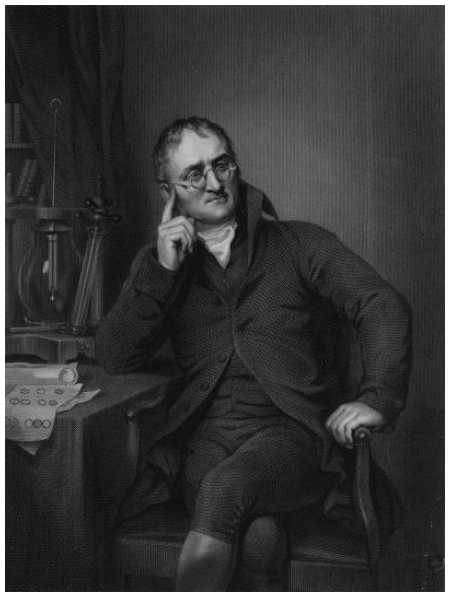
Cores complementares

Os pigmentos se combinam, subtraindo intensidades luminosas da luz que atinge os objetos.

Percepção de Cor

Daltonismo.

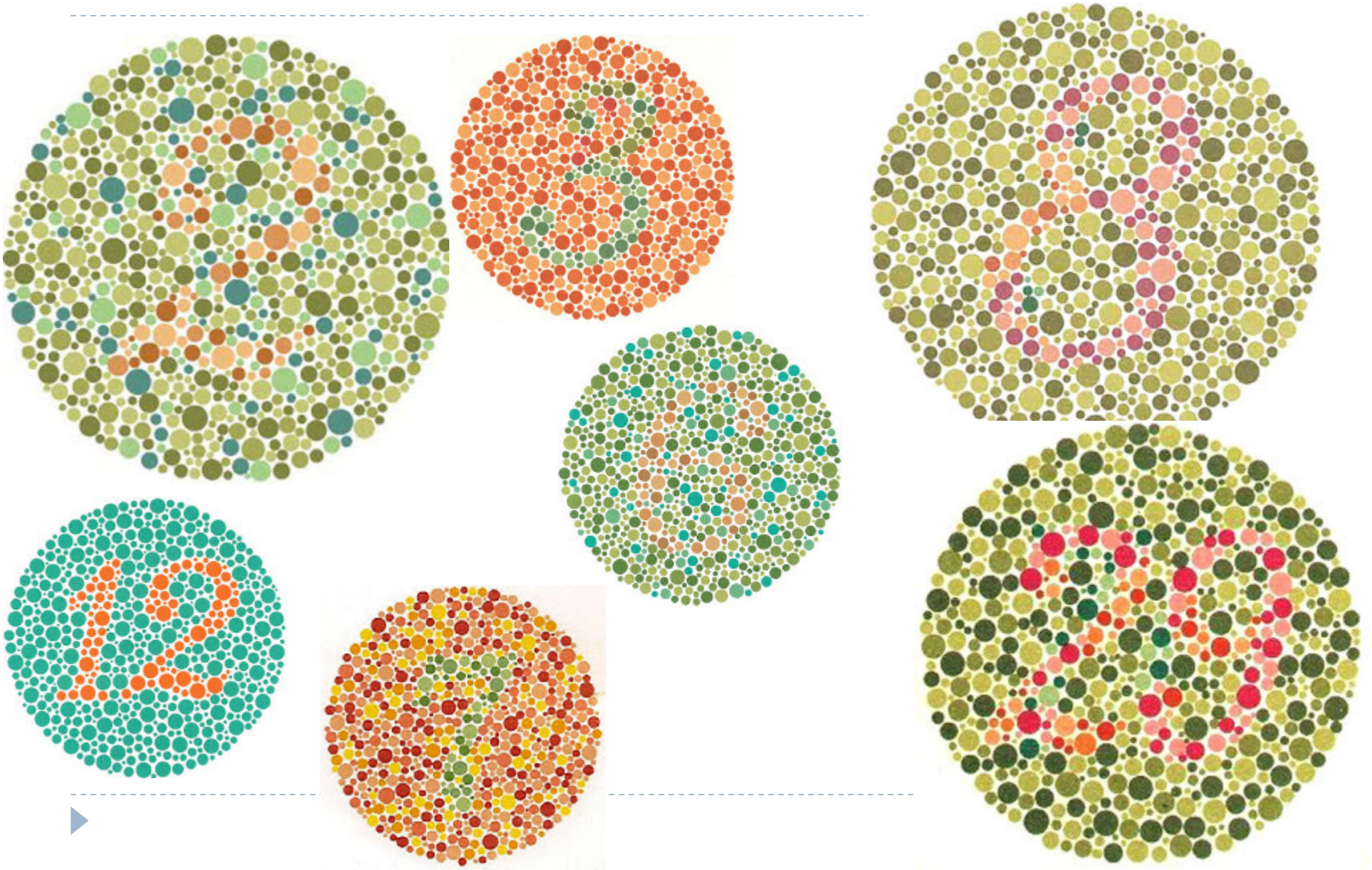
- O primeiro tratado científico sobre a deficiência na visão de cores foi publicado em 1798 pelo químico Inglês **John Dalton [1766-1844] por isso todos os problemas de visão a cores são também chamados de Daltonismo.**



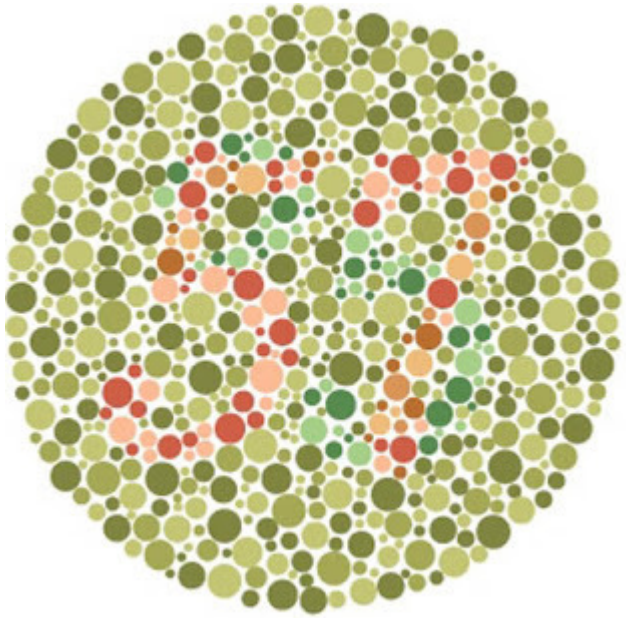
Teste de Daltonismo

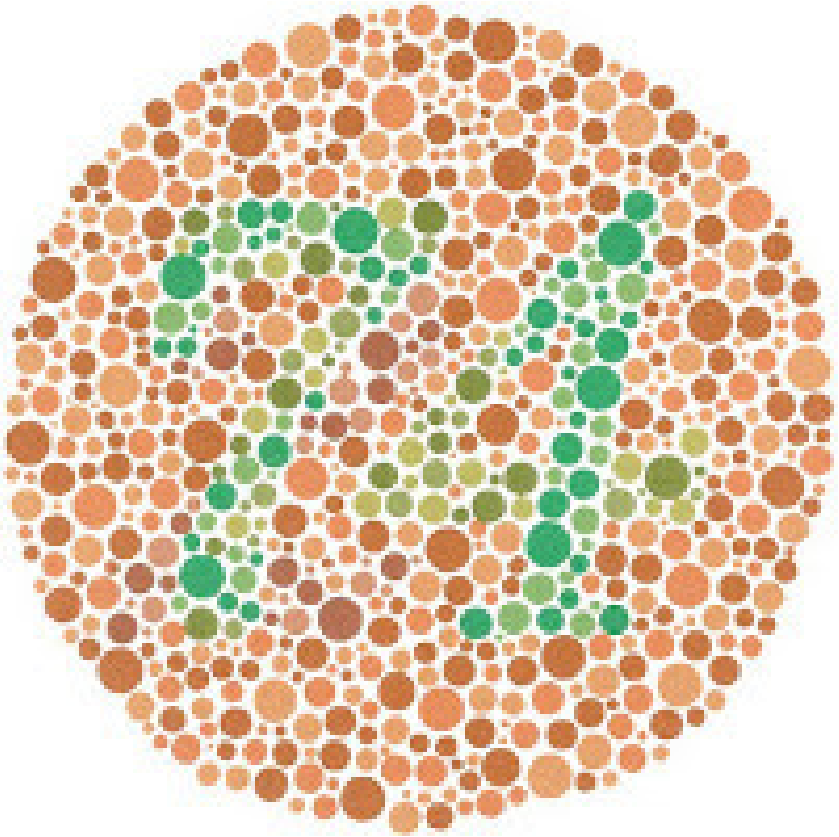
- Na maioria das vezes o daltônico leva anos para perceber sua deficiência: Como sentir falta de algo que nunca se viu?
- Devido a fatores genéticos ligados ao cromosoma X, as mulheres têm muito menos probabilidade de serem daltônicas do que os homens.
- Teste resumido de daltonismo utilizando figuras
- O objetivo deste teste é identificar os números presentes em cada figura.

pessoas com daltonismo não enxergam os números
2 ,12, 3,6, 7,8 e 29



pessoas com daltonismo não enxergam os números 57, e 74

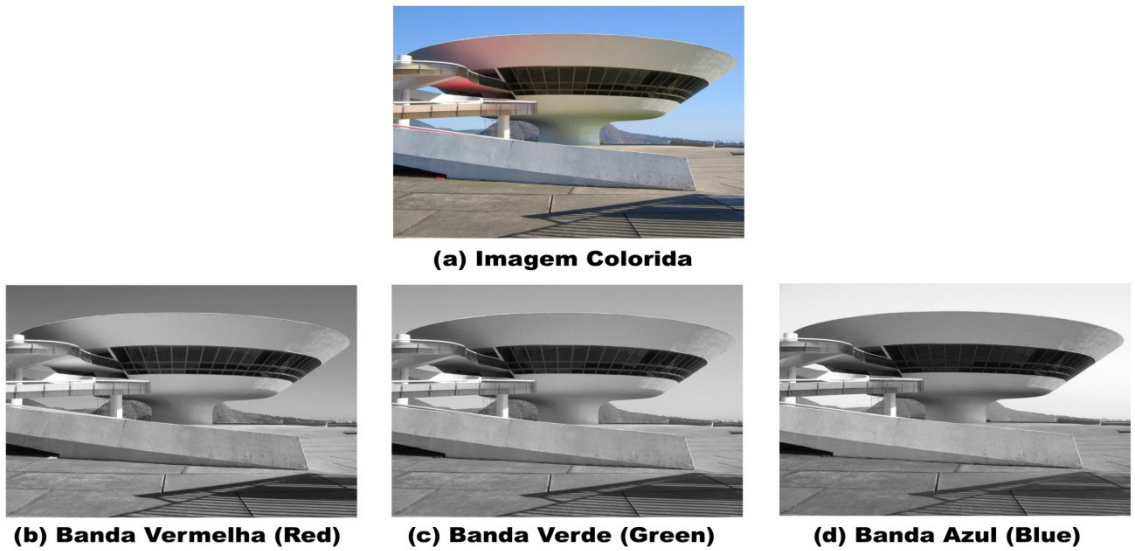




Imagens Coloridas

Imagens multibandas são imagens digitais onde cada *pixel* possui *n* bandas espectrais.

Quando uma imagem é representada pela composição das três bandas visíveis (RGB) tem-se uma imagem colorida aos olhos humanos.



(a) Imagem Colorida

(b) Banda Vermelha (Red) (c) Banda Verde (Green) (d) Banda Azul (Blue)

Imagem colorida e cada uma de suas bandas RGB.

Exemplo ao inved de 0 e 1 as cores podem ser descritas de o a 255 por 1 byte:

COR	RGB
	R = 255, G = 0, B = 0
	R = 0; G = 255; B = 0
	R = 0; G = 0; B = 255
	R = 255, G = 255, B = 0
	R = 255, G = 0, B = 255
	R = 255, G = 100, B = 0
	R = 100, G = 100, B = 100
	R = 100, G = 0, B = 100
	R = 0, G = 60, B = 0
	R = 0, G = 50, B = 255

Cores 1 byte por canal RGB:

RGB	Cor
122 139 139	
52 245 255	
71 60 139	
255 0 255	
218 112 214	
255 140 0	

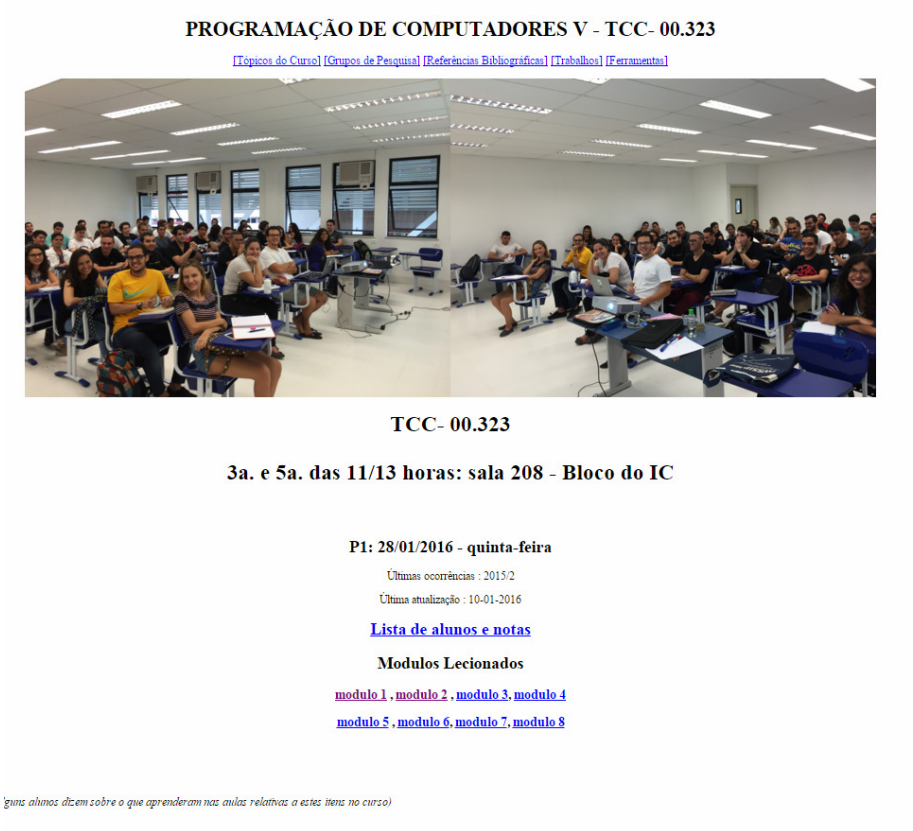
Cores criadas com o vetor cromático R,G,B em percentagem!

Representação como pontos de um espaço 3D de Cor

Cor	R (%)	G (%)	B (%)	
vermelho puro	100	0	0	
azul puro	0	0	100	
amarelo	100	100	0	
laranja	100	50	0	
verde musgo	0	25	0	
salmão	100	50	50	
cinza	50	50	50	

Ainda pode aparcer em hexadecimal!

- É isso que voc vê nos códigos de cores, por exemplo em uma pagina da internet!



PROGRAMAÇÃO DE COMPUTADORES V - TCC- 00.323

[Tópicos do Curso] [Grupos de Pesquisa] [Referências Bibliográficas] [Trabalhos] [Ferramentas]

TCC- 00.323

3a. e 5a. das 11/13 horas: sala 208 - Bloco do IC

P1: 28/01/2016 - quinta-feira

Ultimas ocorrências : 2015/2

Última atualização : 10-01-2016

Lista de alunos e notas

Modulos Lecionados

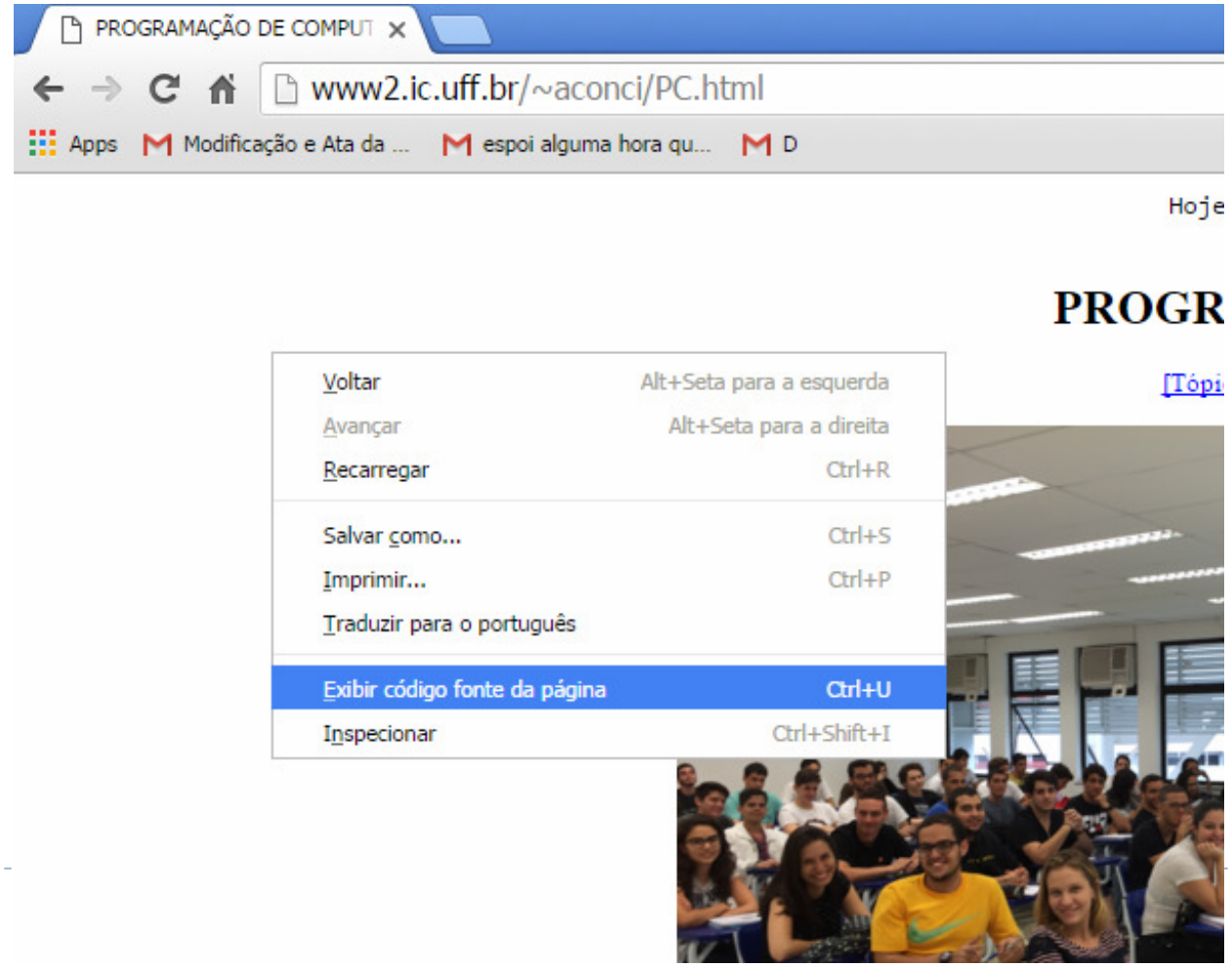
modulo 1 , modulo 2 , modulo 3, modulo 4

modulo 5 , modulo 6, modulo 7, modulo 8

(guns alunos dizem sobre o que aprenderam nas aulas relativas a estes itens no curso)

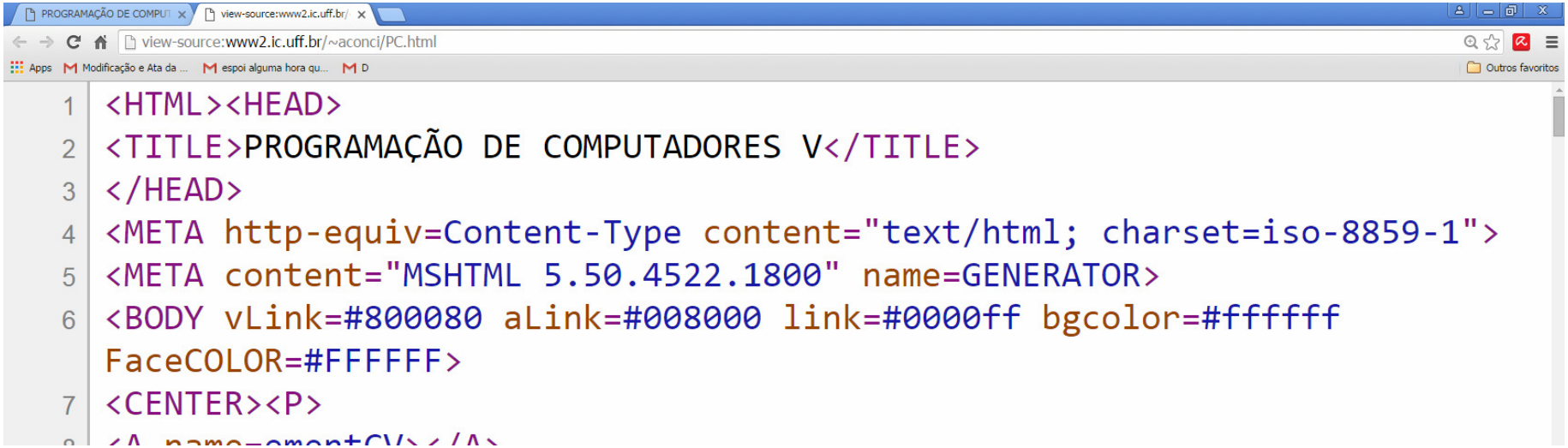
Como ver

- Botão direito no mouse:



Possibilita ver o código HTML

- Que gerou a pagina e nele as cores!



Como vemos filmes e animações?

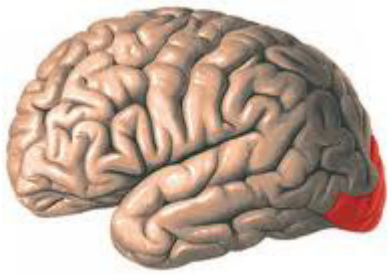
- **Persistência visual**
- Quando vista em uma velocidade adequda não se percebe os quadros

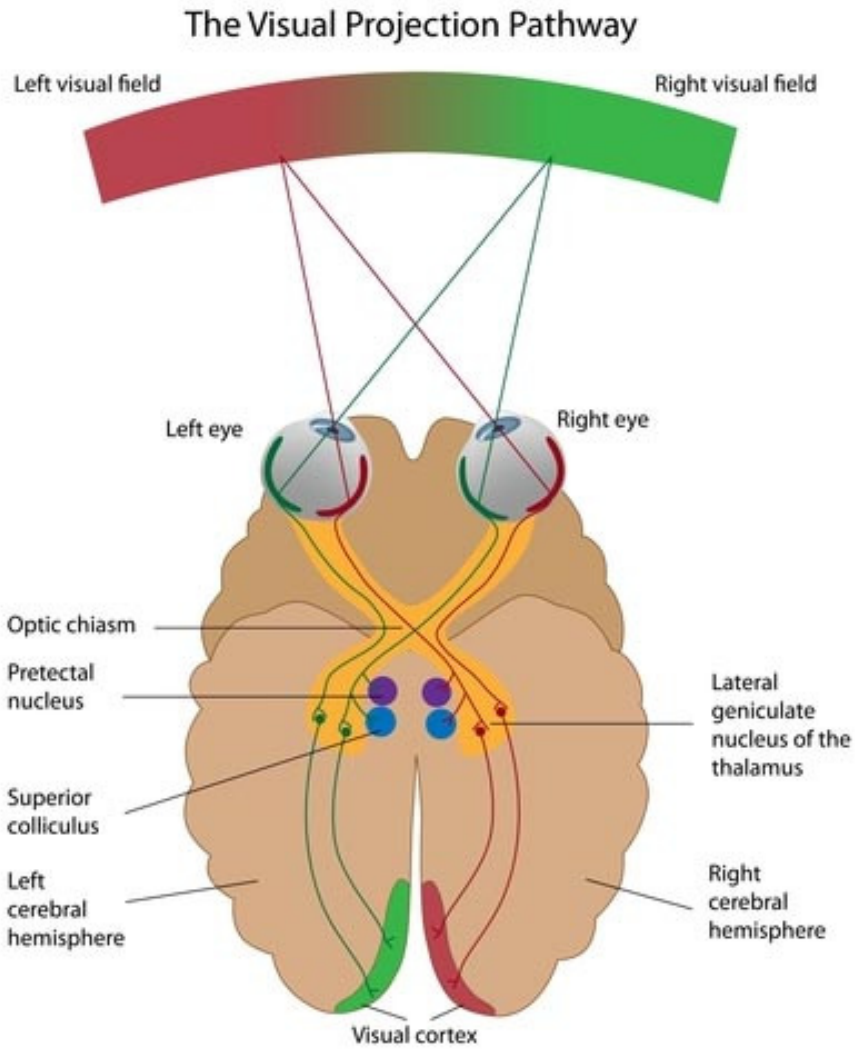
Como se dá a visão 3D ?

- **Estereoscopia** é uma técnica usada para se obter informações do 3D , através da análise de duas imagens obtidas em pontos diferentes.
- É um fenómeno natural que ocorre em muitos animais com dois pontos de visão e também no ser humano, quando uma pessoa observa em seu redor uma cena qualquer.
- O fato de o ser humano ter 2 olhos permite-lhe, através da estereoscopia ter a noção de profundidade espacial, com o objetivo de por exemplo ter a noção da distância a que se encontram os objetos.
- A estereoscopia humana é união de duas imagens da cena que obtidas de pontos de observação ligeiramente diferentes sendo que o celebro as funde e nesse processo, o indivíduo obtém informações quanto à profundidade, distância, posição e tamanho dos objetos, gerando uma sensação de 3D.

Núcleos Laterais Geniculares (LGN)
Campos visuais do lado direito de cada olho é tratado no LGN do lado esquerdo

córtex visual





referencias

- https://www.youtube.com/watch?v=ixk5RIqABjI
- http://www.inf.puc-rio.br/~inf1005/material/slides/funcoes.pdf
- http://www.inf.puc-rio.br/~inf1005/material/slides/intro_c.pdf
- E. Azevedo, A. Conci, Computação Gráfica: teoria e prática, Campus ; 2003