

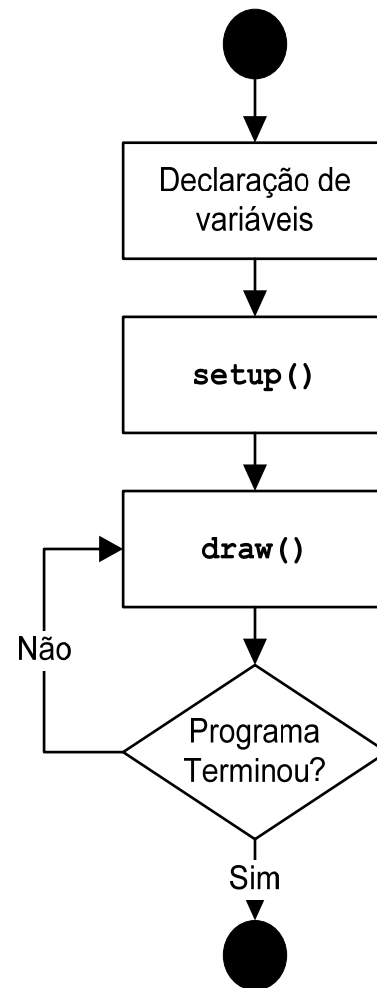
Programação Multimédia

Variáveis

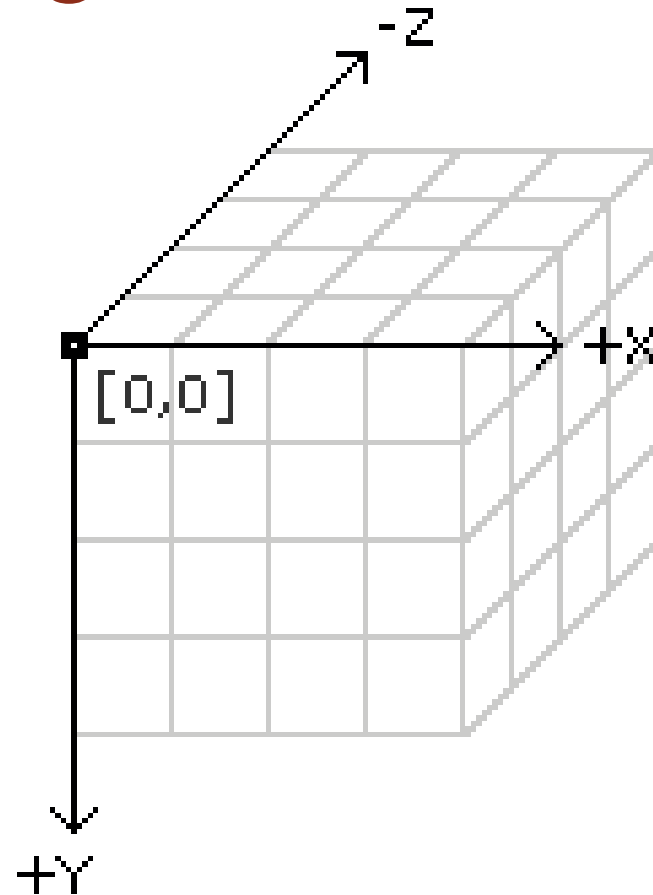
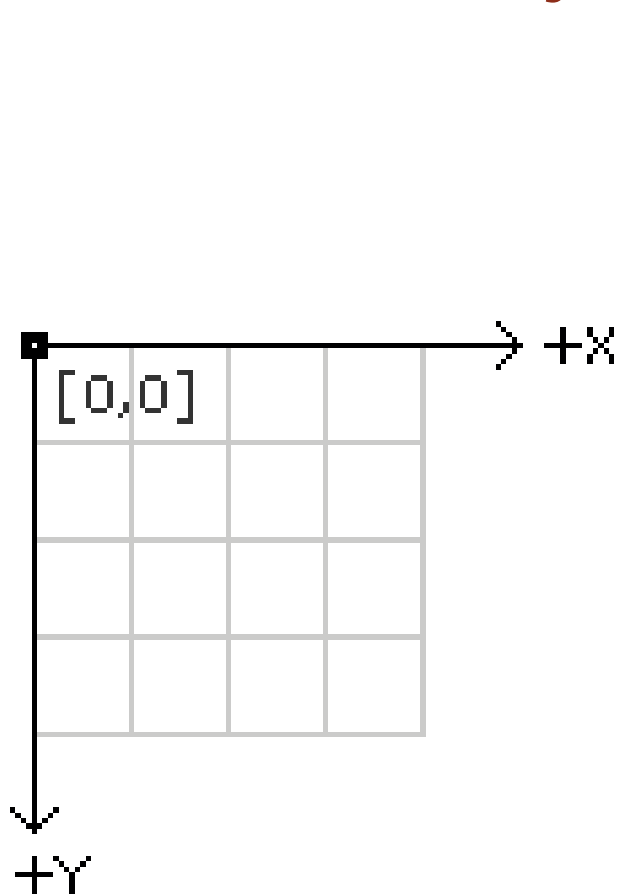
Conceitos Básicos de Um Programa

- **Memória (armazenamento de dados)**
 - Guardar dados temporários (durante a execução do programa)
- **Seleccção (...de caminhos de execução)**
 - O nosso programa pode ter ramos que são executados em determinadas circunstâncias
- **Iteracção (execução repetida das instruções)**
 - Executar várias vezes a mesma acção (sobre dados diferentes)
- **Módulos**

Diagrama de fluxo do Processing



Eixos da janela gráfica



Exemplo #1

- Desenhar um rectângulo

```
// inicializacao do programa
void setup() {
  // tamanho da janela: 400x400
  size(400, 400);

  // numero de frames desenhadas por segundo
  frameRate(10);
}

// desenho das varias frames
void draw() {
  rect(10, 10, 40, 40);
}
```

Exemplo #1

- O exemplo anterior não usa memória!

Exemplo #2

```
int rectX;
int rectY;

// inicializacao do programa
void setup() {
  size(400, 400);
  frameRate(10);

  rectX = 10;
  rectY = 10;
}

// desenho das varias frames
void draw() {
  rect(rectX, rectY, 40, 40);

  rectX = rectX + 1;
}
```

- Declaração de variáveis
- Comentário
- Inicialização do programa
- Desenho do conteúdo da frame actual

Exemplo #2 – Evolução das variáveis

Frame	rectX	rectY
0 (inicialização)	10	10
1	11	10
2	12	10
3	13	10
4	14	10
5	15	10
...

- Resultado
 - Rectângulo “move-se” para a direita

Memória do Programa

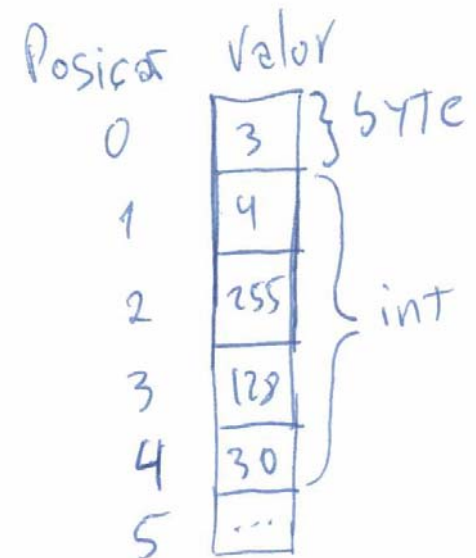
- Permite-nos armazenar dados
 - Usados para operações intermédias do programa
 - Quando o programa termina, estes dados desaparecem
- Os dados são guardados na memória RAM do computador

Exercício #1

- Alterar o último exemplo de forma a que o rectângulo se mova na diagonal (horizontal + vertical)

Memória

- A memória RAM é vista pelo processador como uma tabela
- Cada entrada da tabela pode armazenar um byte
- Cada entrada é referenciada pela sua posição (endereço)
- Um byte é um conjunto de 8 bits
 - Pode representar um número até 255 (em decimal)
- Cada entrada pode ser vista de forma isolada (byte) ou agrupada com outras posições



Variáveis

- As variáveis representam posições de memória que podem guardar um valor
- As variáveis têm um nome que utilizamos no programa
- Não precisamos de decorar a posição de memória!

Memória

- É importante para o programador poder utilizar números maiores do que 255!!!
- As posições de memória podem ser agrupadas em conjuntos de bytes
- Os bytes podem ser estruturados de formas diferentes
 - valores inteiros, decimais, texto

Tipos de Dados

- Cada variável pode guardar valores de um determinado tipo:
 - Inteiros (`int`, `long`)
 - Decimais (`float`, `double`)
 - Lógicos (`boolean`)
 - Caracteres (`char`)
- Tipos simples

Variáveis na Prática

- Num programa as posições de memória são representadas por nomes que o programador atribui
- Se precisamos de guardar uma idade, damos o nome “idade” à posição de memória
- O compilador “traduz” os nomes em posições de memória que o processador entende
- Para além do nome temos de indicar o tipo de variável
 - tipos diferentes têm tamanhos diferentes logo usam mais ou menos bytes contíguos na memória
 - o compilador tem de saber para poder alocar correctamente as posições de memória

Declaração de Variáveis (processing)

```
[tipo] nomeDaVariavel;
```

- Isto declara a variável apenas
- O Programa fica “a saber” que vamos utilizar uma posição de memória para guardar dados de um determinado tipo
- Exemplo:

```
int idade;
```
- Nota:
 - O ponto e vírgula “;” é importante!!!
 - É usado para terminar **todas** as instruções em Processing
 - Não o colocar é um erro sintáctico

Atribuição

- Quando a variável é criada, não tem nenhum valor guardado!
- Para guardarmos valores numa variável temos de *atribuir* esses valores:
`idade = 31;`
- A atribuição é feita com o sinal “igual”
 - Colocar na variável “idade” o valor 31
 - A partir daqui quando usamos a variável “idade”, estamos, de facto, a usar o valor 31
- Em qualquer altura do programa podemos atribuir um valor a uma variável

Inicialização

- Antes de ler o valor de uma variável é preciso colocar algo lá dentro
 - Inicializar a variável

Exercício #2

- Alterar o Exercício #1, de forma a que o rectângulo se mova menos de 1 pixel em cada frame
 - por exemplo meio pixel por frame.

Expressões

- O que se atribui a uma variável é um valor resultante de uma **expressão**
- Uma expressão pode ser:
 - Literal : Um valor escrito directamente no código
 - idade = 33; // 33 é um literal
 - Variável : O valor de uma variável
 - idade = idadeJoao; // idadeJoao é outra variável
 - Função : O valor devolvido por uma função definida previamente
 - idade = random(10); // random é uma função do processing
 - Expressões com operadores aritméticos : Definição recursiva!
 - idade = idadeJoao + 10;

Operadores Aritméticos

- Adição: “+”
- Subtracção: “-”
- **Divisão: “/”**
- Multiplicação: “*”
- Resto da Divisão Inteira: “%”

Exercício #3

- Modificar o Exercício #2, de forma a que a trajectória do rectângulo siga a linha definida pela equação: $y = 0.5 * x + 4$

Exercício #4

- Modificar o Exercício #3, de forma a que a largura do rectângulo varie proporcionalmente com a posição X.

Expressões: Atribuição com funções

- Nota:
 - Funções são conjuntos de instruções com nome que, nalguns casos, representam um valor
- Podemos também usar algumas funções nas atribuições:

```
float a;
```

```
a = 3 * random(100);
```

- `random()`
 - Procurar na referência

Exercício #5

- Modificar o Exercício #2, de forma a que o rectângulo seja desenhado numa posição aleatória em cada frame.

Projecto Semanal

1. Criar um programa que desenhe dois círculos de forma a que a trajectória de um seja
 1. $y = 1 * x + 20$;
2. e a do outro
 1. $y = 2 * x - 30$;
3. Os círculos devem ter cores diferentes