# Linguagens Formais e Autómatos / Compiladores

# Linguagem *geometrics*

**Departamento de Electrónica, Telecomunicações e Informática**
**Universidade de Aveiro**

**abril de 2021**

---

## Objetivos

O objetivo geral deste trabalho é a criação de um ambiente de programação que permita a criação de programas numa linguagem de programação genérica (Java, C++, Python, ...) que, quando executados, produzam uma animação de figuras geométricas 2D.

Este objetivo geral pode ser decomposto em 4 componentes:

- Definição da linguagem principal 'geometrics' que permita desenhar animações de figuras 2D.
- Construção de um compilador que permite transformar uma descrição nessa linguagem num programa na linguagem genérica escolhida.
- Definição de uma linguagem secundária que permita definir um banco de fuguras 2D.
- Dotação da linguagem principal de meios para incorporar figuras extraídas de bancos de figuras.

## Características da solução

Apresentam-se a seguir um conjunto de características que a solução desenvolvida pode ou deve contemplar. Essas características estão classificadas a 3 níveis:

- obrigatória – característica que a solução tem obrigatoriamente que implementar;
- desejável – característica não obrigatória, mas fortemente desejável que seja implementada pela solução;
- opcional – característica adicional apenas considerada para avaliação se as obrigatórias e as desejáveis tiverem sido contempladas na solução.

## Características obrigatórias

- Deve suportar nativamente as primitivas geométricas `line`, representando um segmento de reta, e `circle`.

- Deve suportar nativamente o conceito de figura geométrica (`figure`), que deve funcionar como um contentor de primitivas geométricas e/ou outras figuras geométricas.

  - As primitivas geométricas poderão ser encarados como figuras geométricas primitivas.
  - Cada figura geométrica tem o seu próprio sistema de coordenadas, em relação ao qual todo o seu conteúdo está referenciado.

- Qualquer tipo de figura geométrica deve ter propriedades intrínsecas. A esse respeito:

  - As seguintes propriedades devem ser consideradas: `color`, `thickness` e `filled`.
  - De modo a permitir o aparecimento ou desaparecimento de uma figura durante uma animação, as figuras devem suportar a propriedade `hidden/exposed`.
  - Deve haver um conjunto de cores pré-definidas: `blue`, `yellow`, ...

- A linguagem auxiliar usada para a escrita de bancos de figuras deve ser simples e diferente da linguagem principal. Sobre o uso de bancos de figuras na linguagem principal:

  - As figuras extraídas de bancos de figuras têm de ser embutidas no programa final. Note que durante a execução de um programa os bancos de figuras não estão disponíveis; apenas o estão durante a compilação.
  - Deve ser possível criar-se um programa *geometrics* sem se usar bancos de figuras.
  - Deve ser possível referenciar-se figuras de banco de figuras por ID. Isto significa que as figuras nos bancos de figuras têm de ter um ID único.

- Deve existir o conceito de cenário, onde a animação se desenrola. Sobre cenários:

  - Deve haver uma separação clara entre a definição de figuras e a sua apresentação no cenário.
  - A animação faz-se por alteração das propriedades das figuras e por transformações geométricas (translações e rotações). Isto pode ser feito considerando que as figuras possuem um mecanismo interno que lhes permite mudar de estado.
  - Deve possuir mecanismo de leitura, que permita a interação de um utilizador com a animação.

### Características desejáveis

- Deve suportar nativamente as primitivas geométricas `square`, `rectangle` e `ellipsis`.
- Relativamente a figuras geométricas e seus elementos:
  - Deve suportar a definição de cores no espaço RGB.
  - Deve ser possível associar-se uma função de evolução a uma figura e deve existir um operador que provoque a evolução.
  - A alteração das propriedades das figuras deve ser feito com operadores e não com métodos. (Note que isto nada tem a ver com a forma que venha a usar no código final. Provavelmente, se gerar código Java, os operadores vão transformar-se em métodos da classe que use para implementar a figura.)
  - As figuras podem possuir profundidade (`depth`) que permite decidir como se apresenta a sobreposição de figuras.
- Quanto a animações:
  - Deve ser definido o que acontece quando figuras batem nos limites do cenário.
  - A linguagem deve suportar o tipo de dados `time`, que deve aceitar um sufixo com as unidades. Deve permitir a adição e subtração de tempos com unidades diferentes. Deve permitir a multiplicação e adição de tempo com grandezas adimensionais.

### Características opcionais

- Deve suportar nativamente os elementos geométricos `polyline`, `polygon`, `arc`, ...
- Relativamente a figuras geométricas e seus elementos:
  - Deve suportar a definição de cores no espaços HSV.
- Quanto a animações:
  - Deve ser possível pausar uma animação ou fazê-la avançar/recuar para pontos arbitrário ou definidos por tags.
  - Deve ser possível fazer avançar o tempo, sequencialmente ou aos saltos.
  - A colisão entre figuras com a mesma profundidade deve ser detetada e deve haver um procedimento de atuação quando tal acontece.

## Desafios

Consegue, usando a linguagem que criou, implementar um brick breaker ou um space invaders?