



Computer Culture Advanced

信息表示与处理2

逻辑型编码实践：矢量图形编辑

从基础图元到贝塞尔曲线

逻辑型数字化（编码）回顾

逻辑型数字化是通过人为定义的**编码**规则，把人类创造的**有限的抽象符号**唯一无歧义地**映射**为**0/1 比特串**。**逻辑型数字化的核心动作是编码**。

为什么不同类型的信息要采用不同的编码方案？

内因：不同类型信息的**结构特征的不同**，需要不同的编码实现

- **数值**信息：通过**进位计数制**规则生成**数值**等价的二进制表示，完成编码
- **文本**信息：首先确定**字符集**，然后通过对集合里的字符的唯一标识（比如位置信息）**映射**为二进制比特串，完成编码
- **图形**信息：采用**图元+参数**完成编码，通过几何方程绘制具体的形状

外因：编码具备工程性，需要适配应用场景需求（硬件复用、存储 / 传输效率、标准化、表示精度等），**同一种信息，也可以存在多种编码方案**



逻辑型编码实践：矢量图形编辑

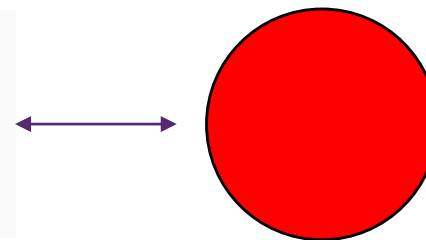
SVG（**S**calable **V**ector **G**raphics，可缩放矢量图形）是一套用文本描述图形生成规则的**标记语言**，属于逻辑型编码，记录 **图元 + 参数**，渲染器（浏览器、PPT 等软件）读取这套描述，实时渲染绘制生成图形

SVG存在
两层编码

绘图描述文本（**图元 + 参数**）→ **UTF-8**文本编码→二进制0/1

图形规则编码（**业务语义**）→ UTF-8编码（**存储层面的字符编码**）

```
<svg width="200" height="200" xmlns="http://www.w3.org/2000/svg">
  <!-- 画一个红色填充、黑色轮廓的圆 -->
  <circle cx="100" cy="100" r="80" fill="red" stroke="black" stroke-width="2"/>
</svg>
```



记事本可直接打开查看SVG文本代码

渲染器根据绘图描述绘制的图形



SVG 是一套用文本描述图形生成规则的标记语言

标记语言：使用**标签**（**标记**）来描述内容“是什么、有什么属性”的文本语言，通常成对出现，用于描述该指令的有效范围。

标记语言本身不执行绘制操作，仅用**纯文本**做结构化描述。

常见的标记语言有HTML、XML

SVG是基于**XML**规范，用来记录图

```
<svg width="200" height="200" xmlns="http://www.w3.org/2000/svg">
  <!-- 画一个红色填充、黑色轮廓的圆 -->
  <circle cx="100" cy="100" r="80" fill="red" stroke="black" stroke-width="2"/>
</svg>
```

标记指令：**circle**（**圆形图元**）

参数

- `cx="100"`：圆心 X 坐标在 100
- `cy="100"`：圆心 Y 坐标在 100
- `r="80"`：半径为 80
- `fill="red"`：内部填充红色
- `stroke="black"`：轮廓为黑色
- `stroke-width="2"`：轮廓宽度为 2

图元基本参数

样式属性参数

记事本打开
的SVG文件

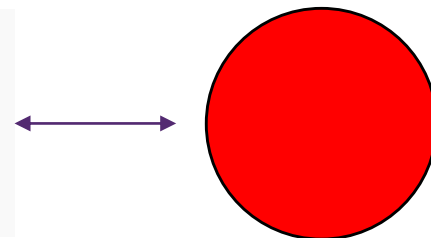




基本参数 vs 属性样式参数

```
<svg width="200" height="200" xmlns="http://www.w3.org/2000/svg">  
  <!-- 画一个红色填充、黑色轮廓的圆 -->  
  <circle cx="100" cy="100" r="80" fill="red" stroke="black" stroke-width="2"/>  
</svg>
```

SVG源代码（纯文本）



渲染器根据绘图描述绘制的图形

思考：

- ① 删掉 **r="80"** 这个参数，渲染器还能画出这个圆形吗？
- ② 删掉 **fill="red"**，渲染器还能画出这个圆形吗？

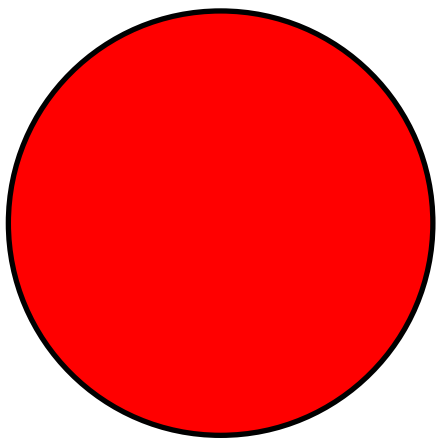




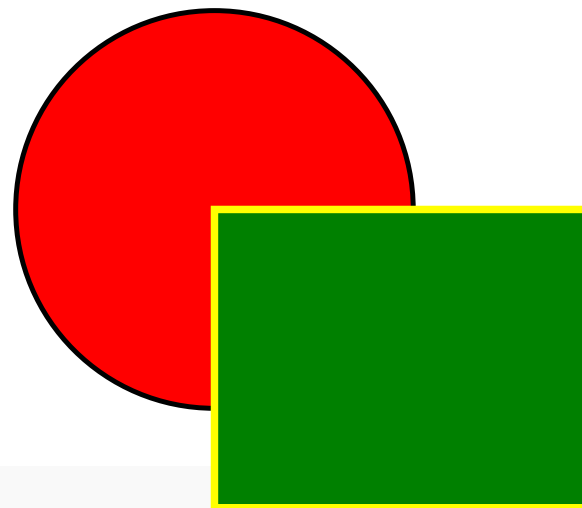
矢量图形编辑练习1

矢量图形： 编码记录图元的几何生成规则与参数，渲染器根据参数实时计算坐标，生成画面

矢量图形编辑： 不操作像素，只新增 / 修改图元规则与参数，修改绘图描述文本



任务： 在原有绘图描述文本中，新增矩形图元规则。渲染器按照代码书写先后顺序，依次绘制圆、矩形，后绘制的图形会覆盖前面的图形



```
<svg width="300" height="300" xmlns="http://www.w3.org/2000/svg">
```

```
<!-- 画一个红色填充、黑色轮廓的圆 -->
```

```
<circle cx="100" cy="100" r="80" fill="red" stroke="black" stroke-width="2"/>
```

```
<!-- 再画一个绿色填充、黄色轮廓的矩形 -->
```

```
<rect x="100" y="100" width="150" height="120" fill="green" stroke="#FFFF00" stroke-width="3"/>
```

```
</svg>
```





矢量图形编辑练习2

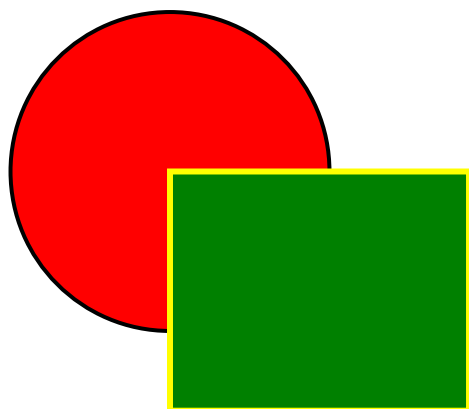
任务：

①修改 SVG 绘图描述

对圆、矩形做布尔差集（圆减去矩形，挖掉一块），再将生成的新图形整体逆时针旋转 45° ，得到右侧缺口图形

② 可视化绘图操作对照

在 PPT 中，用【插入形状】绘制圆与矩形，使用【合并形状】做剪除（差集），再将得到的复合图形旋转 45° 。



PPT 可视化绘图操作 ↔ SVG 标记文本描述，两种方式表达同一套图形规则



矢量图形编辑练习2

```
<svg width="300" height="300" xmlns="http://www.w3.org/2000/svg">
```

```
<!-- defs: 预定义复用绘图规则, 实现视觉效果: 圆形减去矩形 -->
```

```
<defs>
```

```
<!-- 定义遮罩mask: 控制画面哪些区域可见 -->
```

```
<mask id="cutMask">
```

```
<!-- 白色: 保留该区域画面 -->
```

```
<rect x="0" y="0" width="300" height="300" fill="white"/>
```

```
<!-- 黑色: 该区域透明, 实现挖除效果 -->
```

```
<rect x="100" y="100" width="150" height="120" fill="black"/>
```

```
</mask>
```

```
</defs>
```

```
<!-- 红色圆形应用遮罩: 矩形覆盖区域被挖掉 -->
```

```
<circle cx="100" cy="100" r="80" fill="red" stroke="black" stroke-width="2" mask="url(#cutMask)"/>
```

```
</svg>
```





矢量图形编辑练习2

```
<svg width="300" height="300" xmlns="http://www.w3.org/2000/svg">
```

```
<!-- defs: 预定义遮罩规则, 实现圆形减去矩形 -->
```

```
<defs>
```

```
<!-- 定义遮罩mask -->
```

```
<mask id="cutMask">
```

```
<!-- 白色: 保留该区域画面 -->
```

```
<rect x="0" y="0" width="300" height="300" fill="white"/>
```

```
<!-- 黑色: 该区域透明, 实现挖除效果 -->
```

```
<rect x="100" y="100" width="150" height="120" fill="black"/>
```

```
</mask>
```

```
</defs>
```

```
<!-- g分组标签: 把内部多个图形打包成一个整体, 统一施加变换规则 -->
```

```
<!-- 变换规则: 整体逆时针旋转45°, 旋转中心(100,100) -->
```

```
<g transform="rotate(-45,100,100)">
```

```
<circle cx="100" cy="100" r="80" fill="red" stroke="black" stroke-width="2" mask="url(#cutMask)"/>
```

```
</g>
```

```
</svg>
```





矢量图形编辑练习2-拓展

```
<svg width="300" height="300" xmlns="http://www.w3.org/2000/svg">
```

```
<!-- defs: 存放绘图规则定义，不会直接渲染到画面 -->
```

```
<defs>
```

```
<!-- 定义裁剪区域，id用来被后面图形引用 -->
```

```
<clipPath id="cutArea">
```

```
<!-- 矩形作为裁剪模板：只保留圆形落在矩形内部的区域，外部直接裁掉 -->
```

```
<rect x="100" y="100" width="150" height="120"/>
```

```
</clipPath>
```

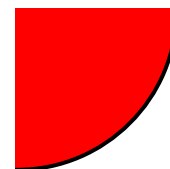
```
</defs>
```

```
<!-- 红色圆形，引用上面的裁剪规则 clip-path -->
```

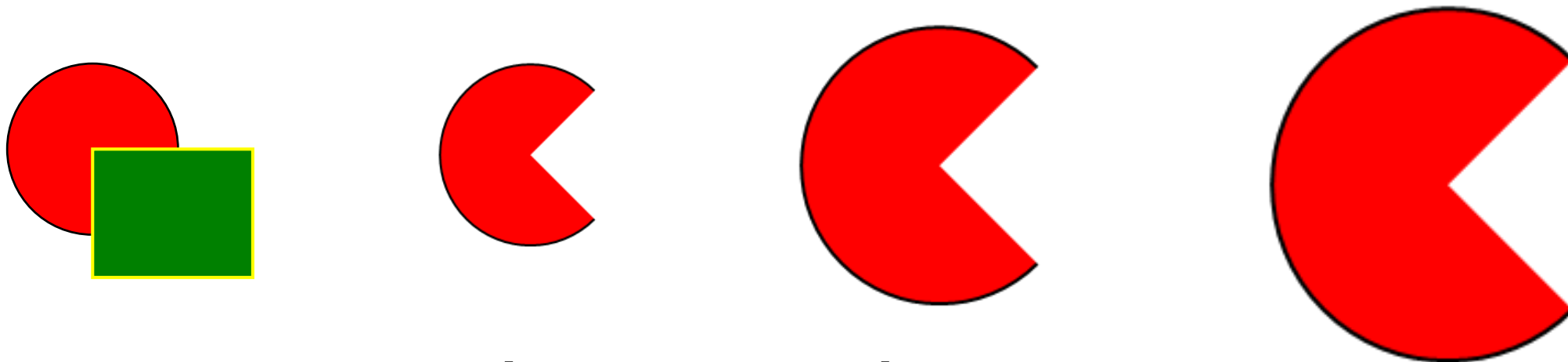
```
<!-- 裁剪运算：保留圆形与矩形重叠相交的区域，属于相交运算，和mask挖除（差集）效果相反-->
```

```
<circle cx="100" cy="100" r="80" fill="red" stroke="black" stroke-width="2" clip-path="url(#cutArea)"/>
```

```
</svg>
```



拓展：不同几何运算都可以写成 SVG 里一套独立的规则描述，属于逻辑型编码



矢量图形的编辑（逻辑型编码）：

- 图形保存为文本描述：图元（绘图指令）+ 参数
- 修改图形，本质就是修改这套规则里的参数

矢量图形的显示：

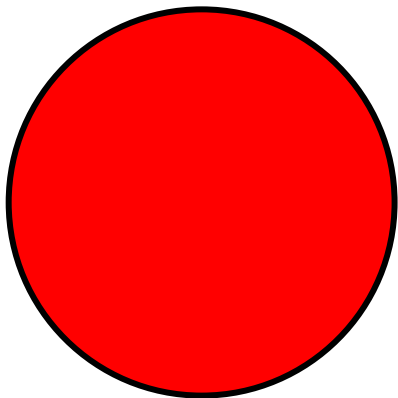
- 渲染器读取全部几何规则，实时计算图形坐标完成绘制
- 放大/缩小只是按新比例重新计算绘制，图形不会失真





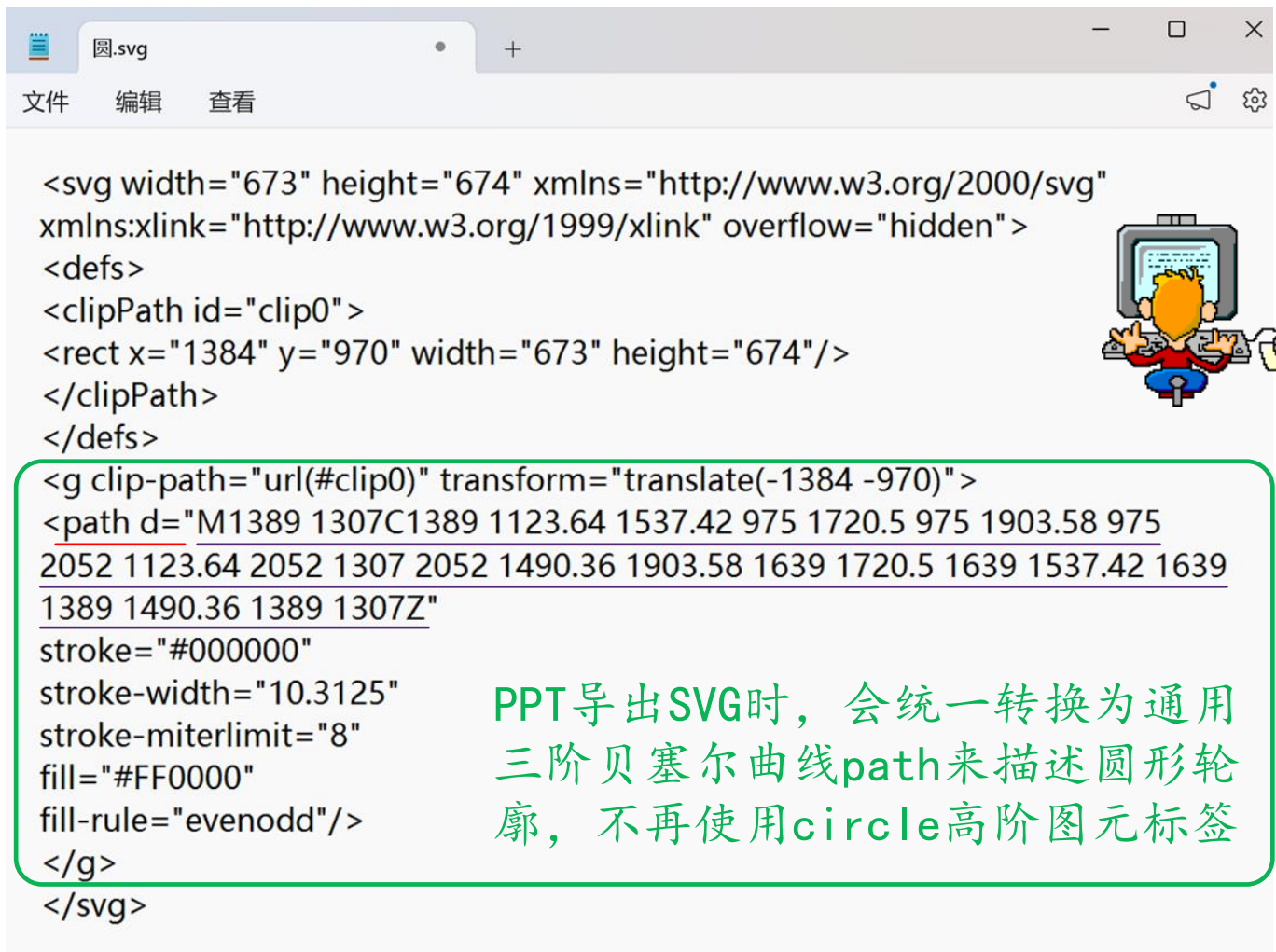
矢量图形练习3

- ❶ 使用PPT绘图工具绘制圆形（下图）
- ❷ 将该图形另存为SVG图片：圆.svg
- ❸ 使用记事本将打开该svg文件（右图）



矢量图形编辑，分为三层抽象层级：

- **最高层**：可视化 GUI 绘图操作
- **中层**：基础图元 + 布尔运算规则
- **底层**：path 路径、贝塞尔曲线坐标编码



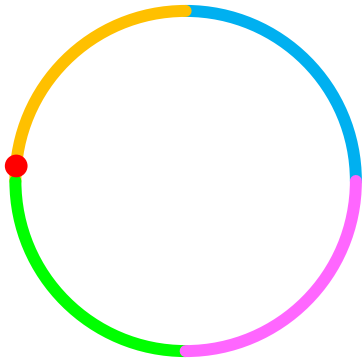
记事本没有图形渲染引擎，无法执行绘图规则，仅展示SVG的编码文本

贝塞尔曲线是一种用控制点来定义平滑曲线的数学方法，由法国工程师皮埃尔·贝塞尔发明。

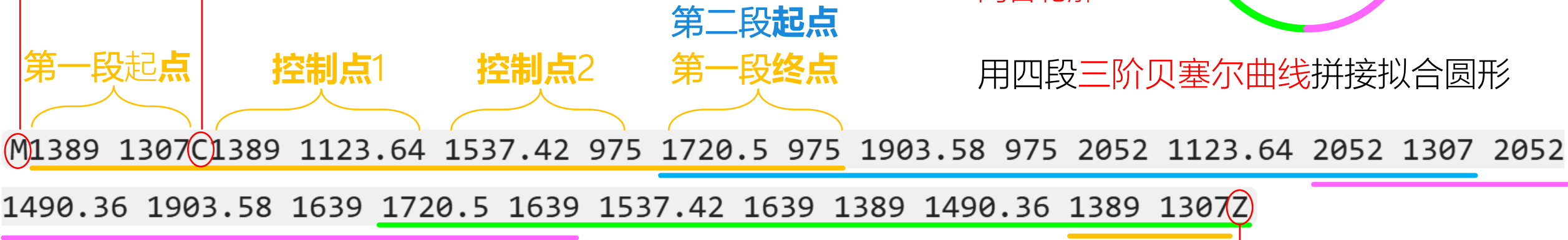
Move画笔指令，定位绘制起点坐标，比如 (1389, 1307)

三阶贝塞尔曲线的标识 (C 是 Cubic 的缩写)

四段曲线首尾依次衔接，最后依靠 **Z** 指令闭合轮廓



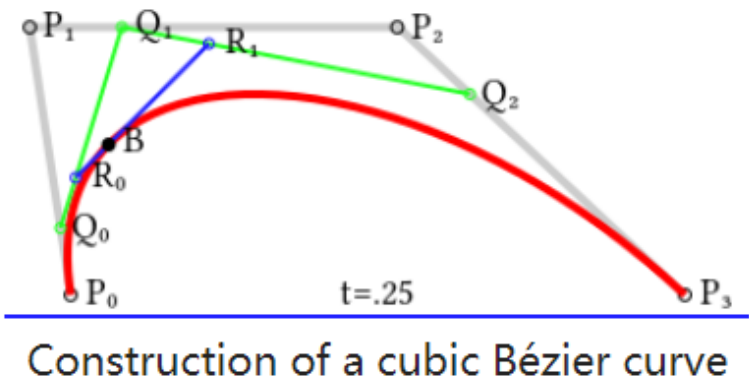
用四段**三阶贝塞尔曲线**拼接拟合圆形



轮廓**闭合标记**

一条**三阶贝塞尔曲线**由**4**个坐标点定义：

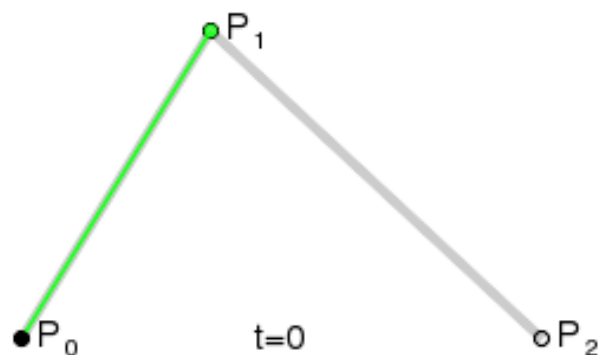
- 起点** P_0 ：曲线的起始位置
- 终点** P_3 ：曲线的结束位置
- 两个控制点** P_1/P_2 ：**不落在曲线上**，只用来控制曲线弯曲方向与弧度
 - 修改起点、终点、控制点坐标，曲线形状随之改变。
 - 直线、多边形、平滑曲线轮廓，都可以通过多段贝塞尔曲线拼接实现



认识贝塞尔曲线

贝塞尔曲线的核心特点是：

- 用少量控制点就能精确描述复杂平滑的形状
- 控制点像磁铁，把曲线“拉”在这个包围区域里面
- 可无限缩放而不失真



二阶贝塞尔曲线（动态绘制）



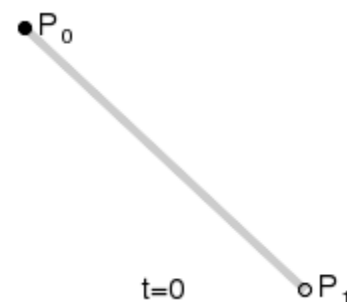
二阶贝塞尔曲线：

1 个起点 P_0 、1 个终点 P_2 ，1 个控制点 P_1

t 代表画笔从起点到终点的绘制进度：

- ① t 时刻，在 P_0-P_1 线段上取一个点 Q_0
- ② t 时刻，在 P_1-P_2 线段上取一个点 Q_1
- ③ Q_0-Q_1 连线上，按同样 t 比例，取出 B 点。所有 t 对应的 B 点连起来，就构成贝塞尔曲线。

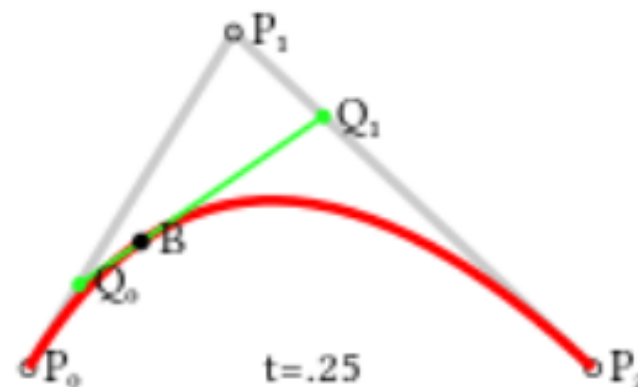
控制点 P_1 牵引着整条曲线的走向，曲线永远不会穿过 P_1 。



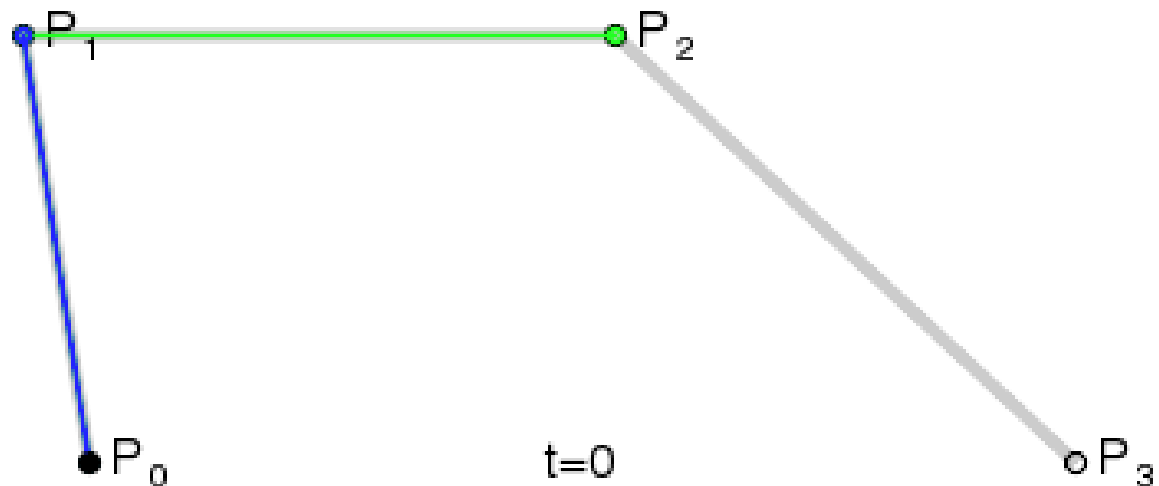
一阶贝塞尔曲线（动态绘制）

一阶贝塞尔曲线：

没有独立控制点牵引，画笔从起点绘制到终点，绘制出一条直线



在某个 t 时刻按比例取到的轮廓点 B



思考：三阶贝塞尔曲线的取点逻辑是什么？

三阶贝塞尔在**曲线形态丰富度**和**人工直观操控性**之间，取得了最好平衡



逻辑型编码实践任务：矢量图形编辑

任务主题：由一张邮票引发的艺术创作任务

实践目标：通过 PPT 形状编辑、顶点调整（贝塞尔曲线），理解矢量图形属于**逻辑型编码**：计算机保存图形的生成规则与坐标参数，而非像素画面。最终导出 SVG 文件，查看底层编码文本。

成品通过图片另存为
邮票艺术-主题.svg

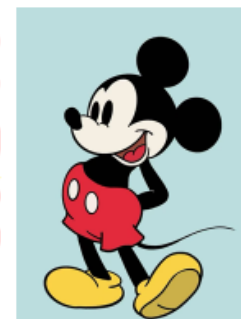
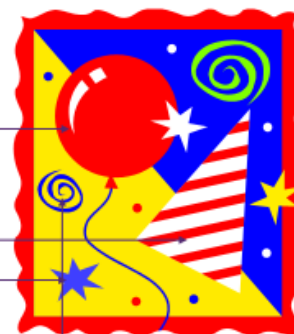
注意：导出的 SVG
打开后，图形可取消
组合、提取图元，仍
然是矢量逻辑编码

③创作的成品



《米奇的世界》

①矢量素材文件 ②选定的主题



作品以经典形象米奇为主体，整体风格为卡通漫画画风。通过将原图像中的圆形和三角形等形状编辑顶点和合并形状，打造了主体的米奇形象。将原图像中星形、螺旋形和三角形进行变形，作出背景中的手套、邮票等装饰物。作品整体童趣可爱，色彩搭配和谐。

⑤必要的素材
应用指示连线

④作品说明文字
包括：选题、关
键编辑、特点等

作品展示页（由5部分组成）保存为邮票艺术.pptx



矢量图形编辑作业要求

具体要求：

- 提交作业文件：学号-邮票艺术.zip (包含2个文件)
- 要求：主要使用作业素材 (邮票.wmf) 中提供的图形元素创作一个矢量图形作品 (主题自定)
- 图形编辑通过PowerPoint绘图工具完成编辑，最终成品保存为邮票艺术.pptx (仅包含1张幻灯片，内容包含素材和作品成品，并配有创作说明等，具体排版参考右下角示例)
- 须将该图形作品单独保存为矢量文件 邮票艺术-主题.svg
- 评分标准 (共计10分)：
 - 按要求完成并正确提交 (1分)
 - 提交的矢量文件能正确插入使用，取消组合提取其中的元素 (2分)
 - 完成度 (4分) (工作量、质量、创意、作品赏析说明等)
 - 作品展示页按要求排版，整体清晰美观 (1分)
 - 按要求在规定时间内完成挂榜分享 (2分)



作品展示页



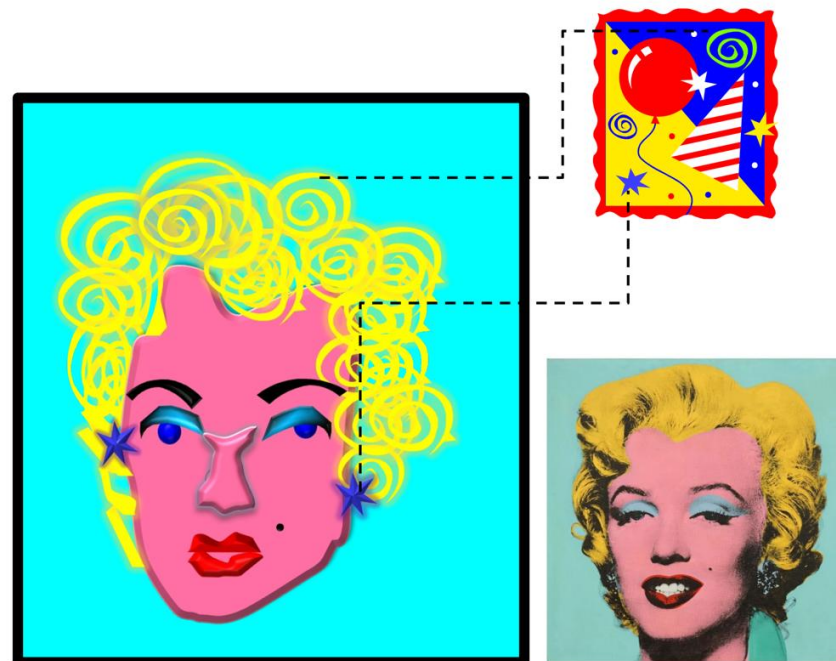
素材文件
邮票.wmf



矢量图形编辑作业规范建议

注意事项：

- **任务目标**：通过形状合成、布尔运算、编辑顶点（贝塞尔曲线）、缩放、旋转、填充纯色、轮廓设置、图层排版、图形组合等功能构建复杂图形，在可视化编辑中体会，修改图形本质就是修改图元的参数（逻辑型编码特点）
- 逻辑型编码适合描述清晰的几何轮廓，不擅长表达连续光影效果。PPT里的渐变、投影阴影、高斯模糊、发光等效果，导出 SVG 图片时，要么直接丢失，要么被转换为像素点阵（光栅化），一旦光栅化，这部分内容就不再是逻辑型矢量编码，变成了像素点阵，属于感知型数字化
- **明确禁止**：图片、纹理作为填充，图案填充导出也有风险



波普艺术与玛丽莲梦露

艺术家安迪沃霍尔
经典的梦露波普画

作品创作很有创意，但存在导出风险



关于矢量图形文件格式的补充

矢量文件存储**图形生成规则与参数**（逻辑型编码），不同格式只是这套规则的不同存储方式。

SVG（可缩放矢量图形），目前跨平台场景下通用性最强的矢量图形格式

- 无限缩放不失真：矢量图形的本质，保存几何规则与坐标参数，不是像素
- 文本可读可编辑：属于文本形式的逻辑编码，便于二次修改
- 跨平台兼容：现代浏览器、Office、绘图软件广泛支持
- 支持交互与动画：可绑定代码实现悬停、路径动画等效果
- 文件体积可控：复杂图形，文件大小仍远小于同等清晰度的位图

EMF（Windows 增强型矢量格式）

Windows 平台专属的二进制矢量格式，是 **WMF** 的升级版。Word/PPT 里内置剪贴画、嵌入矢量图形常用。仅在 Windows 生态内使用，跨平台兼容性弱于 SVG

WMF（Windows 图元文件，本次作业素材格式）

Windows 早期矢量图元文件，仅支持基础图元，不支持现代复杂效果（渐变、透明、复杂贝塞尔路径）

