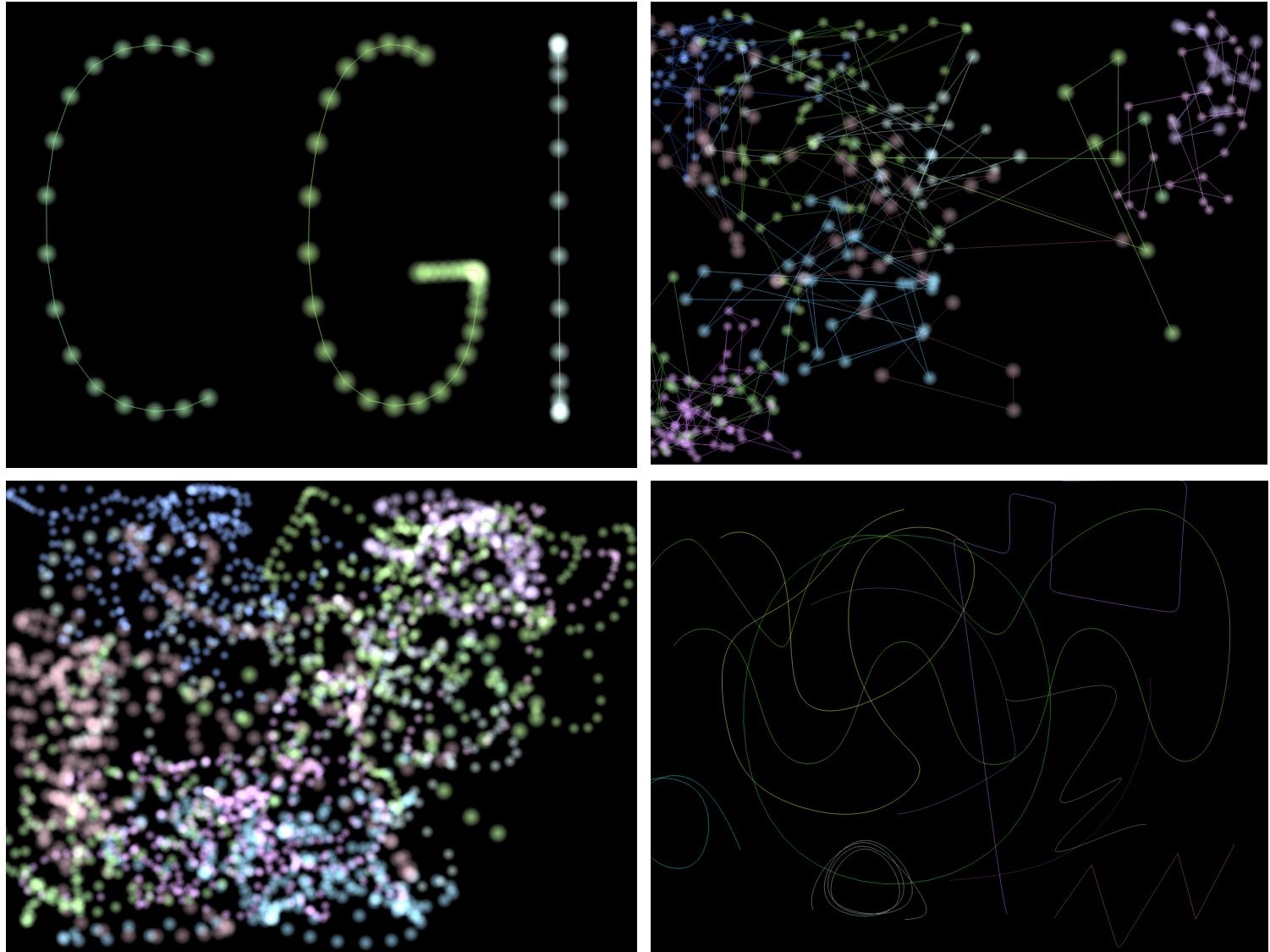


## Visualização de Curvas

B-Splines cúbicos (e não só)



### Introdução

Uma curva cúbica é uma função  $\mathbf{C}(t)$ , com  $t \in [0,1]$ , definida, a partir dos pontos de controlo  $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$ , da seguinte forma:

$$\mathbf{C}(t) = B_0(t)\mathbf{P}_0 + B_1(t)\mathbf{P}_1 + B_2(t)\mathbf{P}_2 + B_3(t)\mathbf{P}_3.$$

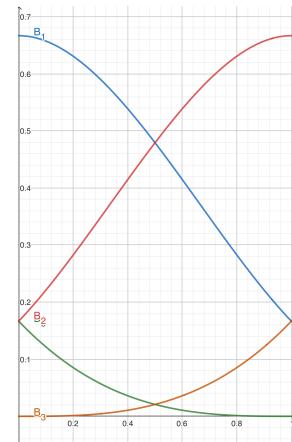
$B_0(t), B_1(t), B_2(t), B_3(t)$  são funções cúbicas polinomiais que determinam, para um dado valor de  $t$ , o peso de cada um dos respetivos pontos de controlo. Estas funções são designadas por *blending functions*.

Diferentes tipos de curvas podem ser criadas fazendo variar estas funções. Uma propriedade importante é a do somatório das *blending functions*, para qualquer valor de  $t$ , somar 1. Esta propriedade, aliada ao facto de as *blending functions* serem sempre positivas, garante que a curva assim definida está confinada à região definida pelo casco convexo<sup>1</sup> do conjunto de pontos de controlo.

### B-Splines simples

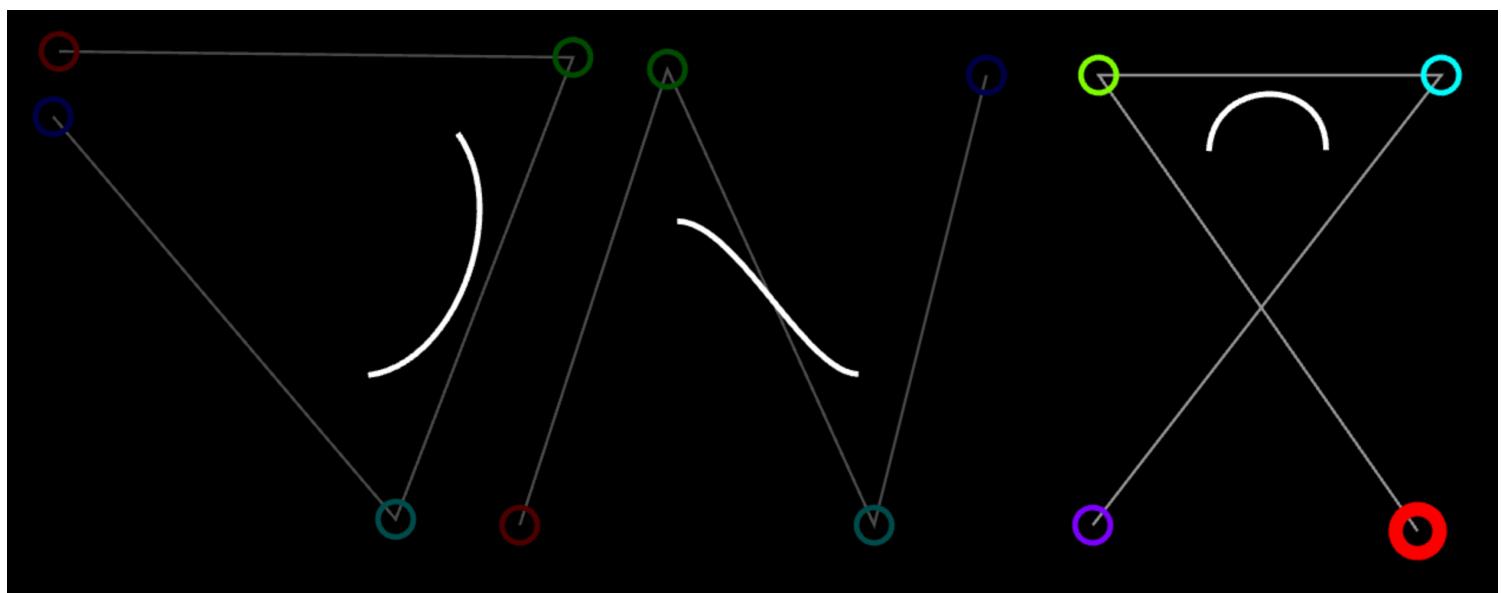
As curvas cúbicas do tipo B-Spline são definidas pelo seguinte conjunto de *blending functions*:

$$\begin{cases} B_0(t) = \frac{-t^3 + 3t^2 - 3t + 1}{6} \\ B_1(t) = \frac{3t^3 - 6t^2 + 4}{6} \\ B_2(t) = \frac{-3t^3 + 3t^2 + 3t + 1}{6} \\ B_3(t) = \frac{t^3}{6} \end{cases}$$



Note-se que no início de cada troço (curva simples), o peso relativo do último ponto,  $\mathbf{P}_3$ , é zero, situação análoga para o final do troço, mas agora em relação ao primeiro ponto,  $\mathbf{P}_0$ .

A imagem seguinte mostra vários exemplos de curvas cúbicas B-Spline simples:



<sup>1</sup> do inglês convex hull.

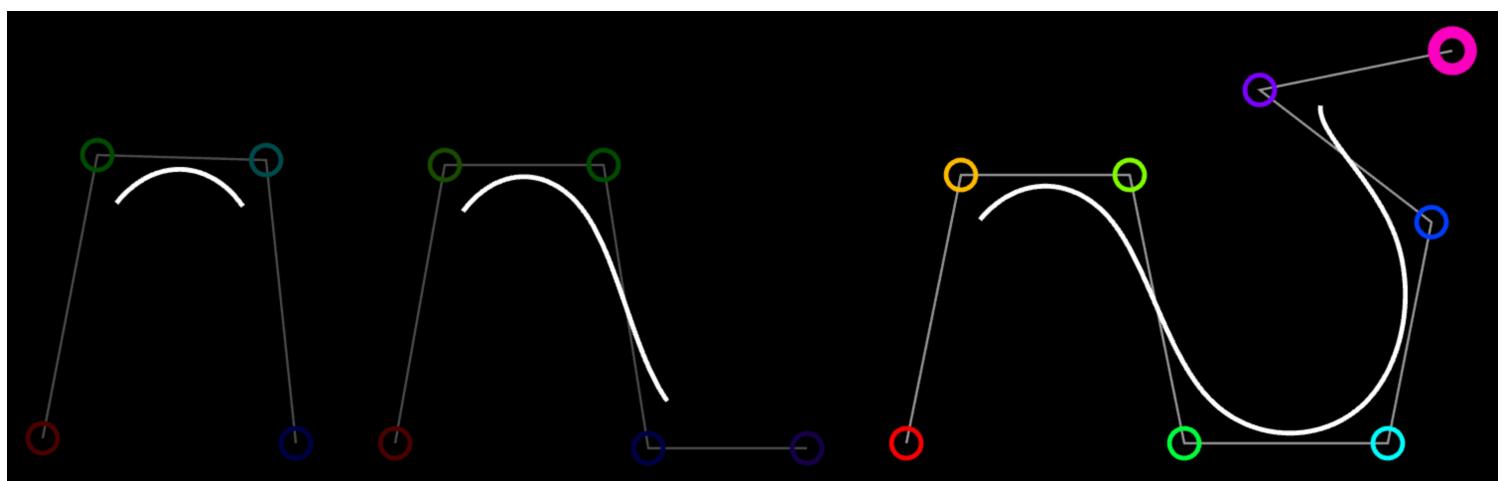
## Curvas complexas

Uma curva cúbica complexa é uma curva formada por vários troços de curvas cúbicas simples, como as apresentadas anteriormente. Para garantir a continuidade da curva é importante escolher o conjunto de pontos de controlo do troço seguinte em função do conjunto de pontos de controlo do troço anterior. Para o caso específico das curvas B-spline, é fácil mostrar que, para garantir a continuidade da curva complexa, depois de serem usados os pontos de controlo  $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$  num determinado troço  $\mathbf{C}_0(t)$ , deverão ser usados os seguintes pontos no troço seguinte,  $\mathbf{C}_1(t)$ :  $\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3, \mathbf{P}_4$ .

**Exercício:** mostrar que  $\mathbf{C}_0(1) = \mathbf{C}_1(0)$ , garantindo assim a continuidade da curva complexa  $\{\mathbf{C}_0(t), \mathbf{C}_1(t)\}$ . Mostre ainda que  $\mathbf{C}'_0(1) = \mathbf{C}'_1(0)$  e que  $\mathbf{C}''_0(1) = \mathbf{C}''_1(0)$ , garantindo assim suavidade nos pontos de união dos sucessivos troços.

Assim, generalizando, dada uma lista de pontos de controlo  $[\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3, \dots, \mathbf{P}_{n-1}]$ , os troços da curva complexa serão  $\mathbf{C}_0(t), \dots, \mathbf{C}_{n-4}(t)$ , definidos pelos grupos de pontos de controlo  $\{[\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3], \dots, [\mathbf{P}_{n-4}, \mathbf{P}_{n-3}, \mathbf{P}_{n-2}, \mathbf{P}_{n-1}]\}$ . Por outras palavras, a curva complexa será desenhada recorrendo a uma janela deslizante de 4 pontos de controlo sobre a lista inicial, avançando um ponto de cada vez.

A imagem seguinte mostra vários exemplos de curvas B-Spline formadas por 4, 5 e 8 pontos de controlo (da esquerda para a direita):



## Edição interativa de uma curva complexa

A aplicação deverá permitir a definição do conjunto de pontos de controlo para a curva que está a ser criada no momento de duas formas distintas:

- **Ponto a ponto** - Cada click do botão esquerdo do rato adicionará um novo ponto ao conjunto de pontos de controlo. A recolha de pontos da curva terminará quando for pressionada a tecla 'z';
- **Com um traço** - ao efetuar um click com o botão esquerdo, e mantendo premido esse mesmo botão, os sucessivos pontos recolhidos durante o deslocamento do rato serão adicionados ao conjunto de pontos de controlo, pela ordem pela qual foram obtidos. Note que deverá impedir a recolha de pontos demasiado perto do último ponto recolhido para evitar a criação de curvas com pontos de controlo demasiado perto dos seus vizinhos. A recolha de pontos da curva terminará quando o botão do rato for libertado.

Sempre que terminar a edição duma curva complexa, o programa deverá ficar preparado para que o utilizador possa iniciar o desenho duma nova curva.

## Objetivo

Neste trabalho pretende-se desenvolver uma aplicação WebGL que permita a criação e visualização de curvas cúbicas do tipo B-Spline. A visualização de cada curva far-se-á com recurso a um programa GLSL capaz de produzir os pontos intermédios duma dada curva recebendo como atributo (obrigatório) um índice inteiro (com início em 0) que assinala a posição do vértice na referida curva complexa.

Durante a edição duma curva, a qual permanecerá estacionária até estar completa, as restantes curvas deverão estar em movimento (excepto se o utilizador tiver optado por parar a animação). Cada um dos pontos de controlo duma curva tem uma velocidade própria, determinada como sendo uma pequena perturbação em relação a uma velocidade base igual para todos os pontos da mesma curva. Esta velocidade específica de cada ponto de controlo será calculada aquando da sua criação e deverá ser usada, a cada frame, para atualizar a posição do respetivo ponto de controlo.

Os pontos de controlo serão animados pelo código Javascript (CPU side) e deverão confinar os pontos à área visível. Sugere-se que ao ultrapassar os limites numa determinada direção, se inverta o sinal da respetiva componente da velocidade desse ponto.

O visor poderá ocupar a totalidade da janela do browser e, consequentemente, o canvas também. Contudo não se proíbe a possibilidade de mostrar na janela uma interface que permita ajustar os parâmetros do programa, devendo esta, se existir, estar encostada a um dos lados.

## Controlos adicionais da aplicação

Para além das funcionalidades já apresentadas, a aplicação deverá ainda permitir as seguintes operações:

- **Limpar** todas as curvas existentes (tecla ‘C’)
- **Aumentar/diminuir** o número de **segmentos** (segmentos de reta) usados no desenho aproximado de cada curva simples, dentro do intervalo [1,50]. Para tal, deverão ser usadas as teclas ‘+’ e ‘-‘, respetivamente.
- **Aumentar/diminuir a velocidade** com que as curvas se movem. Deverão ser usadas as teclas ‘>’ e ‘<’, respetivamente.
- **Parar/retornar a animação** do movimento das curvas, usando a tecla ‘ ‘ (espaço).
- **Esconder/mostrar os pontos de amostragem** de cada curva. Deverá ser usada a tecla ‘P’. Não confundir com os pontos de controlo da curva.
- **Esconder/mostrar as curvas** sob a forma de segmentos de reta unindo os pontos de amostragem. Deverá ser usada a tecla ‘L’.
- **Redimensionar** a janela do browser (com deformação do conteúdo mostrado), mas maximizando o canvas e o visor por forma a ocupar toda a janela.

## Detalhes técnicos

### Representação interna duma curva cúbica

Cada curva consistirá num vetor de pontos de controlo, de tamanho livre, e dum vetor de velocidades com a mesma dimensão, onde cada entrada está associada a um único ponto de controlo da curva. A velocidade de cada ponto deverá ser a soma de duas componentes:

- a velocidade base da curva (determinada aleatoriamente no início da edição da mesma);
- uma pequena perturbação aleatória específica de cada ponto.

A primeira componente deverá ter um peso bastante maior que a segunda.

Cada curva, para além da sua velocidade base deverá ainda possuir:

- um tamanho aleatório para a dimensão dos seus pontos de amostragem
- uma cor e opacidades, também aleatórias, para o desenho dos seus pontos e segmentos.

Note que a aplicação deverá manter um vector de curvas, que deverão ser desenhadas uma de cada vez.

**Dica:** Com vista a uma possível simplificação da lógica da aplicação, considere a hipótese de a curva que está a ser editada não estar armazenada no vetor de curvas, sendo armazenada nesse vetor apenas quando a sua edição terminar.

## Parâmetros globais

Como parâmetros globais (não associados a uma curva específica) a aplicação deverá ter:

- número de segmentos usados para o desenho de cada troço
- velocidade global das animações (controla a escala com que o tempo se desenrola). Quando o seu valor for 1, o tempo da animação evoluirá de acordo com o relógio do sistema.

## Desenho duma curva

Para desenhar uma curva, a aplicação deverá enviar aos respetivos shaders, sob a forma de uniforms:

- um vetor de uniforms com os pontos de controlo
- cor e opacidade
- tamanho dos pontos de amostragem.

Será ainda necessário enviar, como atributo, um índice para cada ponto de amostragem. Por exemplo, se uma dada curva complexa tem  $N$  pontos de controlo, o número de troços a desenhar será  $N - 3$ . Se cada troço for desenhado com  $S$  segmentos, o número total de pontos de amostragem será  $S(N - 3) + 1$ .

Para evitar estar constantemente a criar um buffer para cada curva, deverá criar um buffer com os índices dos pontos de amostragem, dimensionado por excesso, para um máximo de 60000 pontos. De cada vez que for desenhada uma curva apenas se indicarão os vértices necessários na chamada de `gl.DrawArrays()`.

Será tarefa do vertex shader, determinar, de acordo com o atributo índice, identificar os 4 pontos de controlo do respetivo troço e o respetivo valor de  $t$ , para que possa avaliar  $\mathbf{C}(t)$ .

O fragment shader deverá ser capaz de desenhar cada ponto de amostragem com uma forma circular. Note que poderá controlar o tamanho com que a primitiva `gl.POINTS` é desenhada atribuindo um valor a `gl_PointSize` no vertex shader. Consulte também a definição da variável pré-definida `gl_PointCoord`.

## Avaliação, regras e prazos de entrega

A avaliação do trabalho, de acordo com os requisitos pedidos é feita para 16 valores em 20. Os restantes 4 valores serão atribuídos da seguinte forma:

- suporte para curvas Catmull-Rom e curvas de Bézier (2 valores);
- efeitos visuais interessantes (2 valores).

O trabalho deverá ser submetido no Moodle em grupos formados por 2 alunos do mesmo turno. As instruções para a submissão serão publicadas em breve. Trabalhos submetidos a título individual carecem de autorização do respetivo docente do turno prático.

A data limite para a entrega do trabalho é dia **14 de Outubro**, às **23h59**.

## Apêndice 1

Uma curva cúbica, controlada por 4 pontos de controlo pode ser descrita da seguinte forma:

$$\mathbf{C}(t) = \mathbf{T} \cdot \mathbf{M} \cdot \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix},$$

com  $\mathbf{T} = [t^3 \ t^2 \ t \ 1]$  e  $\mathbf{M}$  uma matriz de coeficientes de  $4 \times 4$ . Ou seja,

$$\mathbf{C}(t) = [t^3 \ t^2 \ t \ 1] \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix}.$$

As *blending functions* são o resultado do produto de  $\mathbf{T}$  por  $\mathbf{M}$ . A tabela seguinte mostra os diferentes coeficientes para as diferentes curvas.

B-Splines	Catmull-Rom	Bézier
$\mathbf{M}_{BS} = \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$	$\mathbf{M}_{CR} = \frac{1}{2} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix}$	$\mathbf{M}_B = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$

Note que as curvas de Bézier complexas apenas serão contínuas se a janela deslizante avançar 3 pontos de cada vez, pelo que o número de pontos de controlo terá que ser da forma  $3n + 1$ , com  $n \in \mathbb{N}^+$ . As curvas Catmull-Rom não gozam da propriedade de convex-hull.