FLOP A Free Laboratory Of Programming

Skip Menu

- Home
- Practice Area
- Manage Collections
- Newbies Area
- Suported programming languages
- Join & us
- Resources

Geometría Computacional 2016
Práctica 3

- Fecha límite de entrega: viernes 29 de abril de 2016, 23:55.
- Forma de entrega: a través del Campus Virtual
- La práctica se realizará en un módulo python cuyo nombre será `bezier.py`
- La estructura básica del módulo será la que se indica a continuación. No se importará ningún otro módulo o función.

```
from __future__ import division
import numpy as np

BINOMIAL_DICT = dict()
RECURSIVE_BERNSTEIN_DICT = dict()

def polyeval_bezier(P, num_points, algorithm):
    pass

def bezier_subdivision(P, k, epsilon, lines=False):
    pass

def backward_differences_bezier(P, m, h=None):
    pass
```

- La función `polyeval_bezier` recibirá un `numpy.array` P de dimensión `(n + 1, dim)`. El parámetro `N` será un entero positivo y `algorithm` será una cadena con uno de los valores siguientes: `'direct', 'recursive', 'horner'` o `'deCasteljau'`.
  La función devolverá un `numpy.array` de dimensión `(num_points, dim)` con los valores de la curva de Bézier en los instantes dados por `N` valores equiespaciados entre 0 y 1 (incluyendo extremos).
  La opción `'direct'` forzará una evaluación directa de los polinomios de Bernstein, `'recursive'` hará que los polinomios de Bernstein se calculen usando la fórmula recursiva que los caracteriza y `'horner'` empleará el método de Horner para evaluarlos, dividiendo los valores en dos trozos: los menores que 0.5 y los mayores o iguales a 0.5. Finalmente la opción `'deCasteljau'` evaluará la curva usando el algoritmo de De Casteljau.

- La función `bezier_subdivision` implementará el método de subdivisión. El entero `k` indicará el número de subdivisiones y `epsilon` será el umbral de parada, que mide cuán cercana a una recta está la curva. Se devolverá un `np.array` que contendrá la sucesión de puntos dada por los polígonos de Bézier resultantes. Si `lines=True`, devolverá sólo la sucesión de extremos, sin puntos intermedios.

- La función `backward_differences_bezier` evaluará la curva de Bézier en los puntos de la forma h*k para k=0,...,m. Si h=None entonces h=1/m. Se usará el método de diferencias "hacia atrás" explicado en clase.

- Se permitirá el uso de dos diccionarios, que serán variables globales y tendrán los nombres que se indican en el código. Se podrán incluir otras funciones dentro del módulo, pero no habrá ninguna otra variable global u otro código ejecutable fuera de las definiciones de funciones y los diccionarios mencionados.

- No está permitido precalcular los resultados (por ejemplo, almacenando los números combinatorios dentro del código, etc.)

- Cada práctica tiene que superar un mínimo de eficiencia (<= que 1.5 veces la velocidad de la práctica de referencia). Una vez hecho esto, se calcula una ratio global de eficiencia de la práctica, dada por la fórmula

```
self.global_ratio = (dict_ratios['direct'] +\
            dict_ratios['recursive'] +\
            dict_ratios['horner']+\
            dict_ratios['deCasteljau']*2+\
            dict_ratios['bezier_subdivision']*3 +\
            dict_ratios['backward_differences_bezier']*4) / 12
```

  Las ratios que se comprueba en las sucesivas pruebas son las siguientes: [1.5, 1.3, 1.1, 0.9, 0.7, 0.5, 0.4, 0.3, 0.2, 0.1]

- El lenguaje de la práctica será el inglés.
- Se valorará la corrección y claridad del código y la velocidad de ejecución.

File Examinar… No se ha seleccionado ningún archivo. Submit