CC3501 - Aux 4: Curvas!

Objetivo

- Aprender a manipular curvas de Hermite, Bézier y Catmull-Rom
- Implementar una aplicación simple en OpenGL que implemente movimiento vía curvas.

Problemas

1) Basándose en el archivo ex_curves.py:
    a. Configure una curva de Hermite que posea una auto intersección.
    b. Configure una función que posea dos tramos, para t entre 0 y 1 debe recorrer una curva de Hermite, y para t entre 1 y 2 debe recorrer una curva de Bézier. El punto final de la curva de Hermite debe igual al punto de inicio de la curva de Bezier.
2) Implemente una función que reciba 5 puntos y evalúe una curva de Catmull-Rom con ellos. Esta función tendrá 2 tramos. La función debe utilizar el primer tramo para t entre 0 y 1, y el segundo tramo para t entre 1 y 2.
3) Modifique las curvas anteriores para que se muevan en el plano x-y. Visualícelas utilizando un plot 2D de matplotlib.
4) Modifique las curvas anteriores para que se encuentren en el interior del volumen de visualización de OpenGL. Esto es (x,y) deben estar entre -1 y 1.
5) Implemente otro programa que dibuje 10 pequeños cuadrados que muestren una curva. Esta aplicación no presenta movimiento, usted debe localizar cuadrados utilizando la posición que entrega la evaluación de la curva.
6) Modifique el archivo ex_quad_controlled.py para que el cuadrado sigua alguna de las curvas compuestas que construyó anteriormente. Hints:
    a. Escale el cuadrado en 0.1
    b. Elimine la transformación de rotación
    c. Elimine el control discreto con las flechas del teclado
    d. Defina un parámetro t = 0 fuera del loop del programa
    e. En cada iteración aumente dicho parámetro t en el dt calculado al comienzo del loop.
    f. Con dicho parámetro evalúe la función de curva compuesta de los problemas 1 o 2. Para cada parámetro t, debe obtener un vector en 3 dimensiones. La componente z, por construcción debiera ser nula, y las componentes x e y debieran tener valores entre 0 y 1.
    g. Utilice las componentes x e y de la evaluación de la curva para calcular la matriz de traslación que transforma al cuadrado.
    h. Añada una condición para que si t es mayor que 2, vuelva a 0.
7) Modifique el programa anterior para que el cuadrado se oriente según la trayectoria. La función numpy.atan2(dy, dx) le permite calcular el ángulo dada una diferencia en x e y. Con dicho ángulo, puede generar una matriz de rotación