Universidad de Chile
Facultad de Cs. Físicas y Matemáticas
Departamento de Ciencias de la Computación
CC5502-1 Geometría Computacional

Tarea 1 - Elementos geométricos

Profesora: Nancy Hitschfeld K.

Ayudante: Joaquín Torres

1. Introducción

Teniendo sus bases directamente en la matemática, la idea de utilizar la computación sobre la geometría no sólo viene a resolver problemas, sino también a mejorarlos en términos de precisión, así como en tiempo de resolución.

Sin embargo, la precisión es un tema complejo, ya que la precisión numérica de un computador está acotada justamente a cuanto puede almacenar un número (recuerden, por ejemplo, que un Integer está entre los valores $[-2^{31}, 2^{31} - 1]$ contabilizando negativos) y además a la precisión de un computador mismo para realizar operaciones. Por todo ello hoy en día los software más precisos son justamente los más caros, como los CAD, ya que en la ingeniería un error podría llevar a que un edificio fácilmente se cayera.

Paralelamente, el tiempo de ejecución es algo intrínsecamente importante. No es lo mismo resolver un problema en 5 minutos a demorarse 5 semanas. Por esta razón la mayoría de los algoritmos geométricos se programan en **C++**, siendo este uno de los lenguajes más eficientes[1] que se pueden utilizar actualmente, sólo estando a la par **C**, pero con la diferencia de que C++ tiene clases, que es justamente lo que se utiliza en geometría.

[1]Se puede revisar el siguiente benchmark como ejemplo. En este se puede ver (igual que en otros benchmarks) que lenguajes como **Java** es del orden de 2 veces más lento que **C++**. http://benchmarksgame.alioth.debian.org/

2. Elementos a implementar.

La idea tras esta tarea (y todas en general) es ir construyendo los elementos y algoritmos geométricos más tradicionales de forma incremental. Por esta razón en la presente tarea implementaremos los elementos más básicos para posteriormente usarlos y refinarlos. **Toda esta tarea debe estar implementada en C++**. Algunos puntos incluirán preguntas que deberán ser respondidas en un archivo txt sólo usando la enumeración de la pregunta. No es necesario hacer un informe.

A continuación se presentarán los elementos a ser construidos en un orden sugerido, ya que perfectamente algunos elementos pueden y serán parte de otros.

1. Clase **Punto**. Por el momento se trabajará en 2D, pero se deberá poder instanciar tanto como Integer o como Double u otro tipo primitivo, por tanto debe usar un template. Un punto debe poder realizar operaciones básicas con otros puntos como la suma, la resta, etc.

2. Clase **Vector**. No confundir con un `vector` de C++. Para ello, se les recuerda que un vector parte en el origen y se debe poder normalizar y operar con otros vectores utilizando producto cruz y punto. ¿Qué ocurre cuando se normaliza un vector con base en Integer y otro en base a Double? (Pregunta válida producto cruz y punto) Si se empezaran a usar números muy pequeños o muy grandes y principalmente números primos, ¿Qué ocurre en términos de precisión? Ejemplifique.

3. Clase **Segmento**. Un segmento está constituido por dos puntos, tiene un determinado largo y tiene la facultad de poder identificar si un punto está a la izquierda o a la derecha de este (o eventualmente sobre). Para este caso se deberá probar nuevamente con Integers y Doubles y evaluar los problemas de precisión para puntos muy cercanos al segmento en sí. ¿En qué caso es más preciso? ¿Por qué?

4. Clase **Poligono**. Un polígono contiene puntos que pueden (o no) formar segmentos. Se necesita saber si sus puntos están o no en formato **CCW**[2] para poder calcular el área del polígono. Así mismo se debe poder averiguar si un punto está o no dentro del polígono. Discuta sobre posibles problemas de precisión.

[2]Counter-ClockWise: `https://en.wikipedia.org/wiki/Clockwise`

3. Información extra

- Recuerde usar las facilidades c++ para redefinir operadores (+,-,..), cin, cout. Define todos los constructores que consideres necesario, y el destructor.
- Se recomienda hacer un archivo (o varios) con pequeños ejercicios para ir probando tanto sus implementaciones como para ver las preguntas en sí. No es necesario bajo ningún concepto que utilice librerías de Testing.
- Se recomienda la utilización del IDE **CLion** de Jetbrains, ya que al ser estudiantes de la Universidad de Chile disponen de cuentas gratis. Para esto deben ingresar a `https://www.jetbrains.com/shop/eform/students` y utilizando su correo `@dcc.uchile.cl` deberían recibir de forma casi instantánea la aprobación. CLion funciona tanto en Unix como en Windows (ahí deberán instalar primero MinWin o CygWin y CLion lo reconocerá). A menos que usted haga uso intenso de Visual Studio, este no se recomienda tanto por la dificultad de la curva de aprendizaje así como también el uso en espacio (pesa alrededor de 30GB).
- La fecha de entrega así como los contenidos de la tarea podrían llegar a modificarse según las necesidades del curso y cómo este avance, pero de ocurrir esto se notificará en clases y posteriormente se verá reflejado en U-Cursos.
- Si usted no trabajó utilizando CLion, **debe incluir README sobre cómo ejecutar su programa.**