# Decodificando Imágenes

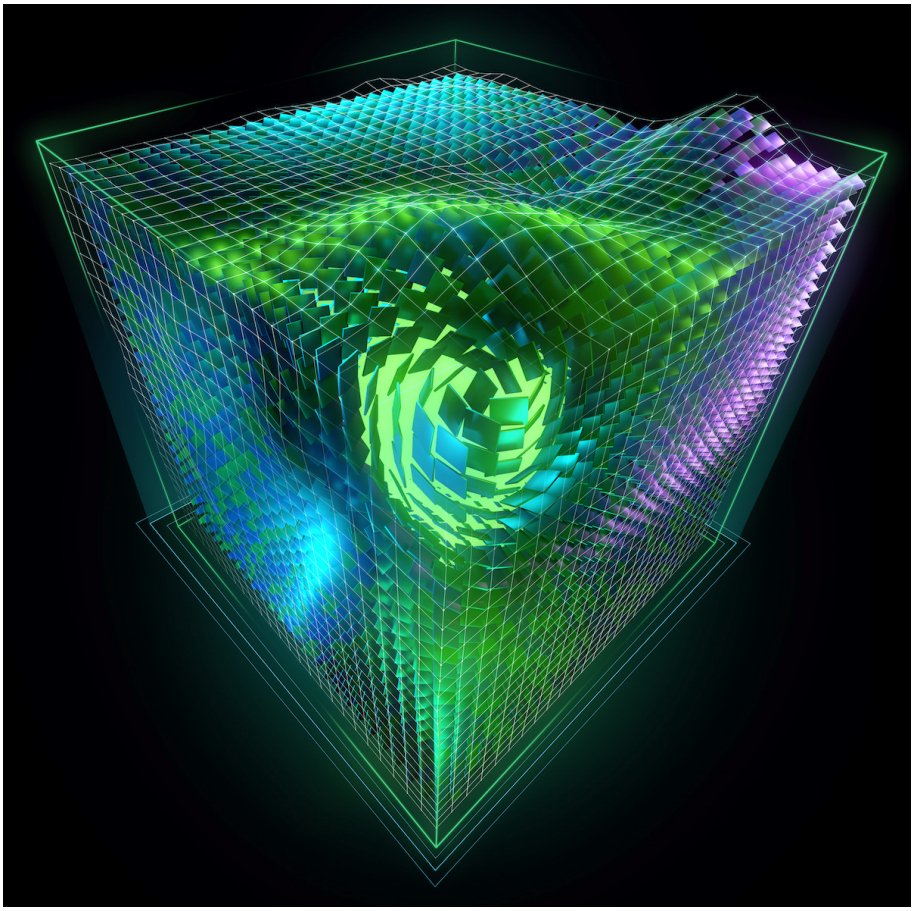
## Laboratorio 2

Ignacio Aedo - 201773556-2
Rodrigo Cayazaya - 201773538-4
Danny Fuentes - 201773559-7
Jean-Franco Zárate - 201773524-4

## Características GPU

Collab:

- Modelo: Nvidia Tesla T4
- Compute capability: 7.5



# Preguntas:

1. Incluya en su informe la imagen resultante que obtuvo. No es necesario que presente las 3 imágenes idénticas obtenidas con cada implementación. Basta con que usted se cerciore de que todos los códigos estén funcionando correctamente.

*Imágen obtenida a partir del trabajo realizado en todos los incisos.*

2. Incluya en su informe una tabla o gráfico con los tiempos obtenidos para cada implementación.

| CPU | GPU (AoS) | GPU (SoA) |
|---|---|---|
| 789.265[ms] | 0.116736[ms] | 0.113344[ms] |

3. ¿Cuál resultó ser la implementación más eficiente? ¿Qué puede concluir sobre lo observado? Comente al respecto y argumente por qué cree usted que el código más eficiente tuvo un rendimiento superior.

   La implementación de GPU con Structure of Arrays (SoA) fue la más eficiente. Se puede observar a simple vista que la implementación por CPU fue la que más se demoró debido a que son muchas iteraciones por las que tiene que pasar para procesar el archivo img.txt, en cambio las implementaciones por CUDA aprovechan el paralelismo de las hebras para procesar más rápido. Además, hay una pequeña diferencia entre el tipo de ordenamiento de datos en las implementaciones por CUDA kernel. Por un lado se utiliza Array of Structures (AoS), que es la forma más intuitiva de ordenar datos y por otro lado se ocupa Structure of Arrays (SoA), la cual, a diferencia de AoS, permite accesos a memoria coalescentes por parte de las hebras, lo que refleja una pequeña mejora respecto a AoS.

4. ¿Qué cree usted que ocurriría con sus implementaciones si P≥32? Argumente su respuesta.

   Debido a que los valores que se leen son *float*, estos son de precisión simple. Por lo que una lectura que cumpla con alineamiento, consecutividad y secuencialidad (como lo son SoA y AoS), solo puede leer hasta 32 *floats* en memoria global en una transacción. Entonces si P es mayor a 32, existirán más de 32 *floats* a los cuales deberán acceder, lo cual conlleva a realizar una mayor cantidad de transacciones de memoria.

## Conclusión

En base a lo realizado en esta tarea, pudimos notar la importancia de tener en cuenta la forma de acceder a los datos, ya que al comparar las ejecuciones de GPU, podemos concluir que el simple hecho de cambiar la forma en que las hebras acceden a los datos, puede significar una mejora importante en términos de rendimiento. En este caso observamos que SoA es ligeramente mejor que AoS, pero creemos que esta mejora no es tan amplia debido a que el problema no es demasiado grande, por lo que al aumentar la cantidad de datos con que trabajamos esta diferencia debería crecer significativamente. Por otro lado, destacamos la importancia de considerar los accesos a la memoria global, ya que como se planteó en clases, esta es la más lenta de todas, por lo que realizar muchas transacciones en ella nos significa un peor rendimiento, por lo que para el caso planteado en la pregunta 4, tendríamos que hacer alguna modificación en el planteamiento de la implementación para minimizar los accesos.