iscte SINTRA
TECNOLOGIAS DIGITAIS
ECONOMIA E SOCIEDADE

Guidelines for Practical Project

High Performance Graphical Computing

1 Introduction

This document describes the structure, expectations, and assessment criteria for the semester projects in High Performance Graphical Computing curricular unit. The work will be developed progressively in three stages, allowing students to explore parallelism, GPU execution models, and performance analysis through **CuPy**, **Numba**, **RAPIDS**, **Dask** and/or other relevant technologies discussed during lectures.

Students can select one topic from the list of proposed project areas, or alternatively develop an original project. Project evaluation will value incremental development, clarity of objectives, code quality, reproducibility, experimentation, and performance results.

2 Assessment Structure

2.1 Tutorial #1 – Project Idea and Objectives

Students shall submit a concise proposal describing their chosen topic and intended direction. The document should clearly state what will be built, why the topic is relevant for GPU acceleration, and what outcomes are expected.

At this stage **no implementation is required**, but the submission should demonstrate an understanding of the computational challenge involved and the role GPU parallelism may play in solving it.

2.2 Tutorial #2 – First Steps Towards Implementation

This second submission should demonstrate technical progress. Students are expected to present the first iteration of their implementation, including early experiments with GPU execution.

A **minimal working prototype** is required, even if incomplete. The focus should be on demonstrating the ability to allocate data to the GPU, launch kernels or vectorised operations, and begin measuring performance characteristics. Reflection on encountered limitations, memory transfer constraints or block/grid configuration reasoning will strengthen the submission.

2.3 Final Delivery – Full Project Submission

The final stage consists of a complete project implementation accompanied by a written report. The solution should include:

- A CPU **baseline implementation** (NumPy or pure Python);
- A GPU **accelerated version** using CuPy, Numba CUDA RAPIDS, Dask, or other technology that envolves GPU processing;
- A **comparison** between CPU and GPU performance, including:
  - Execution times, speedups;
  - Kernel configurations;
  - Performance analysis (through profiling tools);
  - Computational bottlenecks (if applicable);

Students are encouraged to explore optimisations and to include meaningful graphical or numerical results illustrating improvements.

3 Project Topics and Exploration Guidelines

Below are the available project areas. Students may adapt or extend these, provided that GPU programming remains central to the proposal. Project ideas 3.1–3.7 were provided to give you some exploring topics and direction. You can chose from there project ideas, or propose your own. The project idea are briefly presented during Tutorial #1, while later tutorials include additional depth to guide final work.

3.1 High-Resolution Fractals

Students may develop a visual exploration tool for Mandelbrot or Julia fractals computed at high resolution. The CPU version serves as a functional baseline, while the GPU variant should parallelise the iteration per pixel, enabling real-time zoom or colour mapping. This type of project naturally scales with resolution, allowing clear speed comparisons.

3.2 Cellular Automata Simulation

A project around Conway's Game of Life or related automata can highlight stencil-based operations and boundary conditions. After a simple CPU reference implementation, students can port neighbour evaluation and state updates to the GPU. Larger grid sizes will demonstrate performance gains and open space for visualisation experiments.

3.3 N-Body Gravitational Simulation

Simulating gravitational attraction between thousands or millions of bodies offers computational intensity suited to GPUs. The CPU model computes pairwise forces, establishing correctness. The GPU version distributes particle interactions across threads, improving scalability. The project may evolve towards visual orbit displays or alternative integration schemes.

3.4 2D Fluid Simulation – Stable Fluids

This topic invites students to create a smoke or fluid solver using velocity and density fields. The CPU baseline implements advection, diffusion and pressure projection. Transitioning to the GPU demands careful parallelisation of iterative solvers, where grid resolution and iteration count can be used to stress performance. Interaction or visual smoke rendering can enrich the final result.

3.5 GPU-Accelerated Image Processing Pipeline

Students may construct an image processing module capable of accelerating filters such as blur, sharpen, edge detection or bilateral smoothing. An initial CPU prototype should be followed by a GPU version capable of processing large images or folders efficiently. Measuring throughput, showing quality comparisons and experimenting with shared memory kernels will be considered valuable.

3.6 Monte-Carlo Simulation and Statistical Modelling

This project revolves around large-scale stochastic simulation, for example pricing models, random walks or physics interactions. The GPU implementation should distribute independent sampling tasks across threads, while CPU execution establishes correctness. This project should also compare variance convergence, batch size scaling and the performance cost of random generation.

3.7 Graph Analytics on GPU – PageRank or BFS

Graph analytics provides excellent ground for demonstrating global memory behaviour and sparse data structures. In this project, students will implement PageRank or Breadth-First Search on a large graph, ideally containing millions of edges.

4 Requirements

- Each group must clearly explain their project idea and objectives, and outline a GPU-oriented solution, developing first a functional **CPU baseline implementation** before migrating to GPU execution.
- The project should then include a **GPU-accelerated version** using CuPy, Numba CUDA or other frameworks explored in class.
- CPU and GPU versions must be compared through **performance evaluation**, including execution time, speedup, scalability behaviour and discussion of memory transfers and kernel configuration.
- Visual outputs such as performance plots, simulation results, heatmaps or real-time visualisation should be produced using Python visualisation tools (or similar).
- The complete workflow like design, implementation, profiling, comparison, discussion and conclusions must be presented in a single **demonstration application** (in Jupyter Notebook, Pygame, Streamlit, or any other tool / framework).
- Code should be clear, documented, and demonstrate methodological reasoning, including insights gained throughout development.

5 Group Composition

- Tutorials will be carried out in groups of **3–4 students**.
- Final assessment will be performed **individually**, based on understanding of concepts, implementation decisions and performance analysis.

6 Submission

All tutorial submissions (which should be named `Tutorial1`, `Tutorial2` and `FinalProject`) must be uploaded to Moodle in a compressed format (`.ZIP`, `.RAR`, or `.TAR.GZ`) named **CGAD-XX_SubmssName** where *XX* represents the group number. If the file exceeds size limits, students may submit via **WeTransfer**, **OneDrive** or **Google Drive**, including a text file with the access link inside the archive. Only the **final upload before the deadline** will be considered for evaluation.

6.1 Requirements for each submission

- **Tutorial #1**: Written document regarding **Project Idea** and **Objectives**, reflecting how your group will accomplish.
- **Tutorial #2**: Compressed file containing a **minimal working prototype** (*code*) regarding the purposed project.
- **Final Project**: Compressed file containing the **full developed project**, alongside with a demonstration pipeline of the whole project workflow.

6.2 Deadlines

- Tutorial #1 – Project Idea & Objectives: **17/12/2025**
- Tutorial #2 – First GPU Implementation Prototype: **06/01/2026**
- Final Project Delivery: **22/01/2026**
- Final Individual Discussion: **23/01/2026**

7 Oral Discussion

- The final oral defence will take place on the scheduled date announced in Moodle.
- Each discussion session lasts approximately **15–20 minutes**.
- Only **one laptop per group** is necessary to present the project, provided the program and results are ready to run.
- Grades will not be announced immediately; assessment will only be finalised after reviewing all submissions and oral discussions.

8 Evaluation Criteria

- **CA1:** Clear formulation of objectives and contextual relevance of the chosen problem.
- **CA2:** Correctness and completeness of CPU baseline implementation.
- **CA3:** Quality of GPU implementation and efficient use of GPU resources.
- **CA4:** Performance benchmarking and analytical discussion.
- **CA5:** Quality of visual results and clarity in interpretation.
- **CA6:** Technical proficiency and clarity of explanation during oral discussion.

9 Final Notes

Projects should remain technically demanding yet creative. Students are encouraged to experiment beyond the minimum specification, particularly in performance optimisation and visual or analytical presentation.

10 Additional Information

- Maximum grade: 20 points.
- Weight:
  - Tutorial #1: 20%
  - Tutorial #2: 20%
  - Final Project: 40%