Final Project

Intro to Computer Science (CS-UH 1001) - Spring 2022

Code of Conduct

All assignments are graded, meaning we expect you to adhere to the academic integrity standards of NYU Abu Dhabi. To avoid any confusion regarding this, we will briefly state what is and isn't allowed when working on an assignment.

1. Any document and program code that you submit must be fully written by yourself.
2. You can discuss your work with fellow students, as long as these discussions are restricted to general solution techniques. In other words, these discussions should not be about concrete code you are writing, nor about specific results you wish to submit.
3. When discussing an assignment with others, this should never lead to you possessing the complete or partial solution of others, regardless of whether the solution is in paper or digital form, and independent of who made the solution.
4. You are not allowed to possess solutions by someone from a different year or section, by someone from another university, or code from the Internet, etc.
5. There is never a valid reason to share your code with fellow students.
6. There is no valid reason to publish your code online in any form.
7. Every student is responsible for the work they submit. If there is any doubt during the grading about whether a student created the assignment themselves (e.g. if the solution matches that of others), we reserve the option to let the student explain why this is the case. In case doubts remain, or we decide to directly escalate the issue, the suspected violations will be reported to the academic administration according to the policies of NYU Abu Dhabi More details can be found at: https://students.nyuad.nyu.edu/academics/registration/academic-policies/academic-integrity/

Project Objectives

The final project should be a synthesis of what you have learned over the course of the semester. You should display a solid grasp on the programming constructs we have covered, and build something that you are excited about. *At this time, you can do whatever you would like to do, instead of doing what you were told to do.*

The projects should be pursued **in groups of two** and you should clearly mention the names of the group members in the project proposal. Your group member must be from your section. Once you have picked your project partner, you are not allowed to switch partners!

The project consists of the following:

- A project proposal of **a game of your choice** in written form (as pdf and max 2 pages). The proposal should provide a brief description of your idea and a list of features you plan to integrate into the game. The list should also specify which group member is doing what task. To support your idea, you can include a mock design/screenshot/drawing of your proposed game. The project proposal must be submitted via Brightspace(see due date below). You will receive feedback on your proposal within two working days.
- There will be a kick-off presentation about your proposed project during class time (see dates below). The presentation should incorporate the feedback received from your professor or TA and should not be longer than 8 minutes followed by 2 minutes feedback discussion. In the presentation, you should describe your project and if possible also show (mock) screenshots of your proposal. A quick demo of a similar game may also be helpful, but is not required. The presentation should clearly indicate who is doing what in the project, a matrix of responsibilities showing the tasks and ownership is welcome. It is mandatory that both group members actively participate in the presentation.
- A game implementation using Processing and Python, including, but not limited to:
    - Keyboard and/or mouse interaction
    - Object-oriented programming to create appropriate objects for the game
    - Image/Soundeffects (You are allowed to use images/sound files from the Internet)
    - The ability to restart the game on game over
    - A score/result indicating the performance of the player or a win/lose indication
    - A complete level or a set of challenges that the player can play
- A final submission of your project (see due date below).

A word of advice: choose a game that you also enjoy playing yourself as you will be “engaged” to it for a few weeks. Also, it should be a game you are capable of developing yourself and would allow you to improve your programming skills whilst being implemented. The game should neither be trivial nor very difficult.
Excluded games (including variations of it): Super Mario, Chess, Agar.io, Snake, Minesweeper, Tetris, Flappy Bird, Card games, Hangman.

Submission

The submission should include the kick-off presentation as a ppt or pdf file and all files required to run the project. The code itself should also be clearly documented. In addition, a sample screenshot should be submitted that adequately represents what the project is about. Also, one submission per group is sufficient.

In summary, we look at:

- Code structure, style and documentation
- Proper use of classes and functions
- No sudden crashes or apparent bugs
- Overall game design and graphics
- Complexity of the project

Submission Format and System: Please submit the proposal, presentation and a zip file containing **all** files of your final project, including images, sound files, etc on brightspace under the assignment **Final Project** on the below mentioned due dates. Due to technical constraints, submissions via email are not accepted.
Note that your solution must work using Python and Processing, otherwise your submission will not be graded.

Late Submissions: Late submissions will be penalized by 20% per 24 hours.

Important deadlines

Project proposal due:
Monday, April 18, 11:59pm (Section 001)

Project kick-off presentation:
Wednesday, April 20, during class time (Section 001)

Final submission due:
Sunday, May 15, 11:59pm (Section 001)

Grading

Criterion	Description	Points (Total: 15)
Project proposal submission	Clarity, organization, conciseness	1
Project kick-off presentation	Clarity, organization, conciseness	1
Final Submission	Program quality (clarity, comments, style and elegance of solution)	2
	Proper use of classes and functions	3
	Handling user interactivity (mouse/keyboard)	2
	The conversion of ideas into a programming implementation	2
	Code robustness (no crashes, no bugs)	2
	Project complexity	2