# ICS 168 MULTIUSER GAME PROJECT

The goal of this course is expose you to the many engineering challenges, and their common solutions, which underlie network games by having you implement a multiuser game that can be played over the Internet. The focus is on the engineering aspects of game development, not on design. You can exercise your game design ideas in the capstone project, after learning how to engineer these systems. As such, the design of your project is constrained to 3 game options:

1) Frogger
2) Snake
3) Bomberman

The development of this game exercises three main engineering issues in multiuser network games: (1) The effect of network (latency & jittery) and the use of network APIs. (2) Server-side game loops, the effect of concurrency, and the need for coordination. (3) Persistence of data and the use of database connectors. Other aspects such as art and game play are not the focus of this course and will not be part of the criteria for evaluation of the project. The project is to be done in teams of 3 people. However, contrary to the larger, longer-term project that you will do at some point in your program, in this course there will be no specialization of tasks between team members. Grouping in this course is primarily a mechanism for self-help (pair programming), rather than a means to split the work load according to expertise. During the demos, all team members need to demonstrate in-depth knowledge of the tools and the code. The project plan is divided in 4 milestones, to be delivered on Mondays every other week, as follows.

## Milestone 1 (20% Grade): (Deadline: Monday, April 6th + Monday, April 13th)

**Think about your game.** Before starting to code, or search for existing source-code, pick your game and consider the following questions:
1. How is the game played? What are the rules for winning, losing?
2. Your game will need to have login and persistence of user information. What kind of information, besides user login and password, should you store? Game scores? Levels achieved?
3. What screens will your game have (besides login and main game play, if any other)?
4. And most importantly, how will your **multiuser** game look like? What interactions are possible? How do players compete or cooperate? What kind of network interactions will you expect to have? E.g. send user+password, receive game score, move left, move right, etc.

**By Monday April 6:**
Let us know by email your choice of game and *'software stack'* for that game: what language, operating system, etc., from the following general options:
The options are:
(a) C# (e.g. Unity 3D);
(b) Web browser (JavaScript) (e.g. node.js);

(c) Python (e.g. pygame)

**Submit on EEE Dropbox**, one document per team, called "requirements draft", of length 1-2 pages that tells us, in addition to your choice of game, your plans for how the game is played, what are the game objects and their allowed movements and behaviors and, most importantly, how to make it a multiuser game.

**By Monday April 13:**
Create a git repository on github.com and inform the TA. Place on that repository the code you found and/or wrote for a single-user version of your game.
Submit on EEE Dropbox, one per team, the "final" version of your requirements document, updated based on additional conversations and the single-user code you provided. Additionally, include a high level architecture of the source code you wrote/found. You should know what each file/class/method is for (even if you don't yet grasp all the details of its inner working). The high level architecture description can be textual and/or diagram-based.

## Milestone 2 (20% Grade): (Deadline: Monday April 27th)

**Network and Database.** There are two goals in this phase of the project: (1) to add the network layer to your game so that clients can connect to the server to establish the initial handshake; and (2) to add the database layer to your game so that usernames and passwords are stored on the server, and retrieved during the initial handshake. You will develop the first version of the server and will start to add the network code to the game of Milestone 1.

The game should have a screen where the user can enter login username and password information. When the player enters a username that doesn't exist in the server, a new player should be created and stored on the server's database with the given credentials. When the player enters a username that already exists, the server should check that the given password is identical to the one that had been stored before; if it is, the game starts; if it isn't, the player should be shown the initial login screen again. In this phase, it is not necessary to coordinate the clients yet; you simply need to demonstrate that the handshake happens. Once the handshake is established, the game client behaves like the solo version. Additionally, you may also want to implement client disconnections already, as that will be needed for the final game (but demonstration of this is not required in this phase).

**Deliverables:** (1) A demo of one game client connecting to your server. You should show that the credentials are being stored in the database. You should also show correct behavior when the password doesn't match the one stored on the database. The demos will be in the morning on 04/27.

(2) The source code of your client and server should be available on your github repository by 04/27.

## Milestone 3 (20% Grade): (Deadline: Wednesday May 13th)

**Multiuser.** The goal of this phase of the project is to coordinate the game between 2 or more game clients. Your game should ensure that players win/lose when appropriate, prevent inconsistencies, and avoid the bugs related to the interaction between players.

**Deliverables:** (1) A demo consisting of two game clients and your server. The demos will be in the morning on 5/11.

(2) The source code of your client and server should be available on your github repository by 5/11.

### Milestone 4 (20% Grade): (Deadline: Wednesday May 27th)

**More and better.** The goal of this phase of the project is to make your multiuser game more solid and demoable to outsiders. Here is the list of things you need to do in this last phase:

(a) If there were network-related bugs in M3, you need to fix them.

(b) Add a score board that shows the clients currently connected along with their scores. The score is computed as follows: Get a pallet: +1 point Get another player: +10 points

(c) Detect client disconnections, and update the score board accordingly (i.e. delete that user from the score board)

(d) Allow sessions: Users can join separate game instances, by typing the name of a session. If the session does not exist, create it. If it exists, join the session. In the login screen, you would have an extra field for specifying a session. The default can be empty, leading to a “default” session.

Extra credit will be given for the following:

(e) Implementation of a Web service on the server-side to which non-player Web clients can connect and get information about the current scores. This requires flushing the scores to the database from time to time and having a separate Web server accessing the same database.

(f) Implementation of a lobby-like chatroom, where users can talk, and invite other players to start a game session together.

Deliverables: (1) A demo consisting of three game clients and your server. The demos will be on or after the morning of 05/27.

(2) The source code of your client and server should be available on your github repository by 05/27.