The big change in this evolution is that you are going to put a graphical interface *of some sort* on your game. The exact nature of this graphical interface is up to you: do you want to make a web app? a mobile app? a Java desktop GUI? You can certainly do any of these—the choice is yours! Note that if you followed the MVC paradigm well, this change will be much easier than if you did not.

The other major change is that you are now going to add **resources** to territories, and units will be **upgradeable**.

- Draw a prototype of the portion of your user interface that allows a user to enter an attack order with your UI. You can choose how to draw the interface (e.g., by hand, Photoshop, PowerPoint, OmniGraffle, etc.) You will need to draw multiple screens to show how different interactions will change the user interface (e.g., what happens when you touch a button, move a slider, or touch another element on your interface?). Be sure to include in your flow how the user selects (1) the 'from' territory, (2) the 'to' territory, and (3) the number of units.

- Upload these images to Invision (Note: you received instructions on setting up a free account on Piazza). You will need to connect these images based on the user interaction you have designed.

- **Once you have drawn this prototype, you should present it to your TA by March 27.** We suggest that you allow the TA to manipulate your prototype similar to the paper prototype video testing shown in class to make sure your design can be used without you present. During the meeting with the TA, be sure you are able to discuss what UX principles you have used or plan to use when you fully implement your design.

- In your final UI design, we expect to see you apply the UX principles you learned in class: figure-ground, similarity, proximity, common region, continuity, closure, and focal-point. Select 4 of these principles, and explain how you put them to good use in your UI.

- As before,we expect you to use issues, feature branches, pull requests, and perform code reviews. The process you use for software development is also part of your grade.

- As always, we expect your code to be commented, have good variable names, clean formatting, and well-abstracted methods. Your group should define its own coding standard, and you should all ensure that you follow it.

- Your TA is your "customer" but also your mentor. You should meet with them regularly—have sprint reviews with them at the end of each sprint within an evolution. There should be no surprises in grading—you should know what your TA thinks of your project all throughout.

- As we discussed in class, if you are having team problems, you should attempt to resolve them yourselves, and if that fails, involve your TA and/or professor. Whether your team is working well or poorly together, we ask that each of you do an *individual contribution assessment* at the end of each evolution. We will post a link to this form later.

List of things you will be graded on:

- Your prototype of attack orders + discussion of it.

- Inclusion of the UX principles.
- Functionality
- Design
- Documentation, including explanations of your applications of the UX principles.
- Testing
- Other Code Quality Factors (*e.g.*, naming, formatting, smells).
- Process (Issue tracking, CI/CD, Code Reviews, etc).

Here are the updated requirements for this evolution:

1. Networked game play.
   (a) 2-5 players should be able to play from different devices at a time. If you would like to enable more, you can.
   (b) There will be a server and a graphical (old: text-based) client.
   (c) The server is ultimately responsible for enforcing the rules of the game.
   (d) Clients may validate data in advance, however, the server must do it's own check.
   (e) Note: your server is not required to have state that persists after the server exits. You may have such if you wish, but it is not required.
   (f) The server must support multiple games at a time.
   (g) A player should have a login + password so they can return to their games
   (h) The server should allow a player to be a member of multiple games at a time (Note: this does NOT mean the client should display multiple games at a time—just that the player should be able to switch between their active games. If you want to make them log out and back in to switch games you can, even though that is not ideal from a user experience perspective.)
2. The game board shall be a map, divided into territories.
   (a) Each territory shall be "owned" by one player at any given time.
   (b) Each territory shall have a number of units in it.
   (c) Each territory shall be adjacent to one or more other territories.
   (d) The territories must form a connected graph (all territories must be reachable from any other territory).
   (e) Note that the exact map layout and details are up to you.
   (f) The client shall graphically display the map in such a way that it is clear to the player (a) which territories are connected to which other territories (b) which player currently owns a given territory (c) how many units of each type are in each territory (d) which type(s)/amount(s) of resource each territory produces[1] (e) the "size" of the territory (this is a new attribute that reflects the cost to move through the territory)

[1]The exact details of which territories produce how many of what type of resources are, of course, up to you. This includes deciding whether or not each territory produces exactly one type, or some of each type.

3. At the start of the game, an initialization phase shall take place where each player selects their starting territories and army placement:

   (a) Territories shall be divided into initial starting groups (with the same number of territories in each group.

   (b) Each player shall pick (or be assigned) one such group as her starting territories. Each player's starting territory set should have the same resource production as all other players' starting territory sets).

   (c) Each player shall have the same number of initial units, which she may place in her territories as she wishes. These shall all be "basic" units.

   (d) The exact number of initial units is up to you, or may be an option in setting up a new game. However, all players must receive the same number of units. Please do make it sensible if you make this a fixed number of units.

   (e) Initial unit placement occurs simultaneously for all players, with no information conveyed between players about the other's places until the process is complete. Put a different way, each player should be able to place their units, then indicate they are done.

   (f) Once all players have finished their unit placement, the normal flow of game play begins.

   (g) Each territory now also has a "size" which represent the cost to move through it. The total size of territories should also be equivalent between players in the initial territory groupings.

4. Turn structure: A turn has three parts, which occur in the following order:

   (a) Issue orders.

      i. There are three (old: two) types of orders that a player may issue: upgrade, *move* and *attack* (more below).

      ii. A player may issue any number of each type of these orders in a turn.

      iii. A move order must specify the number of units to move, the source territory, and the destination territory.

      iv. Units moving with a move order must have a path formed by adjacent territories controlled by their player from the source to the destination.

         A. Each move order now consumes "food" resources. Specifically, the cost of each move is (total size of territories moved through) * (number of units moved).

         B. The game must pick the minimum total cost valid path when determining the cost.

      v. An attack order results in units attacking a territory controlled by a different player. An attack order now costs 1 food per unit attacking.

      vi. An attack order must specify the number of units to attack, the source territory, and the destination territory.

      vii. Units may only attack directly adjacent territories.

      viii. A player should be able to indicate when they are done, at which point we say their orders are "committed."

ix. No player may see the orders of any other player until all players' orders are committed.

x. The server must ensure that all orders are legal.

xi. Once all players commit their orders, these orders are executed (next step).

(b) Execute orders: perform their effects

i. Move orders move units from one territory to another territory controlled by the same player.

ii. Move orders effectively occur before attack orders.

iii. Units moving *out of* a territory *do not* participate in its defense.

iv. Units moving *into* a territory *do* participate in its defense.

v. Attack orders effectively result in all attackers leaving their home territories simultaneously, then arriving at their attack target simultaneously.

vi. A successful attack (see "Combat resolution" below) results in the attacker taking ownership of the territory.

vii. All moves must follow the rules of common sense and preclude cheating: orders may not create new units nor allow a unit to be in two places at once (attacking two territories).

(c) Receive new units: At the end of each turn, one new basic unit shall appear in each territory.

i. The player shall also increase their resource totals by adding the resource production of each territory that they own at the end of the turn. Note that this means that they get credit for any territories won in that turn.

5. Combat resolution: The server should resolve each combat action, informing *all* players of the outcome.

(a) Combat between one attacker and one defender is an iterative process which ends when one side runs out of units in the fight:

i. The server rolls two 20-sided dice (one for the attacker, one for the defender). Each roll is modified by adding a bonus for the type of unit involved (Shown in the table on the next page).

ii. The side with the lower roll loses 1 unit (in a tie, the defender wins).

iii. The order of evaluation alternates between the highest-bonus attacker unit paired with the lowest-bonus defender unit, and the lowest-bonus attacker unit paired with the highest-bonus defender unit. For example, if the attacker has units with bonuses 15,8,1,0 and the defender has units with bonuses 8,8,3,1, then combat starts with A15 vs D1. Assuming the defender loses, you have now Attacker: 15,8,1,0. Defender: 8,8,3. Next you have A0 vs D8. Assuming the attacker loses, you have now Attacker 15,8,1. Defender 8,8,3, so now you do A15 v D3.

(b) If player A attacks territory X with units from multiple of her own territories, they count as a single combined force.

(c) If multiple players attack the same territories, each attack is resolved sequentially, with the winner of the first attack being the defender in subsequent attacks. For example, if

A,B, and C attack territory X held by player D, then B fights D first. If D wins, then C fights D. If C wins, then A fights C. The sequence in which the attacker's actions are resolved should be randomly determined by the server.

(d) If units from territory X attack territory Y, and at the same time, units from territory Y attack territory X, then they are assumed to take drastically different routes between their territories, missing each other, and ending up at their destination with no combat in the middle. For example, if all units from X attack Y, and all units from Y attack X, then (assuming no other players attack those territories) both attacks will be successful with no units lost by either side (since there will be no defenders at the start of the battle).

6. Victory and defeat

   (a) A player loses when he no longer controls any territories.

   (b) When a player has lost, he may no longer make any moves, and the server should automatically consider his moves to be committed (as the empty set) at the start of each turn.

   (c) A player who has lost may continue to watch the game if he desires, or may disconnect.

   (d) A player wins when she controls all territories in the game.

   (e) When a player has won, the server should announce this to all remaining clients, which should display this information. The game then ends.

   (f) When a game ends, you MAY either have the server exit, OR have it provide the option to start a new game.

7. Upgrades:

   (a) Each player shall have a "maximum technology level" which starts at 1.

   (b) Each player may issue an upgrade order to increase their maximum technology level. The upgrade order costs the following amount of technology resources:

Upgrade Level	Cost
1 →2	50
2 →3	75
3 →4	125
4 →5	200
5 →6	300

   (c) A maximum technology level upgrade takes 1 turn to complete (its effects are not available this turn, but are available next turn).

   (d) The maximum technology level can only be upgraded by one step each turn.

   (e) A player may issue any number of upgrade orders to upgrade the units they have. Upgrades cost technology resources. Upgrading a unit increases its combat bonuses (the modifier added to the d20 roll in resolving combat). Note that the costs listed below are per-unit.

Cost (Total)	Bonus	Tech Level Required
0 (0)	0	Units start here
3 (3)	1	1
8 (11)	3	2
19 (30)	5	3
25 (55)	8	4
35 (90)	11	5
50 (140)	15	6

(f) The particular types that each unit are (*e.g.*, infantry, cavalry... vs *e.g.*, ants, bees, wasps,...) are totally up to you. Just make sure it is clear what the types are and what they mean.

(g) A player can upgrade any unit to any other unit type by paying the difference in total cost, in a single upgrade order.

(h) A player can only upgrade to units of types permitted by their maximum technology level (as indicated by the rightmost column of the table).

Groups with only 3 members do not need to do any of the modifications to requirement 1, Networked Game play (multiple games + login), and do not need to have a maximum technology level (any upgrade is available for the entire game).