programming studio

Multiplayer Game System

CSCE 315 Programming Studio

For this project, your **team** will design, implement, evaluate, and present an interactive multi-player game system. The game client, that is the interactive part, which the user sees and touches, must run in a web browser (Chrome, …) and be implemented using HTML5. Throughout this project we will focus on agile, human-centered iterative design methods. Lightweight prototyping and user studies will inform your design process. Through functional prototypes and further= studies, you will learn about developing complex software while maintaining focus on user needs. You will learn to gather and analyze user data. You will present your final game design in a short video at the end of the semester, as well as in a formal final presentation.

Game Design

You are free to implement whatever multiplayer game you wish, as long as your game and its mechanics are at least as complex as Texas Hold'em (e.g. not Chutes and Ladders). We encourage you to develop an original game with interesting mechanics, which is particularly suited for the medium of the web browser. Interesting games could include implementations of board and/or card games, sprite based video games, interesting alterations of existing games, or whatever creative ideas come to mind.

The most important aspect is to engage multiple players, in real time, in **meaningful play**.

Identify something original that you believe will produce compelling game play experiences! You are encouraged to incorporate interaction with an online social network.

Discuss game and social / community choices with your professor and/or TA to confirm suitability BEFORE you proceed to develop your deliverables.

This assignment serves as the final project for this course. As such, we are expecting a VERY HIGH level of quality in the deliverables.

Deliverables Index

Deliverables Requirements

Design documents, including specifications, flowcharts, etc., will be turned in using various media, including Google Docs and **IdeaMâché**. All deliverables will include:

- Development Log
- Github Repository.

Most deliverables will also be assembled as an:

- **Iterative Design Storyboard as a Whole:**
  Free-form Web Curation
- using **IdeaMâché**.

For each component of each deliverable, work to be very specific and very useful in your process of understanding what to build, and why, in order to support meaningful game play experiences. If any component seems like busy work, it means that you are not understanding what is specified. You are encouraged to ask members of the teaching team to clarify!

Development Log

Each team must maintain a development log as a Google Doc. The development log is used to record:

- meeting notes, which should include, for each member:
    - what has s/he done since last meeting;
    - what questions or problems s/he has;
    - what s/he is going to do next, by WHEN.
- notes on discussions relevant to the project, with team members, the instructor / TA / peer teachers / graders, classmates, and/or friends
- prior work (building blocks, precedents) relevant to the project;
- questions, problems, issues, and difficulties.

Github Repository

For the code, each team should also maintain a private `github.tamu.edu` repository. The name of the repository should be `315-game-<team_name>`, where `<team_name>` is up to you. For each submission, you must make a GitHub release (aka tag) with the name `deliverable#`, where `#` is the index of that deliverable starting from 1, and submit the URL to the release.

You should maintain a README.md file in the root folder of your GitHub repository, containing instructions on:

- what are the requirements to run your code (this may involve operating systems, browsers, compilers, third party libraries, etc.)
- how to compile and run your code
- how the repository is structured (e.g. src/ for source code, static/ for resources, lib/ for third party libraries)
- who are the authors (i.e. team members), and what are the best ways to contact them

The above information should be visible when you open your GitHub repository in browser, if you are doing it correctly.

For the appropriate account names to share access to, see the contact page. Use this Google Form to submit URLs to your work.

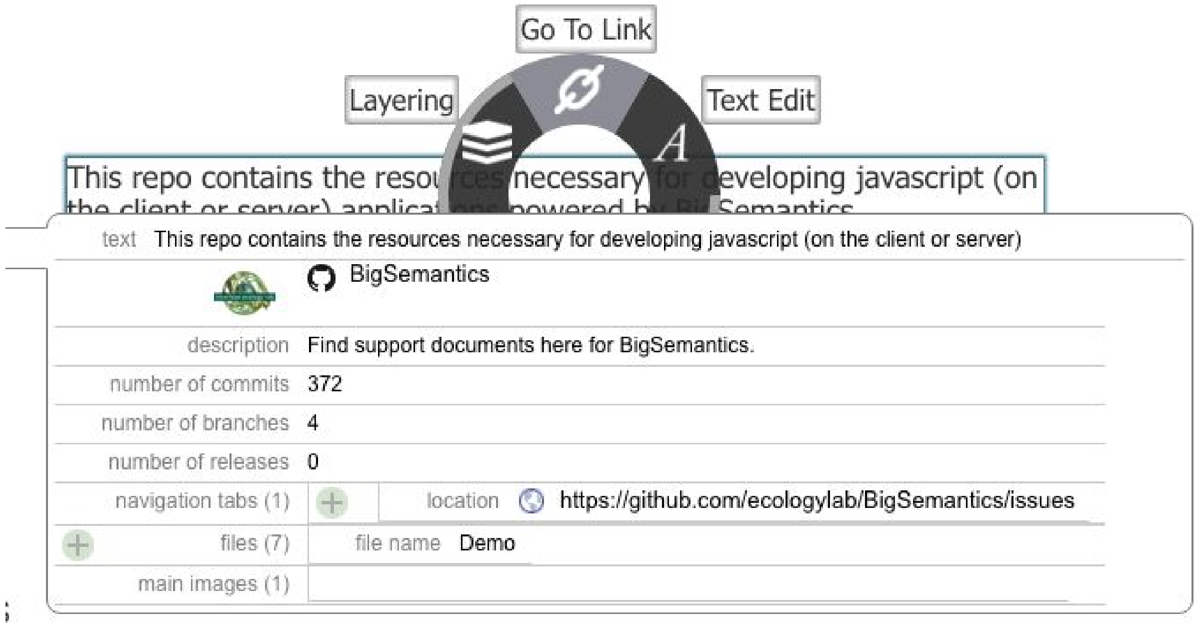
Google Docs

You are asked to present various components of your deliverables in Google Docs. MAKE SURE TO GIVE **WRITE** ACCESS TO THE ENTIRE TEACHING TEAM FOR ALL SUCH. Also, when you iterate on a GDoc between deliverables (after the first turn-in), use "Suggesting" mode so we can easily see the changes.

Iterative Design Storyboard as a Whole - Free-form Web Curation - IdeaMâché

When specified, assemble and exhibit the visual and written components of a deliverable, in the form of free-form web curation, using **IdeaMâché**. Compose the assemblage to tell a story about the user experience your design is intended to manifest.

- Arrange components of your deliverables—Google Doc(s), GitHub repository, wireframes, mockups, functional prototypes—together thoughtfully, using visual and information design principles.
- Make sure that all of these components directly afford navigation in IdeaMache. For example:



This repo contains the resou ces necessary for developing javascript (on

text	This repo contains the resources necessary for developing javascript (on the client or server)
	BigSemantics
description	Find support documents here for BigSemantics.
number of commits	372
number of branches	4
number of releases	0
navigation tabs (1)	location https://github.com/ecologylab/BigSemantics/issues
files (7)	file name Demo
main images (1)	

- Use spatial position, sketching, labeling (text annotations), and layering to develop and convey relationships.
- Use design to intentionally communicate relationships between visual presentation elements, affordances, and feedback.

- Likewise, create temporal sequences and flowcharts of to convey a **storyboard**. Use text annotations to describe the actions that result in transitions between states.
- Think of paragraphs within a Google Doc as visual components, just like areas within a wireframe or mockup.
- Use scale to nest sets of elements, where appropriate, to create readability, since you are sure to have more elements than can fit on screen, and in human working memory, at one time.

Technical Requirements: System

Developing a multiplayer game system for this project requires the development of both a web-based game client and backend server that are connected via a network communication channel. Here are requirements and suggestions for how to develop client and server components.

Game Client

- The game client MUST run in the Chrome web browser. You may choose to also support for other browsers.
- Make HTML 5 (HTML/CSS/JavaScript/...) the primary language set for your client implementation.
- As long as you document their roles, you may use available frameworks/libraries in developing your game. If you take this route, do research to pick frameworks that will actually help you, not make your life more difficult. Take a look at http://html5gameengine.com/ for ideas. Discuss your ideas and choices with the TA!
- Your project cannot require browser extensions.
- HTML5 resources can be found here (click resources).

Game Server and Communication

- The game must use network communication to support multi-user play. You must be able to support at minimum two separate, concurrent player clients.
- You may implement the backend server using any language/platform/library that you wish.
- You must use WebSockets as the communication mechanism for connecting your game clients and server. We specify that you use WebSockets, because it facilitates supporting

asynchronous bidirectional network communication between your browser clients and server. WebSockets maintains open connections between client and server. It provides standard mechanisms for the server to initiate sending messages to clients. WebSockets is an alternative to AJAX as a means for passing information between clients and a server.

- Some suggested websocket/hosting frameworks include:
  - node.js + socket.io (backend javascript; recommended)
  - libwebsockets (c/c++)
  - SuperWebSocket (c#)
  - Tornado (python)
  - There are many other frameworks available for many different languages, choose what you feel will most effectively support your project.
- One introduction to the WebSockets JavaScript API is here. Google for more!
- Data transmitted between the client and server over WebSockets must be encoded using JavaScript Object Notation (JSON). JSON is a desirable choice, because it s a straight-forward, widely-use data encoding. JSON is natively supported by JavaScript and will enable you to easily communicate from a JavaScript based web client. Again, there are many different libraries and utilities for working with JSON on the backend. Please research these and choose wisely. Some suggestions:
  - gson (java)
  - JSON.net (c#)
  - Jansson (c++)
- You may host the web-application using the means of your choice. If you have access to a web-hosting service, you may use that. The department provides basic http web hosting to students. However, WebSocket compatible hosting can be hard to come by. To get around this, you may host the application on a local machine or a departmental machine, such as `compute.cse.tamu.edu` , on a high-numbered (>10000) port. Make sure to not overly tax departmental servers with undue CPU or memory demands. Cross-domain WebSocket connections are possible, so you can make a WebSocket connection to a machine other than the machine that serves your HTML pages.

Deliverables

d1. Interactive Game Design

In your first deliverable, you will develop the design of your game. The components of this deliverable include:

- name of your game: make it compelling and clear, to attract and retain users;
- meaningful play specification [rules, discernable, integrated, core mechanics]
- lightweight prototype
- storyboard
- scenario(s)

Assemble all components together, using free-form web curation, in **IdeaMâché**, to present this entry in your **Iterative Design Storyboard as a Whole**.

Make sure to emphasize meaningful play that engages users with each other. Design the simplest possible game that incorporates your meaningful play concept!

Meaningful Play Specification

What are the operational, constitutive, and implicit rules? How are the relations between actions and outcome discernable? How are they integrated into the larger context of the game? What are the core mechanics? How will all of these lead to meaningful play?

Describe the game environment, rules,and decisions (mechanics) that players may make in the game, and how these mechanics are engaged through the interface and player-player interaction. Describe the game's operational, constitutive, and implicit rules. Be quite specific. Students must discuss potential project ideas with the instructor and TA. **Use a Google Doc for this specification.** Drag the Doc directly into your free-form web curation in **IdeaMâché**.

Lightweight Prototype

Design a **lightweight prototype**. The lightweight prototype is comprised of a combination of sketches and screenshots to depict various stages of the graphical user interface (GUI) of your interactive game system client. The lightweight prototype should map very closely to your envisioned game mechanics and interface. The prototype should be amenable to **Wizard of Oz studies**. Keep it realistic and thorough. Be able to effectively simulate all aspects of play using this prototype.

Through this deliverable, develop your design of the look and feel of web pages and interactive components, in detail. Make spatial relationships and typography precise. Indicate affordances. Create foreground / background relationships. Through all these differences that make a difference, develop layering and separation.

Submit this deliverable component in the form of wireframes and mockups.

Wireframes are widely used in visual design to represent the layout and arrangement of content for a page or module. The wireframes show where content and affordances will be placed, without specifying details such as color or content. For this deliverable, use principles of graphic space, as manifested in your visual program, to design your wireframes. Create each of these using a vector graphics editor such as Adobe Illustrator, **SVG-Edit**, **Inkscape**, or **Dia**. Output as SVG, onto your desktop, and then drag your SVGs onto your free-form web curation. Make sure to label and explain wireframes and their types, in the composition, to contextualize them. You can also make the sitemap, either as a whole, or as components, in the vector graphics editor. Or, you can create it directly in your information composition, using **IdeaMâché**'s sketching capabilities.

Mock-ups are a more fleshed out form of lightweight prototype, which build on wireframes. Design a color scheme. Create mock-ups for each presentation page, making your wireframes more specific.

Each presentation page mock-up should fully include an example instance of real content. It should include all visual elements. Create these mock-ups using a program such as Illustrator, Photoshop, **Paint.NET**, or GIMP. Output as 24-bit color png, and drag into **IdeaMâché**. Position each mock-up, in the curation, in relationship to its corresponding wireframe.

Connect your visual and information design by specifying any forms through which the user will input data.

Use multiple versions of screens and components within them, elaborated with labels, lines, arrows and other glyphs in **IdeaMâché**, to show how state may change dynamically in response to user interaction.

Scenario(s)

As part of this deliverable, develop 2 or more **scenarios** -- each, a narrative that illustrates a specific instance of how the game will be played. **Use a Google Doc to present each scenario.**

Each **scenario** tells a specific story of how a particular user will use your interactive system to accomplish her task. Let us know how important design attributes and features, including information design, visual design, and interaction design, will come into play. Use the scenario to illustrate and explain what will happen, and how it will work.

Make each scenario as specific and detailed as possible! Do not present generalizations, like that the player clicks one of 4 buttons. One scenario does not express everything that is possible. Instead, it depicts a specific user experience.

Storyboard

Create a storyboard based on the game mechanics in your meaningful play specification and the stories in your scenarios. Use spatial layout and sketching in IdeaMâché to visually connect elements from your proposal, wireframes, mockups, and anything else you need. Depict stages of interaction and game play.

The storyboard is a schematic that illustrates how the elements of your game design, including rules, mechanics, and interface will be connected together to support your scenario(s), in specific, and meaningful play experiences, in general. It is a sequence of sketches with clear transition logics.

Associate labels and stories associated with each sketch and transition. As a comic strip is a graphic novel, so a storyboard is a graphic flowchart. Storyboard details are found in the references below.

Grading Criteria

For this deliverable, your submission will be graded on:

- How rules are clearly and explicitly presented;
- How game mechanics are explained and engaged;
- How meaningful play is constituted through rules and game mechanics;
- How meaningful play game mechanics engage players with each other;
- How meaningful play is illustrated through lightweight prototype;
- How your lightweight prototype is amenable to **lo-fi user testing**;
- Visual design of lightweight prototype;
- Interaction design of lightweight prototype;
- Clarity of scenarios in illustrating and exemplifying meaningful play experiences.
- Storyboards demonstrate support of meaningful play experiences.
- Compelling and novel game design.
- Use of visual design to assemble components as a whole in IdeaMache.

In the next deliverable, you will use your prototype, storyboard, and scenarios to perform user testing. Your entire iterative design process depends on your creativity and thoroughness in this stage.

d2. Formative User Study [Lightweight Prototype] [Data, Analysis] + Revised Game Design

After submitting your lightweight prototype deliverable, and revising it in response to feedback, conduct a *formative* **lo-fi user-study** evaluating your initial design, using the prototype. Your study will use the Wizard of Oz method: some of your team-members will act as the computer, some will take notes and record data, and others will facilitate play.

Participants should effectively play your game through the prototype. Recruit your friends, family, and fellow students to participate in your study (take advantage of lab time and spring break for this). ***Observe, record, and take note of interesting interactions, difficulties, and misunderstandings that emerge while your participants engage in your study.*** Report on these findings in your study report. Iteratively revise your prototype, including game rules and interface, in response to study findings.

Make sure to focus your lightweight prototype and your user study process on gathering meaningful data about your game experience. What questions can you formulate about how participants will respond to your game and interface? How can you use this process to investigate them? As we discussed in class, *surveys about how much people like something are not very useful, because they suffer from confirmation bias*. Remember, the point of lightweight prototypes and lo-fi user studies is to prevent you from wasting resources building a functional prototype that is not focused on meaningful play.

As you conduct your study, make revisions to your prototype, in response to your findings.

Continuously interpret your findings to inform revisions of your design. Try not only to address problems that emerge in your study, but also think about how to support interesting interactions that you observed and think are valuable to the play experience.

If you design and perform the study effectively, you will uncover problems in how you communicate to your user about what is possible in the game in each state, i.e., conveying the design model to the user through affordances and feedback.

Pay special attention to

1. issues involving how your interface does and doesn't provide visual or auditory feedback when significant game events occur!
2. how your affordance designs do or don't convey your design model to users, so they understand both goals and what actions are possible.

To begin preparing this deliverable, use IdeaMache's File > Make a Copy option to duplicate your last deliverable (this enables us to see the evolution of your design.). Revise the design and specification components there as a necessary. Integrate these new components into your free-form web curation (mache):

- METHOD: A description of your study method (in a google doc). How did you run your study?
  - How many participants and sessions did you run? Who were the participants (demographics, not names!)?
  - What were the study condition(s)? If there were multiple conditions, was the study within- or between- subjects? If within, how were conditions varied?
  - What methods did you use to gather data?
- DATA / ANALYSIS: What did you learn from your study? Present an analysis of qualitative and quantitative data. For qualitative data, work to identify themes. For quantitative data, as it makes sense, provide mean values, graphs, statistical analyses. For graphs, make sure to label the axes very clearly.

  Note: in many cases, qualitative data is better than quantitative, in pointing out issues with game and interface design. Quantitative data is not required. Only in rare cases, it could be valuable at this stage.

  To articulate your analysis of the formative study data, use vocabulary from the readings, including Norman, Tufte, Itten, Rodgers-Sharp-Preece Interaction Design, Rettig, Prototyping for Tiny Fingers, and Rules of Play. Vocabulary includes:
  - layering;
  - contrast;
  - hue, saturation, value;
  - making visible;
  - feedback;
  - affordances;
  - constraints;
  - conceptual models: design model, user model, system implementation;
  - mappings;
  - knowledge in the world vs. in the head;
  - consistency;
  - rules;
  - play; and
  - core mechanics; and
  - meaningful play experience.

- DISCUSSION: What does your data mean? What problems do they expose in your game mechanics and interface design? What is working? How do you know? Do your issues remind you of any other game system you have seen? How? What new ideas do you have? Again, use the vocabulary to articulate design issues and solutions.

- REVISED DESIGN: Improve the design of your interactive multiplayer game system based on the problems you identified in the study and any other new ideas that have emerged. *Represent your improvements through revisions to your lightweight prototype, storyboard, and scenarios. Highlight the improvements to facilitate grading.*

Your submission will be graded on how well you address the above questions and revise your game design, in response to the data, to improve engagement in meaningful play.

d3. Functional Specification

While achieving the prior deliverables, also be researching, conceiving, and developing your initial functional prototype. In this deliverable, provide an initial functional specification for your game's high level and low level design.

- The high level design must include a system architecture diagram that depicts:
    - The major components / modules, such as game states, user profiles, power ups, hit points.
    - Their relationships.
    - Which components are in separate executables? Which are in the same executable?
    - How is what data passed between components?
    - How will modules communicate and work together, to support multiple players.
    - Interface elements and media assets.
    - Flowcharts / interaction diagrams to illustrate communication processes.
- The low level design must include API specifications and internal specifications:
    - Within an executable, what are the important objects? What state does each keep track of?
    - Within an executable, what are the important functions / methods?
      Between executables, what are the important messages / statements passed between clients and server via WebSockets?
        - What does each one do?
        - What is each one's *signature*? That is, how many arguments, what is the name and type of each?

        - What kind of error checking does each function, method, and message handler need to performing?
    - What code, including data structures and algorithms, will be common to clients and server?
    - Pseudo code that demonstrates the functionalities of key functions / methods / message handlers.

Specify what programming language and environment you will use on the server.

Document any external libraries or frameworks you plan to use to accomplish your proposed project. What will these frameworks do for you? What will they not do?

Consider: do you want to use a DBMS of some sort, to manage persistent state, such as user profiles and leaderboard data? SQL or No SQL? Why?

Document your hosting plans. What machine will your server run on? If you are having difficulty with this, please see the TAs!

We expect that your team will maintain a private `github.tamu.edu` repository and a **team process log** in a Google Doc. Please provide links to these and share with the appropriate instructor accounts, by this deliverable at the latest.

Using Agile Design methods, form a Master Story List. Assign relative sizes to stories. Play Planning Poker to initially assign the development work associated with particular stories to members of your project team. Perform this Scrum / Planning Poker process each time you meet during the course of the week, to iteratively update your development plan.

As part of developing this Functional Specification, begin coding. At a minimum, actually write code headers for important objects and methods / functions. Define data structures. In order to meet next week's deadline, you may also wish to begin developing functionality. Emphasize structure with minimal implementations, to start. Practice test-driven development. Engaging in this way will (1) help you be more specific and definite in your Functional Specification; and (2) help you meet subsequent deliverable deadlines.

Perform a Thin Architectural Spike, to get your technology stack working. Run distributed Hello World, to confirm that you can pass data around, as needed.

Along with your GitHub repository, use Google Docs, svgs, and pngs, as appropriate, to represent components of your Functional Specification. Integrate all components into a copy of your Lightweight Prototype **IdeaMâché** curation, to present your complete Functional Specification. Again, design the mache as a whole, to represent relationships among elements. Use the medium to help make ideas relating components clear both to yourself and to the teaching team.

d4. Functional Prototype I

Develop a first functional prototype of your game. Prioritize making gameplay work. Focus your efforts. You will need to conduct a meaningful user study for the next deliverable with this prototype. This requires:

- having basic interface components in place, which the player can perform actions on and see outcomes (affordances and feedback);
- having basic rules / mechanics implemented at the server side;
- sending and receiving a set of messages between the client interface to the server, and back, to execute the game mechanics and make outcomes visible.
- Creating priorities among user stories so that you implement as soon as possible the core mechanics.
- Focus on core game mechanics. AVOID developing secondary features, such as logins, for this deliverable.

Before you dive into development, develop a consensus in your team on what core gameplay you want to show in this prototype. Identify the subset of your functional specification that is needed now to support your core gameplay. Make sure that everyone in the team has consensus about this. Focus on this core gameplay in development. Having such a specification and shared understanding will help keep your team on the same page and help you divide the work.

Again, using Agile Design methods, form a Master Story List. Assign relative sizes to stories. Play Planning Poker to initially assign the development work associated with particular stories to members of your project team. Perform this Scrum / Planning Poker process each time you meet during the course of the week, to iteratively update your development plan.

Demo in lab to your TA and peers before and after submission. This demo will be graded!

Turn in, by incorporating into a copy of your project mache from Deliverable 3:

- Source code of client and server, as a GitHub release. See the format section for specific requirements on your GitHub repository, including the README.md file. It should also contain media assets needed to run the game.
- Instructions for how the TA / graders can set up and run your server, connect your clients, and experience your game.
- A navigable link to your demo, hosted on a server such as compute.cse.tamu.edu.
- Documents containing:

    - revised functional specification, including annotated diagrams and flowcharts, extending what you turned in for the Functional Specification deliverable. Highlight your revisions.
    - specification of how diagrams / flowcharts map to the directories, packages, class files, and so forth in your repository, forming a graphical overview of your source code. Use hyperlinking to connect the diagrams in your mache to specific GitHub files.
    - In a new Google Doc, textual explanation of how your code, as a whole, works. Think of this as explaining what is going on as if a new member just joined your team, as well as to help the TA know what to look for & where in your source code. Make sure that client, server, and common code are clearly demarcated.
- An explanation that builds on your game mechanics specification to show how your prototype, including interface and backend, are designed to implement your mechanics and thus support meaningful play. Include screenshots of your interface to support your explanation. As necessary, revise your meaningful play specification Google Doc.

  Make sure that your game play specification is connected to your functional specification, and that these are, in turn, connected to your code.

Your submission will be graded on (1) how your functional prototype supports meaningful play experiences, as well as (2) quality of code its documentation.

d5. Formative User Study [Functional Prototype] [Data, Analysis] + Revised Game Design + Functional Specification Revision

Using your latest functional prototype, incorporating all feedback so far, gather data about how users experience your game. This round of evaluation should include both formative and summative components. Develop hypotheses about the benefits of the user experience, and a study that will seek to establish them.

YOU MUST HAVE AN ACTUAL WORKING FUNCTIONAL PROTOTYPE, WHICH SUPPORTS MEANINGFUL PLAY FOR MULTIPLE USERS, IN ORDER TO WORK ON THIS.

The overall design research question your study investigates should resemble: "What is the **user model** of the game? What **meaningful play** do users engage in? How are these related?"

Use a range of techniques to develop a user study:

- Develop game play scenarios that participants will perform. In designing these situated activities, consider the user experience and what you will try to show with the study. These must be connected to your meaningful play specification.

Make clear decisions about what scenario(s) you will study.

- Observe. Consider video recording.
- Use interaction logging to record what's going on in the game system. Make sure you can see which game controls players use and actions they take, and in what sequences. Turn in one example interaction log file, in your mâché, as a Google Doc.

  Perform log analysis to understand which operations and sequences of operations that user are and are not performing, which are possible in your game. In turn, use this data to argue for (1) parts of your game design that are working; and (2) parts of your game design that you need to revise.
- Invoke qualitative data gathering methods, focusing on unstructured interviews. You can also use, questionnaires, and think aloud.
- Develop qualitative data by analyzing significant phenomena that you discover in observations, video recordings, and interviews. Derive themes / codes through grounded theory.
- Perhaps develop conditions / dependent variables that you will vary while the tasks are being performed. Define alternative interface components and game mechanics that can you benefit from comparing. Test to see which works better!
- As you run your study, make improvements in your Functional Prototype. Present a log of what these improvements are and what motivated them. Especially of interest are improvements made in response to data.

For many aspects, engaging in grounded theory to derive significant QUALITATIVE DATA WILL BE MORE EFFECTIVE than quantitative data and questionnaires.

As always, make sure to respect the privacy of your participants when you record the data and report. As you conduct your analysis, make the relationships between the data and your hypotheses very clear. In Google Docs, create a 4 - 5 page Game Data and Analysis report, which presents an analysis of the data, the lessons learned about how to make your game better support meaningful play, and the implications for your next round of development. Use the specification for the previous Lightweight Prototype User Study to guide your presentation.

Again, this deliverable builds on the last one. Create a copy of your last IdeaMache for presentation of this deliverable. Intentionally manage the scale of this now quite large collection of specification, implementation, and data components.

Again, continuously revise, as needed your Meaningful Game Play Specification, your scenarios, storyboards, Functional Specification, and code. Express connections between components across all of these levels. Show what you have revised. Explain why.

Revise your game system, including game mechanics, interface, and backend, in response to what you learned through the study. Develop a **Functional Specification Revision**, incorporating the deliverables that were defined for the previous Functional Specification.

Again, your submission will be graded based on study method, study findings, how well the revisions connect to the findings and are presented, and how well components and their elements are connected, as a whole.

d6. Functional Prototype II -- Visual, Game Experience

Iterate the design of your game, including game mechanics, user interface, and back end, based on what you learned in the study. Demo in the lab to your TA and peers before submission.

Deliverable set is equivalent to Functional Prototype I. However, by now we expect the game to be finished, not just basically playable. Also include a 1 page report on what you changed since Functional Prototype I. Once again, develop this deliverable as an instance of your **Iterative Design Notebook as a Whole**, in the medium of free-form curation, using **IdeaMâché**.

Again, using Agile Design methods, form a Master Story List. Assign relative sizes to stories. Play Planning Poker to initially assign the development work associated with particular stories to members of your project team. Perform this Scrum / Planning Poker process each time you meet during the course of the week, to iteratively update your development plan.

Your submission will be graded on:

- game experience
- functional iterations, i.e. what have changed in functionality to better support meaningful play explanation of the motivation and implementation of functional iterations. connect to recent user study, prior prototypes, and dialogues with the instructor / TA in your discussion
- code quality

d7. Video

Make final updates to your Multiplayer Game System code and interface. Make a clear, coherent video that shows off the best parts of it, demonstrating and explaining what's exciting about the experiences of meaningful play that are supported, both visually and with narration. MAKE SURE TO SHOW HOW MULTIPLE PLAYERS EXPERIENCE THE GAME TOGETHER! The maximum length of this video is 2 minutes. Note that making a shorter clear presentation / explanation is more challenging than a longer one.

Ideally, capture footage and render video at full 1080p. Upload your video to YouTube. Tag it thoughtfully, when you do, to promote viewership.

Drag your video into a mache that represents your iterative design process of the game as a whole.

Note: DO NOT use video / effects to fake the capabilities of your game. THIS WOULD BE AN HONOR CODE VIOLATION. Use them to show off what it really does and how people experience it. Provide explanation for your design choices on all levels.

Your submission will be graded on:

- Game functionalities concurrently engaging multiple players at one time in meaningful play.
    - Visual game elements.
    - Audio game elements.
    - Feedback and affordances.
    - Make sure to focus / zoom in on interactions among players! Visually convey these multi-player meaningful play experiences by combining footage of game screens and camera footage of human players (at their computers). Thoughtfully edit together these types of footage (techniques like intercutting or split screen or montage make sense here.).

- Video production: Effective use of video and audio techniques:
    - video editing techniques;
    - choice of footage;
    - visualization techniques;
    - thoughtful narration;
    - sound mixing / effects / soundtrack;
    - resolution -- should be 1080p, at least 720p.
    - Visual as well as auditory / textual explanation / discussion on support for meaningful play. Use of explanation and editing techniques to show support for meaningful play.

- Effective use of presentation time in the video

- ~~Storytelling of your design process as a whole.~~ [made optional by popular demand]

d8. Final Presentation

Make a tightly focused 5 minute presentation about your project. Use up to 2 minutes of this to show your video. Note that a shorter presentation is harder to perform than a longer one, because you have to make very clear choices about how much space/time to give elements of content.

Plan your presentation with an underlying storyboard / outline that assigns times to content chunks. Consider cutting your video into segments and interleaving other presentation.

Address these pivotal issues about your project development process, in conjunction with the final product:

- How is your game playable? How do users experience it?
- What did you learn through each deliverable?
    - How did user data stimulate iterations in which aspects of your design? Be very specific about what data resulted in which changes!
    - How did game mechanics change?
    - How did the interface change?
    - How did the software architecture and structure change?
- How did your design as a whole evolve?
- What did you learn as a whole?

Address and connect these key components of your multiplayer game system:

- game mechanics
    - What are the mechanics?
    - How do they engage users in meaningful play? (show me, don't tell me!)
    - Which ones in particular focus meaningful play experiences among users?
- software design:
    - How did you structure your software (system architecture, object-oriented design (what are key objects and functions)?
    - How did you design ***message passing*** among clients and server? (what are the messages? What state is passed?).
    - Did you implement common code across client and server; such as for maintaining and sharing game state? Why or why not
    - How did you use agile design methodologies to manage your process?
- Interface design
    - Affordances
    - Feedback
    - User model

    - Visual and auditory elements conveying game mechanics.

How do these combine to support user experiences of meaningful play? Make an evidence-based argument in favor of your game and the meaningful play experiences it is designed to support. Illustrate your assertions with screenshots, video and any other useful graphics / media.

Your presentation will be graded based on:

15	Visual Presentation (Slides or Mache)
15	Speaking
15	User Experiences of Meaningful Play
15	Game Mechanics
15	Software Design (address message passing!)
15	Interface Design
10	Evolution of Ideas

Remember: You need to design visual and oral components of your presentation to complement each other.

Slides: Use all visual design principles we have learned. Make text succinct, as an outline. Don't include all the details in the text on the slides -- you will speak them.

Never simply read from your slides.

All members of your team who have been actively involved in creating your project MUST be actively involved in the presentation.

More about how to give a presentation is **here**.

d9. Final Report

Submit a Final Project Report both in the form of a non-linear IdeaMâché, which contains, among its elements a linear, traditional PDF. Each version must address all of the following:

- Make an evidence-based argument in favor of your game and the meaningful play experiences it is designed to support.
- Updated versions of your meaningful play specification, mockups, and scenarios. Make sure to *show, using evidence, how* multiple players engage in meaningful play with each other [25].

- Your high level software design, presenting major message passing and within-process flows of control [10]. Depict connections between software functions and key aspects of meaningful play experiences. (diagrams + text)
- A narrative account of the highlights of your iterative design process, addressing:
  - problems that you identified in your design, as you proceeded through the deliverable steps [10];
  - the data that enabled you to identify the problems and think about solutions; (at least some of this should be user study data) [10];
  - changes you chose to make to your design to solve the problems you identified [10].
  - lessons learned. How would your approach change in the future? What advice can you give others based on your experiences? These are also called, "Implications for Design." [5, with extra credit possible]

    These should be on the one hand, general, and on the other hand, directly based on your experiences and data.
- Final assemblage of **Iterative Design Notebook as a Whole** using IdeaMâché [10]. Make sure your video, on YouTube, is included.
- Assemblage of whole report as a traditional linear paper, as a Google Doc (~~or in the PDF format~~) [10].
  - Consistent and clear writing, including use of language.
  - **Expository writing structure**.
  - Consistent, clear formatting.
- Up-to-date **Team Process Log** [10].

In additional, each individual team member needs to submit (using **this Google Form**):

- Self Evaluation: honestly rate your performance on the scale of 1 to 5, 5 being the best. Briefly discuss how you contributed to the project.
- Peer Evaluation: honestly rate all other team members on the scale of 1 to 5, 5 being best. For each member, briefly comment on their contribution. If you disagree with the workload distribution reported by the team, make this explicit. Provide your version, and written explanation of what went down.