Submariner Design Document Outline

1 Executive Summary

High Concept: Four players are placed in a submarine setting, and each player is placed in their own isolated room. Players can only communicate with each adjacent player, which will be implemented in demonstrations using a headset and Mumble server. Throughout the game the four players will be put into high intensity tasks and challenges which must be completed using each player's strengths. Specifically, the four players will choose between the following characters: Captain, Gunner, Navigator, and Engineer. In order to survive and successfully complete their tasks, the players must work together and share what partial information they have.

Target Audience: This game will be targeted towards gamers who enjoy puzzles and team oriented challenges like escape the room. The heavy emphasis on cooperative gameplay will appeal to the local co-op audience captured by such titles as Keep Talking, Nobody Explodes, and Space Team.

Market Appeal: While there are examples of games where a puzzle-solver requires input from a local group of friends, we are not aware of any examples of puzzle games where all players are involved in the interaction simultaneously. Submariner allows all involved players to work towards surviving the puzzle together.

Gameplay: The game will be played in virtual reality using a headset for communication. Four players are required to suit the four characters. The submarine is arranged in a series of four adjacent in-line rooms with Captain in front, Gunner behind, Navigator in the third room, and Engineer in the stern. Each player's room contains some subset of information on the state of the submarine and has some interactable objects to influence the state of the submarine. The key aspect of gameplay is that these displays are all separate from the mechanisms to influence them, requiring the crew to communicate across rooms. Communication between Captain and Engineer for example must be "telephoned" across the Gunner and Navigator.

This is not an actively-driven or controlled submarine. This is not a submarine fighting game. It is a puzzle game. There is no periscope and the players cannot see the environment outside. The entire game is one of mitigating disasters within the sub through solving puzzles, such as engine breakdown, while following orders that come to the Captain. As another example, the Captain might receive the order to dive 30 feet. Passing the order down to Engineering will have them pulling a lever to begin descending. The Captain meanwhile must

monitor their depth and send the Engineer the order to stop the dive. Each task has a time limit to complete and tasks might come in simultaneously, requiring multitasking on the part of the Captain and crew. If the submariners manage to keep their vessel intact for some duration, they win. The game itself can be structured as a “campaign” made from various scenarios of pre-programmed tasks. For more detail on potential tasks, see the **Game Ideas** subheader under the Design section.

Implementation: The game will be implemented using Unreal Engine version 4.15, with a Mumble server for the selective-hearing in demonstration. Conducting the demonstration needs headphones and HTC Vive headsets. The game itself will be fully-playable in non-VR via mouse or gamepad. We will try to use pre-existing assets as much as possible to create the setting of the submarine. For implementing multiplayer, we will be using Socket IO in lieu of Unreal Engine’s built-in replication tools. We make this decision to avoid the added complexity of ensuring proper replication and authority management among four players, support dedicated servers, and make the server logic easier to implement in a programming language the team is familiar with.

Schedule: We plan to strictly follow the schedule of requirements as outlined in the main assignment document. More specifically, we propose the following milestones. As of writing this document, we have completed an initial alpha demonstration featuring a working player character, interactable objects, intuitive user interface, single room of the submarine, and communication with a server. To keep team morale and understanding of the project high, we will have (at least) a weekly playtest to work out bugs and new ideas. For every playtest, we will test using fully-packaged versions on live, remote servers. By each playtest we will have established some features:

3/25/17: Finish the game design document and run team through playable alpha.
4/1/17: Finish server communication necessary to drive minimally-viable game in VR.
4/8/17: Add more challenging tasks and finish procedural puzzle generation.
4/15/17: Begin polishing and improving assets.

2 GAME Design - CREATIVE

2.1 High Concept

What is the basic concept of your game.

- Submariner is a game where four intrepid players work together with limited communication to complete tasks (similar to mini-games) such that the team can successfully survive naval engagement and keep their submarine afloat.

2.2 Design Goals

2.2.1 Main Design Features

What is it like to play the game, including:

2.2.1.1 Player goals and objectives.

Main goal: To keep the submarine intact by following a series of orders passed by radio from Fleet Command (remote server) to the Captain.

Secondary goals: Avoid failing any of the orders passed down by Fleet Command to achieve a higher score.

Type of challenge(s): Puzzle challenges, clarification of intentionally-ambiguous communication.

Type of conflict(s): Communication, especially handling the delay between separated crewmembers.

Winning condition: The player wins the game if the submarine survives the length of the travel and crew members successfully complete the Command's tasks.

2.2.1.2 Main rules and procedures

Operational rules: Every player must frantically fulfill their own responsibilities aboard the submarine while making sure not to let their team down. They must keep their own room running in proper order while also assisting in orchestrating crew-wide tasks.

Main game mechanic: Partial information is the name of the game. No one has enough details to accomplish what they need to do to survive alone.

On a second by second basis players will communicate with each other while trying to figure out how to interact with the various parts of the submarine.

On a minute by minute basis players will work to establish more effective modes of communicating across the submarine, especially as they become more used to the challenges facing them.

What makes this fun or interesting: Every player will be working closely with friends to solve some peculiar puzzles in a high-stress environment.

2.2.1.3 Player Resources

The only resources we could think of are **air** and **torpedoes**. Air could periodically run low and require momentarily surfacing during safer lulls in the fighting to restock. Torpedoes are expended in fulfilling orders from Fleet Command.

2.2.1.4 Boundaries and Constraints

Describe the game environment and any limitations it imposes on the game play:

The game environment is similar to an enclosed space in a submarine. This prevents much movement so users will not be allowed to teleport around the scene. For fulfilling the advanced locomotion requirement we may allow some teleportation from one crew-specific compartment to another. For example: the Engineer might actually have two rooms which they are able to move between.

2.2.2 Appeal

What is the game genre? Puzzle.

Who is the target audience? This game will be targeted towards gamers who enjoy puzzles and team-oriented challenges like escape the room. The heavy emphasis on cooperative gameplay will appeal to the local co-op audience captured by such titles as Keep Talking and Nobody Explodes.

What is their age? This game targets users above 12, such that the users have the capacity to really understand the puzzles.

Why is the game fun to play? The game is a unique challenge in a unique setting with a unique implementation of multiplayer VR. The way the game puzzles are designed give us enough leeway to have the game range from a casual experience solving puzzles with a few friends to ridiculously-challenging experiences that only the most practiced of teams could hope to complete.

2.2.3 Look and Feel

Style of artwork. Color schemes. What other games is it similar to?

The style of the assets and the game play will be that of an underwater submarine. Some assets will be similar to Bioshock's damp, dark and shady lighting/environment.

2.3 Worlds, Characters and Story (if relevant)

2.3.1 Backstory

If appropriate, how you will reveal this to the user?

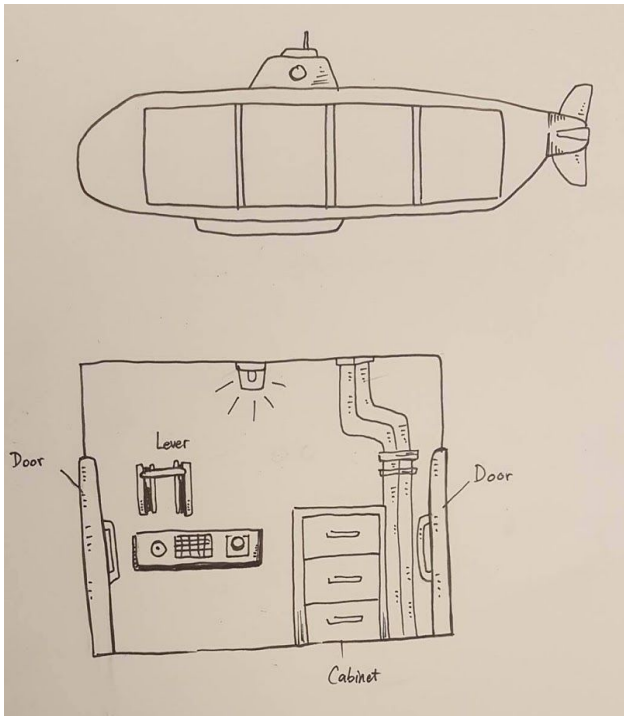
The player is thrown right into the action by beginning the game as their selected crew member, one of four on the submarine. Fleet Command has called on your sub to take part in the most recent naval engagement. You are utterly unaware of the broader state of the battle or war outside your submarine walls, just trusting that if you follow the orders coming through

Captain's radio you will survive. Your team's goal is to successfully complete the time-sensitive tasks assigned by Command to keep the submarine afloat.

2.3.2 Spaces/Worlds

Make sure you include sketches of the worlds and brief descriptions:

All four players will be in a submarine, each in a separate room: Captain's Room, Gunner's Room, Navigator's Room, and Engineer's Room. Details will vary based on the type of the rooms, but they will primarily have various interactive items such as levers and buttons, which are used to solve puzzles. The rooms will be covered with rusty metal textures in a dark, damp, and gloomily-lit environment.



2.3.3 Characters

Make sure you include sketches of the characters and brief descriptions of their personalities and capabilities.

As the players are never actually able to see each other, they cannot tell what their character looks like. As such, we don't have concept sketches of these characters. Instead, character selection will basically be grabbing and wearing some user-defining hat.

Captain: a grizzled, salty and bearded Yankee wearing a large cap and determined to see his crew through to their salvation. Mans a radio, depth indicator, heading indicator, and receives updates to the crew's tasks. Primarily responsible for relaying submarine status to the rest of the crew.

Gunner: a laboring, novice sailor wearing a typical boat cap and skilled in the art of picking up heavy things. The Gunner will periodically load the correct type of torpedo, aim to an appropriate bearing, and fire on Captain's orders. The Gunner's chamber also holds several

fuses which might need to be pulled in conjunction with Engineering. Periodically fires the game-ending nuclear torpedo.

Navigator: the Navigator essentially serves as the Captain's first mate and is a bespectacled, scholarly sort of fellow. The Navigator's chamber contains the rotors necessary to position the sub in the right direction. The Navigator also has a file cabinet containing lookup tables needed to decrypt special Captain orders and communicate the correct torpedo types to the Gunner.

Engineer: the Engineer is a black-lunged curmudgeon straight from a West Virginian coal mine. Of all the people displeased to be on the submarine, he is the most displeased. Wearing a hard hat with light, the Engineer will handle a multitude of valves to keep the maintain functionality of the submarine. The Engineer also controls the depth actuator, communicates impending engine failures to the crew, fixes generator failures to keep the lights on, and maintains speed of the submarine by regulating fuel.

2.3.4 Levels of Difficulty

How will you vary the level of difficulty

At the beginning the game play, the first user will be allowed to select a level of difficulty (easy, medium, hard). There will be a set number of tasks for each level of difficulty, and based on the level of difficulty that the user selects, the Fleet Command will assign more appropriate tasks.

2.4 Interaction Models

2.4.1 VR Interface - Navigation and Movement/Control

What does the game user interface look like and how does it work?

The VR interface is designed so that the user can walk in 1:1 scale around the submarine rooms. There will be bulkhead doors which trigger a teleport to adjacent compartments when needed. These doors will be clearly marked and glow when approached so the user knows they can be used.

2.4.2 Game Play Sequence and Levels

Show as a flowchart and provide descriptions of transitions.

We decline providing a flowchart because this is so simple. Each user begins in a lobby level--because we do not use any of the Unreal Engine's built-in sessions, these lobbies can be random/different for every player. Perhaps a dock scene, a bar scene, or a beach. In any such lobby, the player is able to communicate their character preference to the server. When all characters are chosen, every player is teleported to their room in the submarine and the game begins. The transition for this is a simple fade. Players are unable to leave their set of rooms, so this is where they remain the entire game. When a game is over, a fade transition returns the players to the lobby.

2.4.3 User/Environment – Obstacles and Props

How does the user interact with the environment. How will you handle collision detection? Physics? Object manipulation, etc. Be specific.

The rest of the interface is made to be intuitive--objects which the player interacts with will be visually distinct from inert submarine decor and will glow when looked at, so the player knows which parts of the environment are accessible. Collisions and interaction detection is handled using the Unreal Engine's built-in physics hit detection. For mouse/gamepad interaction, a raycast is used to trigger interaction with whatever object the player is looking at.

2.4.4 User/Character

How does the user interact with his character and other characters. Be specific.

Each player is placed into their own room (a level) and communication will be handled by the headsets and integration with Mumble. Within the room, the player is their character and interacts with them simply by existing and moving around. The player only interacts with other characters via the proxy of their voices coming from other submarine compartments. In a sense, the puzzles which require input from multiple characters is also a way for different characters to be interacting.

2.4.5 Character/Character

How do characters interact. How will you handle collision detection, AI, physics, etc. Be specific.

The characters only interact through their cooperation in solving puzzles. Actions taken by a character in one room, for example, will influence the state of the submarine for all others. Voice communication is another method for the players to interact with one another. There will be no direct confrontation between the players and they do not directly encounter one another.

2.4.6 Motion Tracking

If appropriate, describe the motion tracking features of your game

Using HTC Vive controllers, players will be able to use their own hands to interact with components of the submarine as if they were real. Locomotion is handled by walking around the tracking area in real life.

2.4.7 Multi-Player

If appropriate, describe the multi-player mode

Multiplayer will be hosted using a remote, dedicated Node.js/SocketIO server. The server will receive input events from the various players and return the state of the submarine. Additionally, the server will handle the success/failure of tasks. Because players do not interact with one another at all, there is no need to handle the calculation or replication of expensive, fast tasks such as movement or physics across the network. TCP will be performant enough.

2.4.8 Mobile

If appropriate, describe the mobile features of your game

No mobile features whatsoever.

2.4.9 Networked Play

If appropriate, describe networked play.

See the multiplayer section for a description of the network setup.

2.5 Performance and Scoring

2.5.1 State Variables

What are all the character, environment and gameplay variables necessary to save/restore or pause/resume the game or virtual world experience?

The game is meant to be played in short, multiplayer rounds of approximately five minutes in length. Therefore there is no way to pause, save, or restore the game or world. Within the game, however, state variables about the submarine such as heading, depth, speed, and number of failed tasks will be tracked by the server for handling game logic.

2.5.2 Feedback

What are the positive and negative feedback mechanisms you plan to employ and how do they affect the game play?

Positive feedback will include such effects as a “whoosing” sound when firing a torpedo, engine humming when the Engineer fixes things, confirmation from the Captain that a codeword exists when the Navigator decrypts something, and confirmation from Fleet Command that a task was successfully completed. Negative feedback will include such effects as all the lights in the submarine flickering with a dreadful groaning sound, distant explosion, slowly rising water, and indicators in the Captain’s room about how many tasks/how much damage the submarine sustained.

2.5.3 Performance and Progress Metrics

How will you monitor player progress. How do you win? How do you lose?

The Captain will have a clock in his room. The players win if the submarine is still intact after five minutes.

Game Ideas: In this section we record ideas for potential tasks to implement. These ideas would be worked into various scenarios presented to the submariners by the server. If the crew fails an order (or some configurable number of orders, strikes) the submarine is destroyed by enemy fire. The screen fades to black and the crew is returned to character selection and startup.

- Everyone pulls a lever in their room to get the water out of the ship.
- The Captain receives encrypted orders from command, “Operation X.” Navigator gets message from Captain regarding “Operation X.” Navigator then needs to open file cabinet to find out instructions required to decrypt and follow that order.
- Team is required to fire a certain type of torpedo in a certain direction at a certain time, i.e. the Captain will have the clock, depth, and heading indicators, the Navigator will position the sub in the right direction and use a lookup table of torpedo types to communicate the correct torpedo to the gunner, the Engineer will have to toggle a set of valves and levers to maintain the submarine’s speed and depth, and the Gunner will load the correct type of torpedo and fire on cue.
- Translate intercepted message from enemy using Navigator lookup tables, may force submarine crew to decide right course of action between fleet orders and new information.
- Periodically, lights go off. Engineer has to drop current task and begin work on fixing a generator to restore power.
    - Flashlight feature: when the lights go off the crew must work in the dark. Each crewmember is better or worse equipped to deal with the dark: the Engineer gets an actual flashlight, the Navigator must use a candle, the Gunner sees by the ambient light of the torpedo tube, and the Captain must use a lighter.
- Everyone on the submarine must stop and turn a special key in their room simultaneously to approve nuclear torpedo launch.

3 Game Design - Implementation Details

3.1 Design Assumptions

3.1.1 Hardware

Minimum hardware configuration (i.e. processor speed, memory requirement, graphics card, etc.) necessary to run game

- NVIDIA GTX970 (Graphics) with i5-4590 Processor (minimum specs for the HTC Vive)

3.1.2 Software

Version of Game Engine, Windows, DirectX, etc. required to run game

- UE 4.15, Mumble, and Socket IO

3.1.3 Algorithms and Techniques

What specific algorithms, plug-ins, animation techniques, etc. will you be using to implement your game. Be specific.

- We will be using a topological sort for the tasks and that they are all feasible for the crew team.
- Procedural techniques will be developed to provide variety in order of puzzles presented as well as the underlying components of each puzzle.

3.2 Storyboards

Show storyboard sketches of your game environment and play sequences here. This should convey the look and feel of the game as well as illustrate the game play.

In lieu of sketches we provide a dramatic scene.

The radio chatters and a despondent, pre-recorded female voice drones into Captain's ears.
RADIO: Operation Eye.
CAPTAIN, *to Gunner*: Encrypted order, Operation Eye!
GUNNER: Yes sir! Navigation, encrypted order. Operation Eye!
NAVIGATOR: Got it.
Amidst leaking pipes and groaning steel, Navigator digs through an old file cabinet. He sees four files labeled "EYE," "I", "AYE", and a pictogram of an eye.
NAVIGATOR: Which one is it?
ENGINEER: Which one is what?
NAVIGATOR: Not you-
GUNNER: How do you spell eye?

NAVIGATOR: I don't kn-
GUNNER: Not you-
CAPTAIN: E-Y-E.
GUNNER: E-Y-E.
Navigator retrieves the right file. The order reads, "Dive 30 meters."
NAVIGATOR: Dive 30 meters!
ENGINEER: Starting dive!
NAVIGATOR: Wait-
GUNNER: Dive 30 meters!
CAPTAIN: Copy, watching depth.
ENGINEER: Tell me when to stop!
NAVIGATOR: Don't dive yet!
GUNNER: Captain says begin dive!
NAVIGATOR: We already did!
The lights flicker as the submarine dives too deeply and breaks apart in the inky abyss.

3.3 Design Logic

3.3.1 FSM - State/Effect

Provide as much detail as possible as you can of your game logic in the form of a FSM graph. You can sketch this by hand or use Visio, whichever is easier.

We cannot detail in any significant state the current game logic at this time. Once we begin implementing more tasks, we can revisit this with a map of basic server communication from client in room.

3.3.2 User Solution/Actions

Describe how the player wins the game. Include descriptions of puzzles that must be solved. Provide as much detail as possible.

The initial lists of puzzles we intend to implement are detailed in the game ideas section. The player wins the game by solving the puzzles presented to it by the server without getting a configurable number of strikes in a configurable amount of time. Default will be 3 strikes in 5 minutes with approximately 15 tasks across the 5 minutes.

3.4 Software Versions

3.4.1 Alpha Version Features (vertical slice through total experience)

The alpha version(s) represents the first time the game is "playable". List the complete set of features to be included in the alpha version(s).

The alpha version will fulfill the requirements outlined for it by the full project requirement sheet. We intend for our minimally-viable game to serve as alpha. This demonstration will show

four clients in four different rooms, each room containing some different interactable objects, and a server which communicates with all four clients. The server will propose a task where each client must interact with some specific object in their room within some time frame and relay this task to the Captain. The completion of this single task will dictate a win or loss for the crew.

3.4.2 Beta Version Features

List the complete set of features to be included in the beta version

The beta version will build upon the alpha to feature implementation of many many more interesting tasks and server logic which presents these to the crew in series to result in an enjoyable game. Asset improvement is the other major feature of the beta.

For more detail on both alpha and beta, see the following Milestones subsection.

4 Work Plan

4.1 Tasks

List ALL the tasks and subtasks necessary to build the game application.

Provide separate descriptions of each task and subtask, which members of the group are assigned to it, and the expected task duration.

- **Networking**
    - Determining and communicating the health of the submarine by aggregating the status of player subtasks. (Tim).
    - Communication of tasks between different players. (Jason).
- **Voice-over Communication**
    - Selective communication with headsets using a Mumble server. (So trivial to implement. Tim).
- **Environment Design**
    - Room Designs
        - Captain's quarters (Tim).
        - Navigator's quarters (Ray).
        - Engineer's quarters (Jason).
        - Gunner's quarters (Kunal).
- **Puzzles**: Idea generation & Implementation.
    - (All in conjunction, each of us will implement some number of puzzles. Tim will implement the least due to networking overhead. Santi will lead).
- Submarine Health Framework:
    - Engine that takes in subtask statuses (completion, time) to determine health of the submarine and evaluates win / lose conditions (Kunal).
    - Feedback for Danger: Red lights flashing, ship vibrating / shaking (Ray).
- **Testing**
    - Making sure all components listed above are in order for demo (All).

4.2 Milestones

Alpha Version

Describe the functionality you plan to achieve and demonstrate in the version.

You will show, in class, the work you've done up to this point.

- While VR may not be functioning, the focus for the alpha version will be on the mechanics (locomotion and novel feature). We will try to have these done and have all the kinks worked out so you have a good foundation for your product--already we have a working version of a simulation for a single crew member.
- Submit a screenshot of the Unreal Profiler (or equivalent).
- Team problem solving: each team will bring forward one or two things they are finding hard or difficult, then the class will work together to solve them.
- A GitHub “Release” by this date.

Beta Versions

Describe the functionality you plan to achieve and demonstrate in the beta1 and beta2 versions.

- At this stage, our game experience will be playable/usable in VR.
- Our focus will be on the procedural content and C++ for this stage; we will have hopefully implemented several tasks with socket.io working to allow users to properly navigate multiplayer. Our levels and worlds should be populated with the procedural content.
- A GitHub “Release” by this date.

4.3 Development Schedule

Organize your work plan tasks in some kind of readable format and attach it to this document.

We plan to strictly follow the schedule of requirements as outlined in the main assignment document. More specifically, we propose the following milestones. As of writing this document, we have completed an initial alpha demonstration featuring a working player character, interactable objects, intuitive user interface, single room of the submarine, and communication with a server. To keep team morale and understanding of the project high, we will have (at least) a weekly playtest to work out bugs and new ideas. For every playtest, we will test using fully-packaged versions on live, remote servers. By each playtest we will have established some features:

3/25/17: Finish the game design document and run team through playable alpha.
4/1/17: Finish server communication necessary to drive minimally-viable game in VR.
4/8/17: Add more challenging tasks and finish procedural puzzle generation.
4/15/17: Begin polishing and improving assets.