CMPT370 – Group Meetings Document

Meeting 1 – Tuesday September 20th /16

- Dutchyn's expectations:
  - Progress made each week, whether it be documentation or code
  - Everyone speaks and has something to say
  - Requirements will probably change mid term
- Leadership Style:
  - Transformational
  - Trello (for SCRUM system)
- Group Expectations:
  - Everyone commits 10 -15 hours per week
  - If a group member cannot complete a task, notice must be given to the rest of the group at least two days before the due date
  - Group meetings are mandatory
  - Be prepared for group meetings
- Next meeting:
  - Saturday at 4:00, Spinks
  - Design docs
    - Nothing language or implementation specific should be included in design docs
  - Project expectations

Meeting 2 – Saturday September 24th /16

- Design model:
  - Model View Controller
  - Actors = computer robots, people
- Questions:
  - Is online networking feasible for the scope of the project?
  - Phone application or software application?
  - How difficult would it be to implement sound effects and music?
- Planning Model:
  - Spiral planning because we're a bunch of newbs
- Game Ideas:
  - AI programming:
    - Move in random direction towards the opposite target, when you hit the edge of the map change directions
    - Each player has a planned move which gets scrapped if new information is provided
    - Overarching method that looks for tanks. If a tank appears in it's view, it changes its plan
      - The longer the robot travels in a certain distance and doesn't see another robot, the more likely it is to change its direction

        - Robots should try and go to everyone's starting area. Depending on what turn it is, move out from the starting area
        - Include both an Aggressive AI and a defensive AI
            - Start with a basic AI
        - Add a chance that it will do something unpredictable (random behaviour)
        - AI should have a basic set of moves, to avoid going in circles or something dumb
        - Heuristic that it bases its next move on where a player is most likely to be
            - Certain tiles have higher values of probability
            - Calculate in separate program and implement as an array map
        - Search and destroy mode:
            - Keep a log of moves. If no tanks have seen any other tanks for x number of moves, have the AI switch into Search and Destroy mode where the tanks travel towards the middle of the map. Search and Destroy mode would be disabled as soon as the tanks see each other
            - Instead of a log, have a timer (i.e. duration 1 minute), after the given time has elapsed, tanks start searching for other players
            - Search and destroy could be activated after it's travelled across half the map
            - The longer the robot travels in a distance and doesn't see another robot, the more likely it is to change its direction
            - Direction change could be random
    - Watch tower square (in the middle) where, if a player is on it, the entire board can be seen
    - Once only two players remain, the entire map is revealed
    - If a certain number of turns have occurred (i.e. 50), and there's still 2 players left, reveal the entire map
    - Mouse and keyboard input only
    - Local multiplayer for now
    - Design for Windows OS

- Must Haves:
    - Some form of multiplayer
    - Map system (view)
    - Robots (controlled by AI)
    - Player controlled tanks
    - Win conditions
    - Game board
    - Menuing system
    - User input

- Would be nice:
    - Different tile sets, like obstacles or water tiles
        - Obstacles – can’t see passed but can shoot over
        - Water – takes longer to go through them, if turn ends and you’re in water, lose a health
    - Monuments that give upgrades to units (i.e. plus one to fire range or plus one to health)
    - Customizable tank
    - Custom robot
    - Obstruction tiles
    - New tanks, select tanks (i.e. all snipers, etc.)
    - Mirror tanks
    - Custom skin
    - Networking
    - Sounds

- Arianne:
    - Make pictures for:
        - start screen
        - menu

- Must Haves Priority Queue:
1. Basic game model
2. **Basic view**
    a. **Menu system**
    b. **Player view**
3. Player controls (including menu navigation)
4. Basic tank super class
5. Tank movement for single player
6. Basic multi-player
7. Basic AI for one tank
8. Observer class that can view the game
9. Multiple tank types
10. Varying tanks and strategies required for each tank
11. Ai that prevents stalemate

- Would be nice priority queue:
1. Sounds
2. Different behavior for each teams AI
3. Graphical animations (e.g. explosion animations)
4. Make the teams have some personality based on colour
5. Additional tanks
6. Alternate maps with new features

Meeting 3 – September 26th /16

Scenario = must have's for project

Document Overview:

1. Title page
2. Table of contents
3. General Overview
    a. Given requirements
    b. Game overview
4. Development Strategies
5. Architecture Overview
    a. Model
        i. Tank stats
        ii. Map model
        iii. Forth Interpreter
    b. Control
        i. AI
        ii. Control scheme
        iii. Window navigation (controls)
    c. View
        i. Map
        ii. Game board
            1. Tanks
            2. Tank stats
            3. Tiles
            4. Term stats
        iii. Windows navigation
        iv. Player stats
6. Delivery timeline
7. Executive Timeline

Diagrams:

- Architecture view with MVC
- Game board view
- Actors board
- Development strategy diagram (waterfall, spiral, etc.)

Meeting 4 – September 27th /16 @ 4:15pm

- Questions for the TA:
    - Once finished, can we deviate from our requirements document?
        - Yes, but not on the game rules or any provided documentation
    - Does the interpreter need to be included in the requirements document?

        - Yes, but don't go into any implementation detail
    - How many diagrams do we need for the game board?
        - Only what is necessary
- We are currently between meetings; division of labour is being organized.
    - All labour will be divided by this evening

Meeting 5 – September 27th /16 @ 9:00pm

- Design document outline is complete and this meeting is to discuss division of labour.
- Division of labour:
    - Arianne – sections 1 and 7, merging everyone's sections, editing of document
    - Sam –section 6, table of contents, title page
    - Kristof – section 3, help with section 4 diagrams
    - Mike – section 4, help with final editing
    - Chris – sections 2
- Due date for each section: Friday by midnight
- Every diagram must be accompanied by a detailed description (GIMP?)
- Posting assigned tasks in Trello
    - Once complete, each member should update in Trello and Gitlab/REQUIREMENTS branch

Meeting 7 – October 4th /16 @ 4:15pm

Agenda:

- No documents have been posted regarding the design phase but we've had discussions about the possible different areas
    - Is this where we design implementation?
        - High level AF
    - Will we need to write pseudo code and plan our algorithms?
        - No pseudo code
    - Should the interface be designed in its entirety? Should the design be identical to the final product?
        - Lower fidelity, not identical
    - How do we include our "nice-to-haves" in the design phase without being locked down?
        - Include class diagrams for nice-to-haves with labels
        - Or just include them and leave them out if there isn't time
- We've assigned rough areas to different group members
    - Arianne = graphics/interface design
        - Start-up screen
            - Instructions screen
            - Player-selection screen
            - Team profile screen

        - Game board screen
            - Game map
            - Overlay end of game stats
    - Mike = AI
        - Use messages to carry over the turns?
        - A*
        - Progressive a* to map out short distance and based on what happens in the short distance, change heuristic based on that.
    - Sam, Kristof, Chris = general brogramming and support staff
- Set up next meeting time to designate tasks
    - Wednesday at 5:00pm
- What was expected in terms of actors?
    - No specific instructions were given to the TA
    - Depends on the design of the system (i.e. client/server, etc.)
- We did our scenarios by putting actions inside the scenarios
    - A scenario is triggered by an action

Meeting 8 – October 11 /16 @ 4:15pm

- Without feedback for requirements doc, how do we proceed with the design doc?
    - If there are areas we aren't sure about, go to Dutchyn and ask
- When will the spec for the design doc be released?
    - Not sure
- UML diagrams specifics:
    - Include prototypes, private, public variables, member functions, etc.
    - Everything on a normal UML

Meeting 9 – October 16$^{/16}$ @ 8:30pm – 9:50pm (1 hour and 20 minutes)

Agenda:

- A design doc is a way for us to communicate and justify our design decisions
    - Should be a clear explanation of our intentions
- In the real world we would have 2 design documents:
  1) For peer developers
  2) For the client
    - Since our Requirements doc wasn't so hot, we should perhaps consider making two via documentation reuse
- Let's meet with Dutchyn this week and clarify that we're on the right track
- All aspect of design must be justified. If a clear justification cannot be made, cut it.
    - These justifications must be clearly explained
    - If a design decision gives rise to new/unexpected benefits, document them

    - If a design decision gives rise to new assumptions, document them
    - If a design decision gives rise to new risks, document them
    - Let’s sort out which aspects we can’t justify clearly and discuss them with Dutchyn
- A good design doc outlines a common vocabulary which can be used between developers and clients to discuss the project
- A design document should provide a very clear logical understanding of the system before any implementation is attempted. This includes a vision for the organization of all objects, entities, and their relationships
- Discuss the outline and insert the components of the requirements document
- If you take responsibility for a section, please do your best at this section. Editing shouldn’t mean rewriting the entire thing
- Everything should be proof read by the entire group before it’s considered finished

Meet Dutchyn this week, set up a time.

Have rough draft of Design Doc done by Wednesday

Possible Architectures:

- N-tier Architecture (for server inquiries):
    - Presentation tier
    - Logic tier
    - Data tier
- Component Based (for many smaller components)

High-Level Entities:

- Forth code with robot functions
    - Interface with Forth code
- UI (how the player interacts)
- Game board
- Player
- AI
- Robot Librarian
- Robot will need to store location

Questions:

- What specifically are we uploading and downloading in terms of robot information?
    - Are we simply uploading stats for the game?
    - Stats per team?
- Do we make a wrapper for forth code?
- Can we see a sample JSON file?
    - A function that parses JSON
- How much data does our system need to take from Forth and use within our own classes?

Meeting 10 – October 16/16 @ 1:30pm – 2:30pm

Understanding Forth:

- The purpose of Forth is to port our AI to other people's systems
    - Gives an interface to the AI
    - This allows is to share AI with other groups
- Download all of the robot functions as a JSON file at the start
    - Our way of downloading things
    - Robot functions in Forth, write our own versions in aspect J
- Forth is like a description for what the robot is doing, we parse that description, then we take that shit and run our own version of that function
- Forth is a standard, not a language that needs to be compiled
- Parse the string of the Forth functions
    - Call the functions in aspect J
- AI
    - From the perspective of one robot
    - Messaging between robots because if one robot out of the three can see another team's robot, it can tell another robot to move somewhere within the range of the other team's robot
    - Each robot has their own AI
    - Our AI is limited to the robot language but we can combine functions to come up with something new
- JSON:
    - For sharing AI's with other groups
    - Uploading robot stats
    - Uploading Forth functions

Architecture:

- MVC with two extra components for Forth and JSON (JSON is the robot librarian)
- Robot librarian is the interface to the server
    - Handling http/get/post/put
    - The game master calls the robot librarian and says get this robot, the robot librarian gets the robot from the webpage
    - At the start, download the stats for each robot
    - Throughout the process, store the updated stats
    - At the end, upload the new stats for each robot
    - Also downloads the Forth functions:
        - Forth functions are mirrored in our Aspect J code

Questions:

- How do we make an interpreter when we don't know the interpreter's role?

Meeting 11 – October 16 /16 @ 5:30pm – 7:30pm

- Discussing high-level components of our system
- Brainstorming on chalk board for connections between components
- Chose two architectures: Model-View-Controller + Connected Components
    - Model-View-Controller + Forth Interpreter + JSON interpreter
    - Split up the high level components
    - Still unsure of how certain components will interact with the system
- Model Component:
    - Stats Logger
    - Robot Model
    - Player Model
    - Team Model
    - Game Board
- View Component:
    - Start Screen
    - Instruction Screen
    - Player Selection Screen
    - Robot Selection Screen
    - Game Screen
- Controller Component:
    - Start Controller
        - Sets up initial game and passes control to Game Controller
    - Game Controller
        - Robot controller
        - Input controller
        - AI (Forth functions)
        - Aspect J functions
- Forth Interpreter Component
- JSON Interpreter Component

Meeting 12 – October 17 /16 @ 4:15pm – 4:25pm with TA

Agenda:

- Questions for Jordan (the TA):
    - The forth language is only used as a standard of communication between our system and other group's functions?
        - yes
    - Aspect J will take parsed strings from Forth and call functions in Aspect J?
        - yes
    - Should we streamline our input from a human player and a computer AI from the same place?
        - Yes
    - Is there a component in the system that has knowledge of all of the pieces?

        - Ask Dutchyn
    - Should we include state transition diagrams?
        - Yes, if we want to
    - Do you have any recommendations for going through the design process?
        - Answer the following questions:
            - What are the models?
            - What are the views?
            - What are the controllers?
        - Include return types and certain data structures that are being used
    - What if one of our classes from the Design Doc needs to be split come implementation time?
        - That's fine.