Department of Electrical and Computer Engineering - University of Auckland

COMPSYS 302 S1 2016 - Project (45%+45%)

Background

You and your partner have decided to take your engineering skills and create a start-up software development business. Clients can come to you with ideas, and you make it happen. Your philosophy is that nothing is impossible – if a client can dream it, you can build it (as long as they'll pay for it!). On the first day, a client comes in the door, explaining that he wants a computer game that his eight year old son can play against his friends, but he is paranoid about big corporations building profiles about his children and selling it to other companies. Therefore, he wants to either be able to retain control over the data, or ensure that no one has control over the data. You have a think and ask a university lecturer for some advice, who suggests that a peer-to-peer (P2P) system could be a suitable approach.

Objective

The overall goal of this project is to design a peer-to-peer online multiplayer game. There are two main parts to this project:

1. Designing an offline game as a Java application that takes input from users and responds accordingly, and developing a game AI to play against the user.
2. Designing a P2P protocol and web interface using Python that allows users to play against each other over the internet.

The goal of the project is to design a simple game as a Java application that takes input from users and responds accordingly, and then develop a Python webserver that enables internet-based multiplayer of the game. The purpose of this project is to learn Object-Oriented Programming, Threading, Graphical User Interface (GUI), Artificial Intelligence, Networking, Web Development, Security, and Software Project Planning & Development. Languages learnt are Java, Python, and HTML/CSS. The project is completed in pairs. The project will be completed and tested using Ubuntu.

Game Description

The aim is to create a game similar to Combat, a game originally manufactured and sold by Atari in 1977. The aim of the game is to destroy the opponent tank as many times as possible within two minutes by shooting enemy tanks with bullets.

Image from Wikimedia

Game Specifications

The input of the game is through one keyboard. The display of the game is a Java application window shown on the monitor. In the game, each window boundary (top, bottom, left, and right) acts as permanent walls and no objects should exceed these limits.

The game must conform to the following minimum specifications:

1. A welcome screen should be presented, allowing the user to select a game mode using the keyboard. The valid game modes should be: training (enemy tank is stationary but appears at different locations), single player (enemy tank controlled by AI), and local multiplayer (enemy tank is also controlled using the keyboard).
2. When the game is started, the two tanks should be stationery in their starting positions. A countdown timer (from 3) should indicate to the user when the game starts. Before then, the tanks should not be able to move.
3. Objects must not be able to exceed the 1024x768 boundaries of the window.
4. When a shot collides with an enemy, a 'hit' should be registered on that enemy. A point should be awarded to the shooting tank, and the positions reset.
5. There must be some mechanism for keeping track of the score and the time remaining, and showing this on the screen.
6. During normal gameplay, the 'p' key should pause and resume the game. If the user hits 'Esc' (Escape), then they should be able to exit the game (a confirmation screen may be appropriate).
7. After the end of the two minutes, there should be an exit screen summarising the results and congratulating the winner. The user should then be able to return to the welcome screen and start again. To help with testing, pushing Page Down (PgDn) should skip the game and go straight to the exit screen.
8. Appropriate sounds must be played when shots are fired by the player, fired by an enemy, and when a shot hits something.
9. A logfile for each game should be generated, with timestamped details of each keyboard action.

The developers (as a class) will also collectively decide on some additional **rules** for the games. All games developed must follow these rules, since part of the second half of this project is dependent on the game conforming to these rules. There will be protocol discussion sessions held during lecture times throughout the semester as the class iteratively develops the protocol together.

Design Elements

This is where you have the opportunity to differentiate your project from others. Object-oriented design is compulsory (especially because the second half of the project depends on this). You should aim to include some aspect of threading (and this will be worth marks), but it should be implemented intelligently – don't just use threading for the sake of it, use it where it will help. **Do not expect to implement all of these elements.** It is up to you to decide what you want to try to design and implement given the time constraints presented.

As a general guide, meeting the minimum specifications will get you around 50% of the marks. The other 50% is for good design of the below elements in order to meet the objectives of making the game playable, intuitive, and fun. Full marks are extremely rare to achieve and may only be granted for projects which perform exceptionally well on all

minimum specifications and design elements. Clients in the real world are rarely 100% pleased!

Gameplay
The minimum specifications leave some gameplay elements to the designers, for example, the maximum speed of the tanks, how the tanks are controlled, maximum shooting rate, etc. These are the designer's choice, but should be implemented in a balanced way that keeps the game fun! Keep in mind that in order to make the game compatible with other games in the second half of the project, there will be certain rules decided upon by the class, which may affect some gameplay elements.

Graphics
We want the game to look modern if possible rather than having 1980-style graphics. The original Combat is a starting point – we want you to design a game that looks much more aesthetically pleasing and exciting for the user.

Welcome and Exit Screens
While you need to design a basic welcome and exit screen, you could develop a more sophisticated GUI for the welcome and exit screens. You should determine what information needs to be displayed to the user, and what buttons are needed for the user to complete required actions. You could also try to make it aesthetically or graphically appealing. While it is not necessary to have a help screen or an options screen, if you think that these are useful they can be designed and implemented.

Difficulty Levels
As part of the project, you must develop an AI that can play the game against a human user. At a minimum, there should be one setting for the AI, but you could introduce different difficulty levels, as long as the AI doesn't cheat!

Power-ups
In the original Combat, the tanks moved around and shot each other and that was that. We want to make things a bit more interesting than that! Perhaps power-ups could randomly appear on the screen to be picked up by the tanks. It is up to you to design various events that could occur in a balanced manner in order to make the game more exciting. Some examples could include:

- A shield that absorbs one shot
- Making the tank move faster or slower
- Making the tank shoot faster or slower
- Any other ideas you might have!

Level Layout
In the original Combat, there were only a few level layouts available and it was largely random which layout you would get in any given game. We want to make it a little more difficult. There are two options here:

- Using many levels of fixed design (i.e. the walls and tanks are placed in fixed positions), and either allowing the player to choose which level they want to play or randomly selecting one.

- Procedurally generate each level so that the player gets a “random” level each time they play the game. There will have to be certain limits such as ensuring that walls are in sensible places, the tanks don’t start too close to each other, and so on. The game should therefore have an infinite number of level layouts. Importantly, the processing and generation of the levels must not affect the gameplay – we do not want the game to become ‘laggy’ or for there to be long waits when generating the levels. This option is significantly more difficult than the first option, so the clients will be more impressed if you choose to implement this.

Other Options/Ideas

- A high score screen (and a mechanism for storing those high scores).
- Allowing users to enter their name, and showing those names on the screen.
- Making the computer play against itself in an AI DEMO mode.
- Suggest other options to the lecturers and/or TAs!

Peer-to-Peer Networking

Making the game is only half of what the client asked for! The goal of the second part of the project is to allow users to play the game against each other over the internet, using direct P2P connections instead of the more common central server model. The intention is that the users do not necessarily have to be using the same game client (i.e. they can be playing different versions of the game), but there will be a common communication protocol that allows the game clients to report game information such as the position of the tanks and when shots are fired to each other and update the screens locally. The protocol and requirements are to be developed collectively by the class as part of the second half of this project. As part of this, groups will be invited to submit protocol proposals (to be discussed further in class). Groups will likely have to retrofit their games to comply with a class-developed protocol.

Web Interface

As part of the second half of the project, students will build a web interface that allows a user to see who else is online, and challenge them to a match. Finding out who else is online should also be done in a peer-to-peer way. The protocol and requirements for this part of the project are also to be developed collectively by the class.

Versioning

The use of the Git versioning system via BitBucket is compulsory and will be monitored (a lecture on how to use Git will be presented). Steady progress from all groups is expected throughout the semester. Students should create their own private repository at the beginning of semester, and invite a teaching staff account to access their repository.

Assessment

This project is completed and assessed in groups of two. We do not want to see any identical games. We will be checking for code plagiarism, and if any plagiarism is found then we will not hesitate to fail you and report you to the University Senate. There are three assessment phases (more details of the specific requirements will be explained in lectures):

1. **Project Plan (0%) – [Due 5pm Friday Week 2]**

Before substantive development of the game can begin, each group will need to submit a project plan, the substantive part of which should not exceed two pages. It is not worth any marks but you must submit one to the client by the due date or fail the course. If there are any issues you may be called in for an interview to discuss progress with the lecturer. It should include:

- Brief background about the developers (e.g. previous relevant experience)
- An outline of the system to be implemented
- What you expect your product to be able to do (features)
- Provisional schedule (when you expect to do things, interim milestones)
- Foreseen challenges and how you plan to overcome them

2. **Game Prototype Demo + Interim Report (45%) – [Due 10am Friday Week 7]**
   In addition to submitting code, at this first demo, as a **group** you will present your single-player game (vs. AI) and local multiplayer game, and have an opportunity to discuss the extra design features that you have implemented into the game. Your design should be documented in an interim report, the substantive part of which should be no longer than 4 pages (font size no smaller than 10). Submission is via Bitbucket – tag the appropriate commit with PHASE-1-DELIVERABLES.

3. **Final Project Demo + Final Report (45%) – [Due 10am Friday Week 13]**
   On the final demo day (in addition to submitting the code), there will be **individual** interviews with each student, where each student individually presents the game and web interface, and discusses the networking elements of the project. There will also optionally be a small tournament that students can compete in. Your design should be documented in a final report, the substantive part of which should be no longer than 8 pages (font size no smaller than 10). Submission is via Bitbucket – tag the appropriate commit with PHASE-2-DELIVERABLES.

As this is a group project, there will also be a compulsory peer review (details to be given later). Additionally, in your final project demo and reports, your group should explain what the individual contributions were (i.e. who did what). Individual grades may be adjusted based on peer review and our observations.

Reports

In each report, we recommend that you include some of the following elements:

- How the developed system meets the requirements
- A top-level view of how the system works (diagrams help)
- One or two significant issues during development and how they were overcome
- Features that improve functionality of the system
- A comment on the suitability of the tools (e.g. Java, Python) for the application
- Suggested improvements for future development

A high standard is expected for these reports, including in the presentation of the reports. Feel free to include appendices (not included in page limit) if you have class diagrams, flowcharts, etc. that help aid the explanation of the system. A template will be provided.

Final Note

The project is intentionally open-ended in nature, so there is a lot of room for individual design and opportunities for you to make the application your own. **Do not leave things to the last minute**. Do not mistake the minimum requirements as the only requirements; if you

want to do well in this course you will need to design your application carefully and manage your time well. Do not hesitate to contact the lecturers and TAs if you are in doubt! They are available to help with concepts, troubleshooting, and debugging, but they will not write your project for you.

Academic Integrity Notice

The University of Auckland will not tolerate cheating, or assisting others to cheat, and views cheating in coursework as a serious offence. The work that a student submits for grading must be the student's own work, reflecting his or her learning. Where work from other sources is used, it must be properly acknowledged and referenced. This requirement also applies to sources on the world-wide web. **Do not copy code from other students or the internet.**