HCI Evaluation

Part 1: Qualitative

Ruzanna Chitchyan, Jon Bird, Pete Bennett
TAs: Alex Elwood, Alex Cockrean, Casper Wang

Today’s Workshop

- Think Aloud (~40mins)
- Heuristic Evaluation (~40mins)
- Homework:
    - Write up today's evaluations on your repo.
    - Start prepping for further evaluation.
    - Add two difficulty levels to your game

Think Aloud

- Plan Study (15mins)
    - Select a **facilitator** and two **observers**.
    - Choose **two short tasks** for the participant to do.
- Recruit **one participant** (from the group next to you! (5mins)
- Carry out study (10mins)
    - Explain tasks.
    - Remind participant to **think aloud**!
    - Observers record **critical moments.**
- Combine Data from observers. (10mins)
- HOMEWORK:
    - Do the analysis.
    - Add to your repo!

Heuristic Evaluation

- Select a **facilitator** to run the study and make sure the game is running.
- Select an **observer/expert** from another team.
- Facilitator asks observer to run through the game to familiarise themselves (10mins)
- Observer goes through in their own time to record any usability issues using the form provided (10mins)
- Either:
  - Run a second one in parallel on a second machine with a second facilitator and observer. Or…
  - Run a second one after the first!
- HOMEWORK:
  - As a group, analyse the experts' feedback and identify issues to address going forward.
  - Update your repo/report with findings!

Heuristic Evaluation

Heuristics (Nielson: http://www.nngroup.com/articles/ten-usability-heuristics/)

1. **Visibility of system status**
   - The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
2. **Match between system and the real world**
   - The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
3. **User control and freedom**
   - Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
4. **Consistency and standards**
   - Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
5. **Error prevention**
   - Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.
6. **Recognition rather than recall**
   - Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
7. **Flexibility and efficiency of use**
   - Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
8. **Aesthetic and minimalist design**
   - Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
9. **Help users recognize, diagnose, and recover from errors**
   - Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
10. **Help and documentation**
    - Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Heuristic Evaluation

Severity Ratings (Nielson: http://www.nngroup.com/articles/how-to-rate-the-severity-of-usability-problems/)

The severity of a usability problem is a combination of three factors:

- The **frequency** with which the problem occurs: Is it common or rare?
- The **impact** of the problem if it occurs: Will it be easy or difficult for the users to overcome?
- The **persistence** of the problem: Is it a one-time problem that users can overcome once they know about it or will users repeatedly be bothered by the problem?

The following 0 to 4 rating scale can be used to rate the severity of usability problems:

0 = **I don't agree** that this is a usability problem at all
1 = **Cosmetic problem only**: need not be fixed unless extra time is available on project
2 = **Minor usability problem**: fixing this should be given low priority
3 = **Major usability problem**: important to fix, so should be given high priority
4 = **Usability catastrophe**: imperative to fix this before product can be released

Heuristic Evaluation

Name:_____ Project Title:_____

Interface	Issue	Heuristic(s)	Frequency 0 (rare) to 4 (common)	Impact 0 (easy) to difficult (4)	Persistence 0 (once) to 4 (repeated)	**Severity** = Sum Total of F+I+P /3

homework / groupwork

- Add **two difficulty levels** to your game
    - e.g., change speed, make level harder, make enemies tougher, remove power-ups
    - make sure that players can select between the difficulty levels
- Write up the findings of today's user evaluations and add to your github repo
- Plan a more in-depth qualitative evaluation (please test on other people in the unit)