Choosing a project for your 1st Code-Review

You can choose one of the options below or pick your own idea. You should coordinate with your teacher about your project and fix its approximate functionality and ways to get extra points before you start working on a project.

The aim of the 1st code review is to learn how to write good and readable code and also to have fun implementing interesting tasks and ideas.

Deadline for the first working version of your code - **October 31th, 23:59**.
“First working” version means your teacher is able to execute your code without getting any errors. Your code may be not fully functional but it must work with no errors, so you are recommended not to wait until the deadline to send your code for review so you could elucidate any troubles while you still have time to solve them.
Remember: if a program runs perfectly on your computer it still can fail to run on another computer.

Steps to pass the project:

1) Create an empty private repository on github.com;
2) Make initial commit into master;
3) Add your teacher as a collaborator of the repository;
4) Create “dev” branch from master and work in dev from now on;
5) Write first working version of your project and push the code into dev;
5) Create a pull request from dev into master. In the description write instructions - what your code can do and how to execute it;
6) Your teacher has **one week** to review your code and send remarks;
7) You have **one week** to fix your mistakes and/or to improve functionality;
8) Repeat steps 5)-7) until your project is accepted.

1st Code-Review

Shell

Your shell should be a program with partial functionality of sh, bash or cmd. At the start, the program displays a prompt and waits for user's input.

Basic functionality

- *ls*. List the files and directories for the current directory.
- *pwd*. Print the full path for the current directory.
- *cd <path>*. Change directory.
- *cp <filename> <filename>*. Copy file.
- *mv <filename> <filename>*. Move file.
- *rm <filename>*. Remove file.
- *rmdir <dirname>*. Remove directory if it's empty.
- *mkdir <dirname>*. Create a directory if it did not exist.

All paths can be absolute or relative. The program should check the ability to perform operations (the existence and non-existence of files and folders), and report errors if they occur.

Extra points

- Running executable files
  run <executable filepath> <arguments>. Run the file, print its output stdout and stderr and wait for completion before accepting new input.
  Sample:
  $ run echo 'hello'
  hello
- Pipes
  Ability to specify several executable files with arguments separated by "|". In this case, the output of the first program is transmitted to the input of the second. The output of the last command and the entire stderr are printed.
  Sample:
  $ run cat my_file.txt | wc -l
  10

Console text editor

Basic functionality

- Ability to edit file;
- Text navigation using arrows;
- Hotkeys to save and exit.

Extra points

- Text selection;
- Clipboard;
- Text highlighting;
- Moving the cursor by words and to the beginning/end of the line;
- Removing words and lines;
- Find and replace;
- Menu and UI.

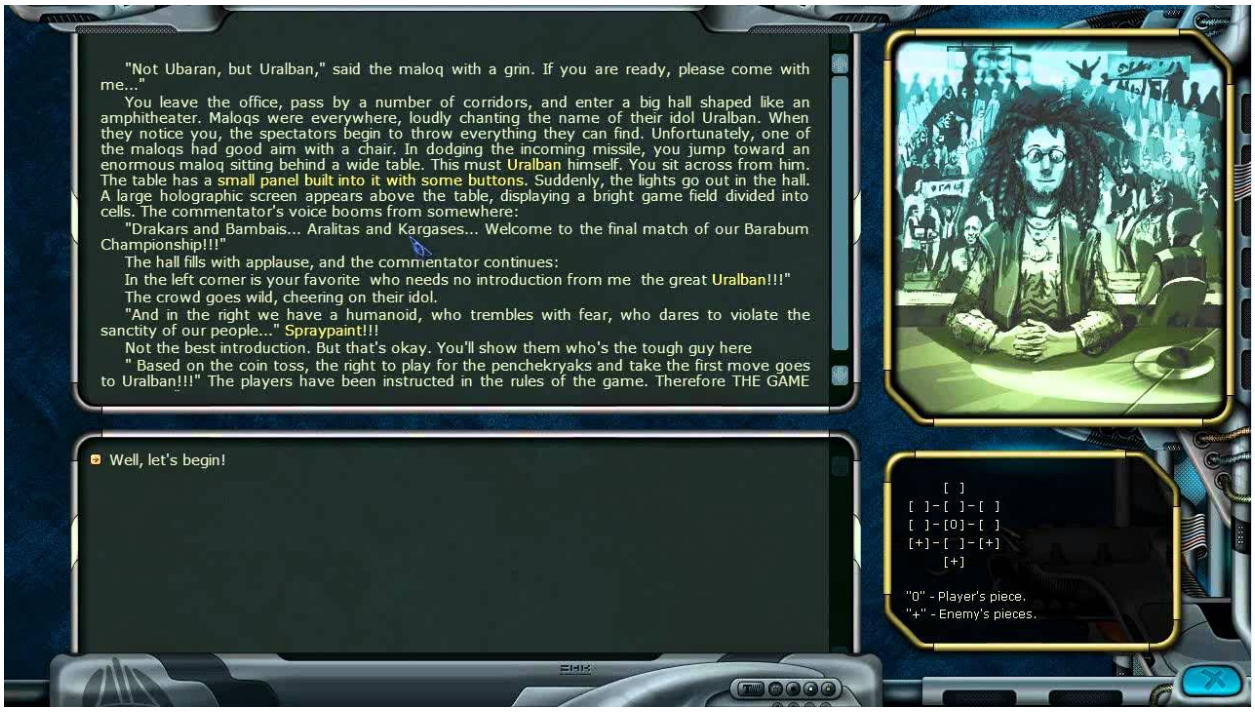
Interactive fiction (Text quest)

Sources of inspiration

- https://en.wikipedia.org/wiki/Interactive_fiction
- https://en.wikipedia.org/wiki/Visual_novel
- http://www.lysator.liu.se/mud/questdesign.html
- https://github.com/textadventures/quest
- https://renpy.org/

Basic functionality

- Interface: description of location + set of possible actions + character status. Example (you need the same 3 blocks, but in the console without pictures):



- Development of a script format that should contain:
  - Locations and their descriptions
  - Description of transitions and their required conditions
- Loading and playing quests from script files. The path to the script file is specified by the command line argument
- Writing several different scripts for demonstration

Extra points

- GUI for quests or for script editor
- Replacing options with commands in natural English

Maze generator

Sources of inspiration

- https://en.wikipedia.org/wiki/Maze_generation_algorithm
- http://www.astrolog.org/labyrnth/algrithm.htm
- https://franciscouzo.github.io/maze/
- https://www.algosome.com/articles/maze-generation-depth-first.html
- http://www.neocomputer.org/projects/eller.html
- https://fizzbuzzer.com/simple-maze-generating-algorithm/

Basic functionality

- Generation using DFS or a minimum spanning tree (support both options). The generation option is selected using the command line argument;
- Display mazes in the console using special characters;
- Saving/loading mazes to/from files;
- Maze solution and displaying the path.

Extra points

- More complex algorithms;
- GUI;
- Allowing user to solve a maze by himself.

Tamagotchi

Sources of inspiration

- www.programsinformationpeople.org/runestone/static/publicPIP/Classes/Tamagotchi.html
- https://medium.com/@it.belkin/opentadpole-the-first-cybernetic-animal-1dcba8a01e9d
- http://www.stager.org/articles/tamagotchi.html

Basic functionality

- Pet creation;
- Displaying its status;
- A set of actions that affect the condition of the pet: feed, entertain, train, etc.;
- The pet changes its state, even when the user does nothing.

Extra points

- GUI;
- Real event loop.

Encryption

Sources of inspiration

- https://en.wikipedia.org/wiki/The_Code_Book
- https://en.wikipedia.org/wiki/Caesar_cipher
- https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher
- https://en.wikipedia.org/wiki/One-time_pad
- https://en.wikipedia.org/wiki/Steganography
- https://en.wikipedia.org/wiki/BMP_file_format

Basic functionality

- Several operating modes (see below). The mode is selected using command line arguments;
- Encryption and decryption of files with Caesar, Vigenere and Vernam ciphers. File paths should be passed as command line arguments;
- Automatic Caesar cipher cracking using frequency analysis methods.

Extra points

- Implementation of any more complex encryption algorithms;
- Steganography: embedding and extracting text to/from a bmp format image using the last bit (or last few bits) of each pixel byte;
- Steganography over jpg or png;
- GUI.

Clicker

Sources of inspiration

- http://cookieclickergame.com/?play
- http://www.decisionproblem.com/paperclips/

Basic functionality

- Clicking by pressing a spacebar;
- Some points are given per click;
- Points can be spent for purchasing various autoclickers or other various improvements.

Extra points

- GUI;
- Real event loop.

Typing tutor

Sources of inspiration

- https://www.speedtypingonline.com/typing-tutor
- https://play.typeracer.com/

Basic functionality

- Interface: a line that a person needs to type and an input field. Prevent typing incorrect characters;
- Design a format for tasks and upload files in this format;
- Count errors and typing speed, save statistics among runs.

Extra points

- GUI;
- Display statistics while typing in real-time;
- Composing heatmaps for keys on which a person most often makes mistakes.