Software Studio Projects

Igor Wojnicki

2021-10-07

1 Software Studio II, eng., 5 sem., 2021

1. All projects are GNU GPL 3.0.
2. Each team has to propose:
   (a) software design and development methodology (classical or agile),
   (b) proper tools for project management and communication.
3. Each team has to keep track of who is responsible for what.
4. All detailed requirements have to be acquired from instructor and written down in the docs.
5. Development has to be aligned with instructor, discussed, approved, etc.
6. Both the development process and the final result are graded.

Docs have to be an integral part of the software repository and editable without using specialized tools (no PDF, DOCX etc. allowed) and accessible via web. If there are multiple files they should be interconnected with hyperlinks. The docs have to cover:

1. requirements,
2. development plan (what, who, when, why),
3. fulfilment of requirements, or detailed explanation why the are not met,
4. installation/deployment instructions,
5. user's manual,
6. team member responsibilities.

ITEM	PEOPLE
DokuWiki LCARS theme	2
Whiteboard	2
A*W Python library	2
OCR UI	2
Map data	2

1.1 DokuWiki LCARS theme

Gather guidelines and ideas behind LCARS, start at: `https://en.wikipedia.org/wiki/LCARS` Implement a DokiWiki theme: `https://www.dokuwiki.org` Design UI/UX accordingly. Color coding matters, you are not building 24th century spaceship UI, henceDokuWiki requirements might differ, however make it logical and concise.

Number of students: 2

1.2 Whiteboard

Make a JS library that minimizes a lag while interactivelly drawing on a HTML canvas. Check out the example: `https://developer.mozilla.org/en-US/docs/Web/API/Element/mousemove_event#examples` While dragging the mouse you should see that the trail is drawn behind the mouse cursor, not at it. The faster you draw the line the more it lags.

Build an algorithm that tries to predict where the trail should be based on past track, implement it and test.

Number of students: 2

1.3 A*W Python library

Papers to read: paper 1 paper 2 .

N-dimensional route planning which compensates for plan deviation. There are the following steps of the algorithm:

1. make A* plan,
2. identify collision areas/deviation subspaces,
3. make Wavefront plan for each deviation subspace.

Consider von Neumann and Moore neighborhoods.

Number of students: 2

1.4 OCR UI

A web application being a web interactive interface to Tesseract OCR. A two-pane approach with the scanned page on left, recognized text on right. Store both the OCRed text, the corrected version along with metadata (where the text came from, bbox etc.). Hint: use hOCR output (`https://en.wikipedia.org/wiki/HOCR`).

Number of students: 2

1.5 Map data

Make a web application that can visualize and interpolate geolocated time series data. You might consider using open layers or leaflet. Map performance is an important factor, esp. number of markers (from datapoints) to display, gradients etc. Additional requirements:

- ability to time travel,
- data interpolation (with arbitrary grid resolution, forming additional datapoints which hold interpolated values).

A test dataset (JSON/csv) is available.

Number of students: 2