Interim review – report:

[https://moodle.royalholloway.ac.uk/course/view.php?id=2125]

(must be submitted by 23:59 on Thursday, 1st December 2016)

Must document the structure of your submission directory, possibly in a short appendix

Normally about 5,000 words

Contents:

* Aims, objectives and literature survey;
* Planning and time-scale;
  + Notes for time scale:
    - In the ‘early deliverables’ section I initially planned to have finished my “first running version [of the game] with simple game rules” by the 14th October and, consequently, had planned to finish my next deliverable of “allow[ing] players to place towers and win or lose on one map” by the 22nd October. However, due to a number of set-backs (most notably the death of laptop and thus being unable to access my work on the hard drive until the data was recovered) I was unable to finish these early deliverables until [date of deliverable completion]. However, this did not have any major impact on my other deliverables, as I had given myself a considerable gap between this initially predicted completion date and the beginning dates for my next deliverables.
* Summary of completed work;
* Bibliography and citations;
* Some form of diary;

Contents specific to

my project:

* Design pattern and code organisation [Nov 14th – Nov 17th]
  + Design patterns
  + TD\_Game class is broken down within a single class into various functions (methods?):
    - *Initial global variable declarations etc:*  
      The initial lines of code in the TD\_Game class represents the declaration of fundamental global variables required for the initial set up of the game. These variables are declared here at the beginning of the class mostly due to them currently being hard-coded variables or simply temporary variables which are in place while the class’ structure is still in development.
    - *text\_objects:*

This simple function takes in a string variable which is the ‘text’ which will be displayed on the screen, along with a ‘font’ variable which is simply the pygame “font.font” function containing the font-type file with which I want to use and the size at which I wish the type-face to be rendered. In the function, a variable is declared, containing the pygame “font.rend” function which takes in the previously mentioned ‘text’ variable, True, and the colour in which the text will be rendered.

Once this variable is set, this newly declared variable is returned, along with the result of said variable with the ‘get\_rect()’ python function initiated.

* + - *Button:*

The following function afterwards takes in 7 or 8 arguments, and defines one variable if said final argument is empty, at it’s initialisation(?). The first variable is, same as with the text\_objects function, the ‘message’ that will be contained within the button. The arguments after that are simply the ‘x’ and ‘y’ coordinates at which the button will be generated and the width and height of said button. The next two arguments taken in are the ‘inactive colour’ and ‘active colour’ of the button, meaning the colours that the button is when the mouse cursor is hovering directly over it or not. The final argument relates to the ‘action’ that should be taken when the button is “pressed”, specifically which method should be called to.  
…

* + - *intro\_menu (method)*
    - *main (method)*
    - *drawMap*
    - *drawCreep*
    - *pygame\_quit:*

The ‘pygame\_quit’ method simply calls to the “quit” pygame function, effectively closing all pygame libraries, and subsequently calls to the “exit” function from within the sys library, which thereby closes the program.

* Principles of free software and open-source code constraints [Nov 18th – Nov 19th]