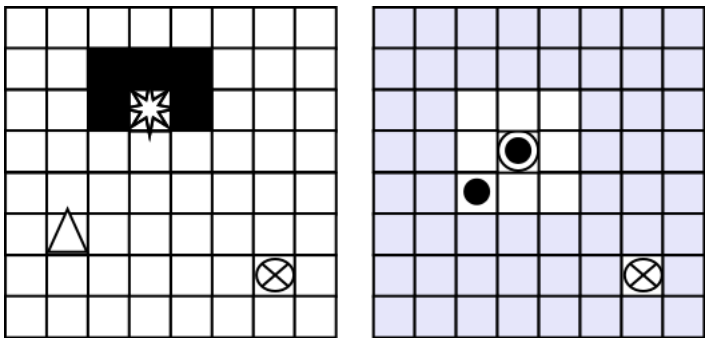
Practical 8 (due 2021-05-14 @ 09:00)

The purpose of this assignment is to introduce the type of problems presented during the semester test 2 and exam. 2020 ST2 SSA.

Please note: Submissions will be checked for originality. If you use someone else's code or code is taken from the Internet, then your prac will come under scrutiny for a potential copy, which may result in zero marks being awarded along with potential further disciplinary action.

A Trip to the Dark

The Utopian Electrical Supply Commission has been struggling to maintain a stable power supply ever since the onset of "The Event". In contrast to it's previous periods of load shedding there are now random spikes in the power supply which cause electrical equipment to fail. To combat this, rather than fix the underlying problems, the commission has asked you create an educational game themed around restoring power to a two story house. It is worth noting that the commission has resisted modernising domestic electrical systems so Utopian houses must be reset using a fuse rather than a more modern trip switch.



Player (Double circle), Visible space (empty squares), Fuse (black circles), Torch cabinet (triangle), Darkness (blue-grey squares), Fuse box (star), Walls (black squares), and Stairwells (crossed circle)

In the game you will need to move a player controlled character around a pair of linked two-dimensional playing areas. Your logic must be placed in the `DarkSpace` **`namespace`**.

Initialisation:

- The size of the environment as well as the numbers of both stairwells and fuses are specified via command line arguments.
- There are two environments of equal size namely the ground floor and basement.
- The fuse box appears in a random location at least 3 squares away from the edges of the game world. It is surrounded by impassable walls on 5 sides as per the image.
- The torch cabinet appears in a random location in the last three rows of the ground floor. It may not appear in the same column as the fuse box.
- The stairwells are randomly placed in the same positions in both the ground floor and basement. They may not appear in the same column as the fuse box.
- The player starts in a random location on the ground floor.
- They player may be holding a torch and any number of fuses. The player starts out with neither.
- The fuses are placed randomly in the basement.

/..Continued on the next page

Movement:

- The player may move one step in each direction (up, down, left and right). The player may not move outside of the game area.
- The player may not move into a wall
- The player may occupy the same squares as the stairwells. While the player does so they may climb the stairs as a movement option. This moves them into the same location in the other floor. If the player goes downstairs without a torch they trip and the game is over.
- If the player moves into the same location as the torch cabinet they begin holding a torch if they were not already doing so (the player does not occupy the square afterwards).
- When downstairs the player can only see in a 1 square radius around themselves.
- If a player moves over a fuse they pick it up and occupy the square afterwards.
- If a player is holding at least one fuse and walks into the square occupied by the fuse box the game ends in victory.

End-game:

- The game ends in failure if
  - The player moves downstairs without a torch
- The game ends in victory if
  - The player brings a fuse to the fuse box

Design

The design must model the initialisation of the two floors.. Make use of the design template provided in P01. Make sure to save your design as a PDF document.

Please format your practical for submission according to the structure established in the previous practical (`Docs` directory containing design, `Bin` directory containing executable files, `Source` directory containing `*.cpp, *.h` files and your `.cpb` file). The submission archive must be named according to the convention established in the previous practicals.

Please note: Programs that do not compile will be capped at 40%

The marksheet is on the next page.

UNIVERSITY OF JOHANNESBURG

Mark sheet		
Competency	*Description*	*Result*
C0	Program Design	/10
C1	Boiler plate code • Standard namespace (1) • System library inclusion (3) • Indication of successful termination of program (1)	/5
C2	Coding style • Naming of variables (1) • Indentation (1) • Use of comments (1) • Use of named constants (1) • Program compiles without issuing warnings (1)	/5
C3	Functional Abstraction • Task decomposition (5) • Reduction of repetitive code (5)	/10
C4	Separate Compilation • Header file (1) • Guard conditions (2) • Inclusion of header file (1) • Appropriate content in header file (1) • Use of programmer defined namespace (5)	/10
C5	User Interaction • Menu System (5) • Appropriate use of input, output and error streams (5)	/10
C6	Command Line Argument Handling: • Appropriately overloaded main function (1) • Handling incorrect argument counts (1) • Use of supplied arguments (3)	/5
C7	Error Handling • Use of assertions (2) • Use of conventional error handling techniques (3)	/5
C8	Pseudo-random number generation (5)	/5
C9	Dynamically allocated two dimensional array handling with structures • Allocation (5) • Initialisation (10) • Deallocation (5)	/20
C10	Algorithm implementation • Logical Correctness (5) • Effectiveness / Efficiency of approach (5) • Correct output (5)	/15
B	Bonus	/10
Total:		**/100**