Problem Identification – Computational methods

There are five computational methods that apply to this course.

1. Thinking abstractedly [and visualisation] (a) the nature of abstraction. (b) The need for abstraction. (c) The differences between an abstraction and reality. (d) Devise an abstract model for a variety of situations.
2. Thinking ahead a) Identify the inputs and outputs for a given situation. (b) Determine the preconditions for devising a solution to a problem. (c) The nature, benefits and drawbacks of caching. (d) The need for reusable program components.
3. Thinking procedurally [and ‘decomposition’] (a) Identify the components of a problem. (b) Identify the components of a solution to a problem. (c) Determine the order of the steps needed to solve a problem. (d) Identify sub-procedures necessary to solve a problem.
4. Thinking logically (a) Identify the points in a solution where a decision has to be taken. (b) Determine the logical conditions that affect the outcome of a decision. (c) Determine how decisions affect flow through a program.
5. Thinking concurrently (a) Determine the parts of a problem that can be tackled at the same time. (b) Outline the benefits and trade-offs that might result from concurrent processing in a particular situation.

The actual thing

The first computational method is thinking abstractedly and visualisation. This applies to the game because abstraction is when you remove attributes of the entity i.e. simplifying the game from reality. This works with my game because it extremely hard to mimic the game as though in real life it is unrealistic to put that much details into my project, because apart from being out of my skill range would use large quantities of memory and resources and would be unnecessary. In my game there are some elements which will be simplified. These include: simplifying the visual graphics so features like the background (might not be the best one to do) if it had too much details it would distract from the characters and other parts of the game and would look bad. Also the characters will be visually simple to stop from distracting from the game.

The second computational method is thinking ahead. This consists of considering data or inputs which are required form this program to work: the input of the arrow keys will be used for the characters to move around the 2D maze. (There will be checks to see if the characters go into the wrong ponds which will kill them and the characters will fail the level. This is shit)

The third computational method is thinking procedurally or “decomposition” this is when a problem is broken down into its component parts. This can be done in my program