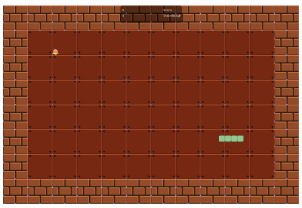
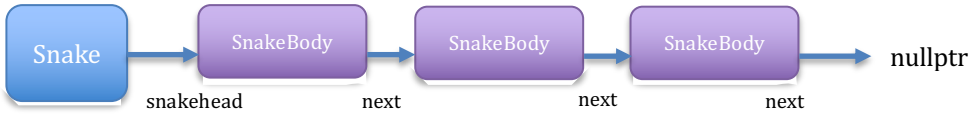
Game Programming: Exercise 3: C++ Pointers and memory

	**Learning objectives:** • Allocate dynamic memory in C++ using pointers (`new`, `delete`, -> operator, `nullptr`) • Write programs that manage memory without any leaks • Iterate tree structures e.g. using recursion • Explaining the relationship between pointers and memory addresses • Explaining the memory layout of stack-based and heap based memory layout. • Use shared pointers For exercise 3.1 to 3.5 you only need to modify Snake and SnakeBody. An online version of the final game can be found here: http://www.itu.dk/~mnob/snake/Snake.html
Exercise 3.1	**Move snake** Setup project as a SimpleRenderingProject using CMake. Make snake move by implementing Snake::move. The member function should compute the new position of the snake-head and call moveTo() on its object. The move-direction is defined in Snake::moveX and Snake::moveY. When implemented correctly you should be able control a single element snake.

Exercise 3.2	**Initial length and snake grow.** • **Snake::init()**. Must create a snake with an initial length, position, and move-direction. If a snake already exists make to destroy it to prevent leaking memory (the number of SnakeBody parts visible should correspond to the counter). • **Snake**::**grow()**. Called when the snake eats a piece of fruit. This should increase the length of the snake by adding a SnakeBody to the snake. You must use raw pointers (using new and delete). The Snake object has a pointer to the snake head (an object of SnakeBody). Each SnakeBody object has a pointer to the next part in the 'linked-list'. The last has a next pointer to nullptr. Also make sure that the snake moves as expected (modifications to SnakeBody::moveTo() may be needed). 
Exercise 3.3	**Implement Snake::collide** This should return true if snake collides with itself.
Exercise 3.4	**Modify Snake::setMoveDirection** If the user attempts to move in the opposite direction than the current move-direction then ignore the call by not modifying Snake::moveX and Snake::moveY.
Exercise 3.5	**Using smart pointers** Replace use of raw pointers in Snake and SnakeBody with std::shared_ptr.
Exercise 3.6 (Optional)	**Optional additions** Extend the game to a two-player snake game, where the other snake is controlled using the AWSD keys. Show a score per snake. Other additions (such as other level-layouts). Multiple levels. Timers. Enemies.