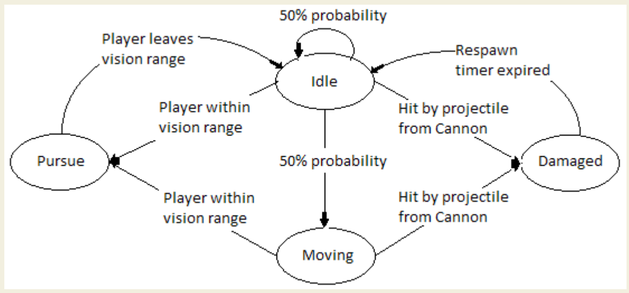
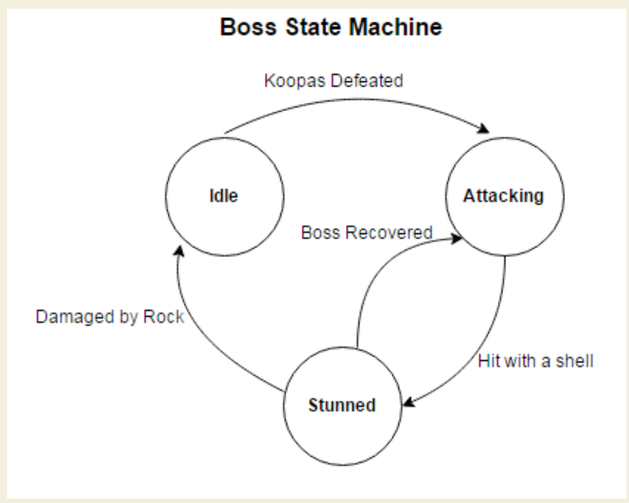
Technical Achievements

ARTIFICIAL INTELLIGENCE



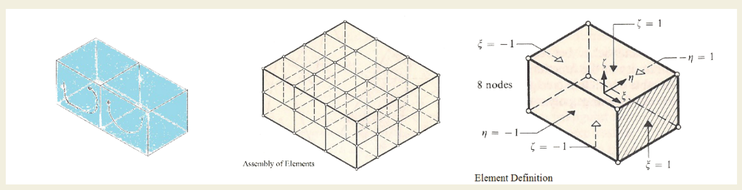
This state machine outlines behavior of enemy Robots inside the House level.



This state machine shows how the Tomb Boss changes based on game events.

Artificial intelligence is implemented to create computer-controlled 'agents' with more interesting behaviors. Here we implement this behavior through the use of state machines. A state machine is implemented through a series of possible 'states' (represented as bubbles in the diagrams to the left). States represent all possible conditions of the agent (ie. "Alive" or "Dead"). A state machine also requires 'transitions', events that cause an agent to move between states (ie. moving from "Alive" to "Dead" via the "Die" transition). Many kinds of events can be transitions, making state machines a flexible AI implementation.

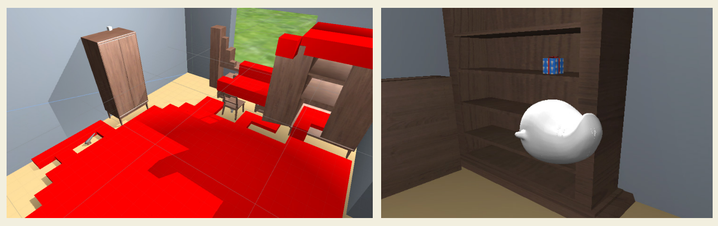
WATER PHYSICS



(Left to Right) In order to simulate currents, the space around the submarine is defined and broken into sections. Each section is analyzed for forces at 8 different points, and the resulting effects are processed and then applied to the submarine.

The water world submarine is subject to physics-based effects such as currents and gravity. To maintain good performance, the effects of such things are approximated to reasonable accuracy rather than being directly computed. (Currents are described to the left).

FLOOD FILL



A visual representation of the algorithm (Left) and the final result as seen in the game (Right).

A Flood Fill algorithm was used to place gifts randomly in the House level. The algorithm recurses over the entire level finding flat spots which can hold objects. Valid locations are written into a text file, and those values are selected to place the gifts at runtime. This ensures the random placement of gifts and also helps prevent very long loading times.

MAZE GENERATION

These two mazes are visibly different, but they were both created by the same algorithm.

The Tomb maze is randomly generated through use of a depth-first search algorithm, which takes a graph of points and generates paths to every point in the graph. This algorithm ensures that the maze is different every time, and that it is always able to be solved.