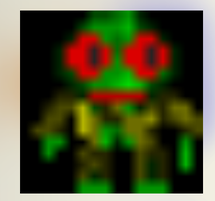
Concurrent Zombie ~~Flock~~ ~~Horde~~ Swarm

University of Kent

Featuring:

- {  } Functional programming
- {  } Lots of processes
- {  } Websockets
- {  } Flocking algorithms

Erlang allows us to model the simulation as a concurrent system. Each zombie, human and just about any other entity involved is it's own, independent process.

A message passing framework lets entities communicate with each other, updating other processes that are located nearby.

Using concurrency to model the behaviours present in this simulation means that an entity may change it's mind about what it is doing, before another entity can react to it; this creates a realistic sense that each entity is thinking for itself, as opposed to automatically updating in the next discrete time step.

The entities in the simulation all act using computational intelligence. They analyse their environment and make decisions without user input.

Decoupled client/server architecture provides scalability and the simulation has the potential to be executed across an Erlang cluster.
D3.js proof-of-concept client and websocket/JSON API for maximum flexibility.

Oh, did we mention it's made of Zombies?

Erlang