Overview of dhewm3 code

- lines of code for the id engines

	Doom	idTech1 (quake1)	idTech2 (quake2)	idTech3 (quake3)	idTech4 (doom3)
Engine	39079	143855	135788	239398	601032
Tools	341	11155	28140	128417	0
Total	39420	155010	163928	367815	601032

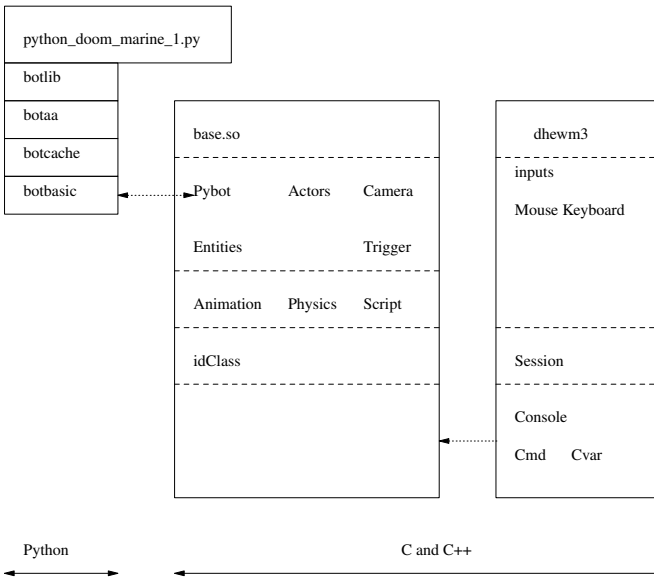
- the 0 lines of code for the doom3 is due because doom3 includes them in game
- dhewm3 is about 681075 total lines of code
  - however this includes many tools and libraries

Overview of dhewm3 code

- actual engine is closer to 137905 lines of code
- abstraction and polymorphic code employed
- all assets are stored in human readable text form
  - this works well on our fast machines now, this design also guides our python bot library
- templates are used in low level utility classes (mainly idlib) but are never seen in the upper levels
  - no STL used
- well commented code base

Overview of dhewm3 and Python Bot code

-



Python Bot Overview

- note, the bot code is only partially complete
- layered approach
- top layer, `botlib` will ultimately be the interface to user level bots (`python_doom_marine_1.py`)
- will contain both access to basic movement and access to navigation mechanisms
- maintains the transform between the doom3 map and penguin tower map

botaa

- contains area awareness code for the python bot
- parses the equivalent pen map and creates internal simple 2D map of the world in Python
- implements Dijkstra's algorithm for routing to pickups and player/monsters

botcache

- provides the same interface as
  - `botbasic` but it caches all results
  - which allows the higher layers (`botaa`) the ability to query the bot without having to worry about the overhead of the remote procedure call
- it also contains a method
  - `reset` which deletes the cache, forcing all future requests to be sent over to the dhewm3 engine to retrieve up to date values

botbasic

- provides socket connection to the dhewm3 engine
- it also implements the underlying remote procedure call mechanism
- generally maps onto basic commands
  - such as: move, fire, reload, turn, aim, etc

Homework: read how Remote procedure calls work

- you need to understand the concept of remote procedure call argument marshalling
  - make sure you have read Tanenbaum's description of how RPC's operate
  - and how marshalling is employed and why it is necessary

- Andrew Tanenbaum, Modern Operating Systems, Prentice-Hall International
  - sections 10.3, 10.3.1, 10.3.2