Introducing unique global object into doom3 bot API

- currently the bot has access to basic movements
  - jump, crouch, step forward, step backwards, step left, step right, turn, aim, fire
  - the bot can also select weapon, reload
- the bot can navigate around the map using Dijkstra's algorithm and A*
- there is *some* ability to wait for one of a number of events to occur

Introducing unique global object into doom3 bot API

- it would be good to introduce `labels` into chisel which could be accessible by the python bot API
  - the bot could navigate to various labels
  - the bot could walk until a label position was visible etc

- it would be good to unify all global objects
  - ammo pickups, weapon pickups, health pickups, armour pickups, other bots, human players
  - labels

Introducing unique global object into doom3 bot API

- suggest design change:
  - all global objects accessible to the python bot API would be registered by the doom3 engine
  - after reading in the map

- each unique global object (UGO) would be assigned a unique `integer` value
  - the API would be extended to allow access to an object via the UGO

Introducing unique global object into doom3 bot API

- it would be good to introduce the ability of the bot to obtain items in sets
  - for example

-
```
get_human_player_set ()
get_pybot_player_set ()
get_monster_set ()
get_ammo_set ()
get_armour_set ()
get_label_set ()
```

Introducing unique global object into doom3 bot API

- where the API returns a list of UGO's
- this can be further utilised in a new implementation of a function `select`
-

```
def select (eventlist,
            visibleobjectset, visibleobjectlist,
            movementobjectset, movementobjectlist,
            audibleobjectset, audibleobjectlist,
            timeout):
```

select

- blocks the bot for at most timeout seconds or when an event from, eventlist, occurs
  - it might also return early if the list of visibleobjects moves

- pre-condition: `eventlist` is a list containing any of: `['move', 'fire', 'turn', 'reload', 'movement', 'visible']`

select

- `visibleobjectset` is the list of objects that the caller is interested
- `visibleobjectlist` is the list of objects which are currently visible
- the combination of these two parameters allows the caller to check for objects becoming visible and non visible

select

- for example using this call

```
select (['visible'], [1, 2 ,3], [1, 3], [], [], [], [], 5)
```

- the list `[1, 2, 3]` indicates that the caller is interested in `UGO`'s `1`, `2` and `3`
- and `select` will block until object `1` or `3` becomes invisible, or `2` becomes visible
- it blocks for at most 5 seconds

select

- the movement and audible list and sets behave in exactly the same way as the visible
  - modulo obvious differences

post-condition: select

- select returns the following:
  - eventlist, visibleobjects, movementobjects, timeout
- eventlist
  - is the list of events which have occurred
- visibleobjects
  - is the list of objects which have changed their visible status.
- movementobjects
  - is the list of objects which have moved
- timeout
  - is the amount time consumed since the call was initiated.

Benefit of changing select in the API

- the bot can be written to use an event based paradigm
- `select` could almost be considered as a fundamental component on which one could build a scheduler for bot activity

An example of a very simple bot scheduler based on select

```
action = get_next_movement ()  #  stepForward (...)
perform_movement (action)

eventlist, visibleobjects,
movementobjects, timeout = select (['move', 'visible'],
                                   [monster, label, human_player],
                                   [], [], [], [], 5)
if 'visible' in eventList:
   ai_doVisible (visibleobjects)
if 'move' in eventList:
   ai_movement_complete ()

ai_decide ()   # decides what the bot should do next
```

- further reading see André LaMothe, "Tricks of the Windows Game Programming Gurus: Fundamentals of 2d and 3d Game Programming", Sams; 2 edition, June 2002, ISBN-10: 0672323699, ISBN-13: 978-0672323690, P. 740