

Architecture of Penguin Tower

- penguin tower consists of three main components
 - server, protocol and client
- server is written in Modula-2 and is multi-threaded and uses one thread for each player
 - a thread for each players normal arrow
 - a thread for each players magic arrow
 - a thread for the hand grenade

Hand grenade

- a single thread which waits in a loop for the pin to be pulled

```

LOOP
  Wait(PinPulled) ;
  Sleep(25) ;
  CalculateDamage
END

```

Arrow threads

- again LOOP waiting to be released at which time they will be given start coordinates and direction
- they then move the arrow a square at a time (independantly from the rest of the game)
 - apart from areas of *mutual exclusion*
- when an arrow hits something it determines whether it has hit a person or an object

Player threads

- again are continually in a LOOP reading a single character
- depending upon the character the particular operation is performed (see key commands in penguin tower documentation)

Multi Threading in the server

- main advantages of threading are:
- conceptually easy to consider each activity in isolation
 - global data structures are protected by mutual exclusive semaphores
- allows independant action of players, arrows and hand grenade
- when a new player joins a new thread is created appropriately
 - likewise when a player leaves or disconnects the thread is disposed

Protocol

- the client attempts to connect using a tcp socket on port number 7000 (or higher)
- once connected the client sends single characters to the server
 - except Esc which if pressed terminates the game
- the server sends lines of text to the client
 - each line is terminated by a newline \n character

Protocol

- the advantages of this approach is
 - simplicity and the ability to test the server and client independantly
 - you can connect to the server using telnet on port 7000
 - likewise you can construct a file containing lines of text which will be read by the client

Protocol lines from the server

- each line from the server instructs the client to perform some activity
 - each line received is stored in the history list which could be re-read to give a slow motion (or fast motion) replay
 - see the function `processLine` in file `penguin-tower.py`
- `sync` erases the history list
 - useful so the replay does not become excessively long
- `f1` flush, forces the client to flip the double buffer
- `quit` or `abort` terminates the client
- `clear` clear the screen
- `status` draw the status screen with blank entries
- `dWriteLn somestring`
 - write *somestring* to the screen and move the cursor to the next line

Protocol lines from the server

- dWriteStr *somestring*
 - write *somestring* to the screen and leave the cursor at the end of the string

Protocol lines from the server

- dC *character*
 - write the single *character* to the screen
- eC erase character at the current cursor position
- dN *somestring*
 - display the name *somestring* on the status screen
- dMap *somestring*
 - display the mapname *somestring* in the status window

Protocol lines from the server

- dC1 *somestring*
 - display *somestring* in the first line of the comment area
- dC2 *somestring*
 - display *somestring* in the second line of the comment area
- dC3 *somestring*
 - display *somestring* in the third line of the comment area
- dC4 *somestring*
 - display *somestring* in the fourth line of the comment area

Protocol lines from the server

- dCMD *somestring*
 - display the action command in the status window

Protocol lines from the server

- eman *x y*
 - display player facing east at position, *x, y*
- wman, sman and nman
 - same as eman but facing in respective directions
- Wman, Eman, Sman and Nman
 - display opponent facing the appropriate direction

Protocol lines from the server

- hwall *x1, y1, x2, y2*
 - display a horizontal wall using the coordinates
- vwall *x1, y1, x2, y2*
 - display a vertical wall using the coordinates
- hhinge *x1, y1, x2, y2*
 - draw a horizontal hinge using the coordinates
- vhinge *x1, y1, x2, y2*
 - draw a horizontal hinge using the coordinates

Protocol lines from the server

- war, ear, nar, sar *x, y*
 - draw an arrow facing in a specified direction at position, *x, y*
- dW *somestring*
 - draw *somestring* in the wounds field
- dF, dA, dM dR dT
 - draw fatigue, draw arrow, draw magic arrow, draw room, draw time in the same way as draw wounds

Protocol lines from the server

- pS *somestring*
 - play sound file *somestring*
- eL *x1 y1 x2 y2*
 - erase line