MUDDL TO MUDDLE

DESIGNING A (SCRIPTING?) LANGUAGE FOR MMOS

8TH MAY, 2018

CODE EUROPE

PROF. RICHARD A. BARTLE

UNIVERSITY OF **ESSEX**, UK

INTRODUCTION

- SO, THIS TALK CONCERNS THE **HISTORICAL** EVOLUTION OF A **PROGRAMMING** LANGUAGE **NONE** OF YOU HAVE **HEARD** OF
  - **MUDDLE**
- ITS PURPOSE IS TO MAKE SOME **WIDER** POINTS ABOUT LANGUAGE **DESIGN** AND **USE**
  - AT LEAST FOR **CREATIVE** PROJECTS
- HOWEVER, IT **ALSO** SHINES A LIGHT ON HOW **GAME** DEVELOPMENT WAS IN THE OLD DAYS
  - SO **MAY** BE OF INTEREST TO **HISTORIANS**...

MUD

- THE GAME I'LL BE TALKING ABOUT IS *MUD*
  - "MULTI-USER DUNGEON"
- ALMOST **ALL** MODERN MMORPGS ARE **DIRECT** DESCENDANTS OF *MUD*
  - **INCLUDING** THOSE DEVELOPED IN KOREA AND CHINA
- *MUD* WAS WRITTEN BY ROY **TRUBSHAW** AND RICHARD **BARTLE** (ME!), IN **1978**
  - **ROY** INITIATED IT; I BECAME INVOLVED A FEW WEEKS **AFTERWARDS**
- THIS YEAR IS ITS **40TH** ANNIVERSARY

SCREEN

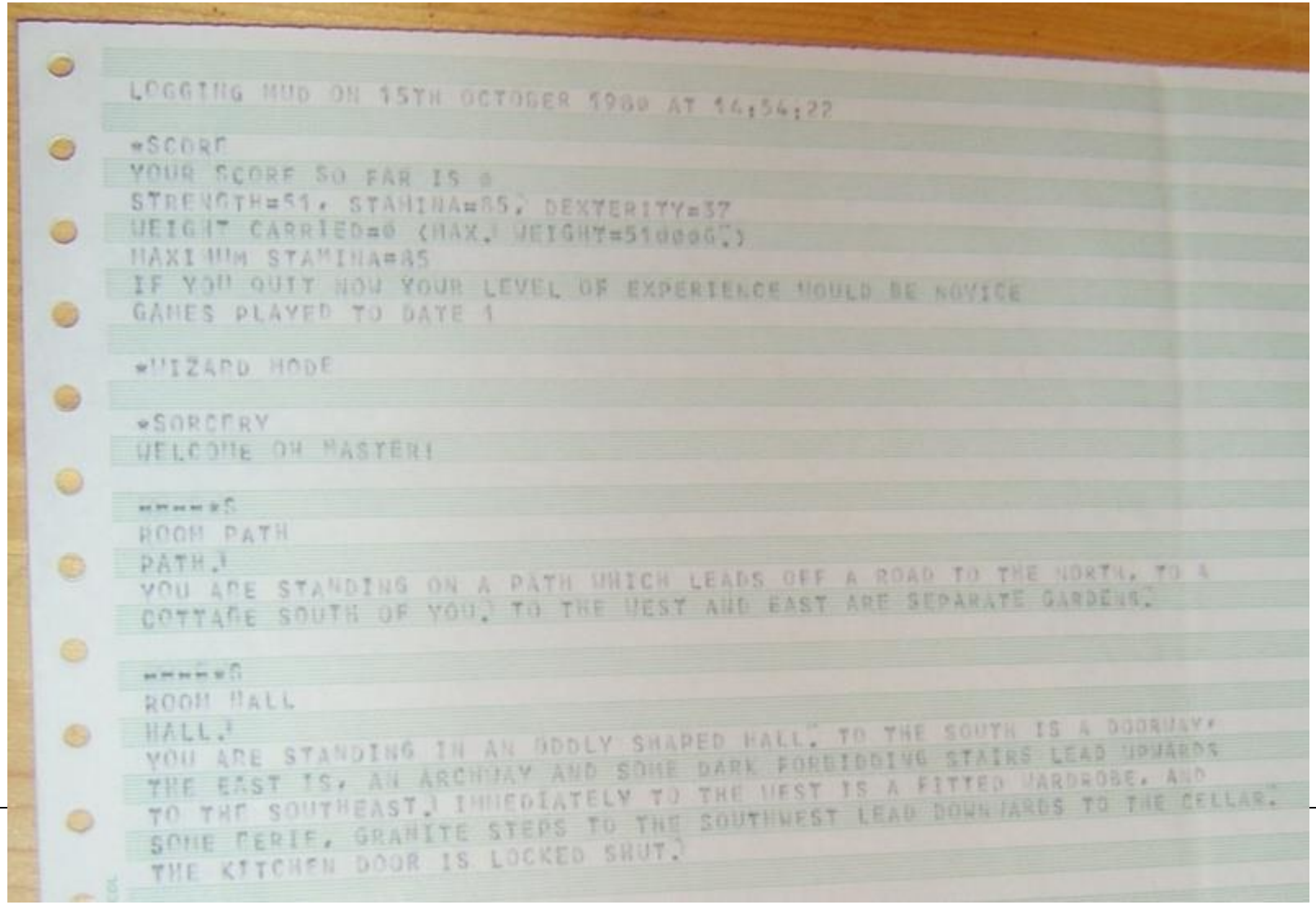
- THIS IS WHAT *MUD* **LOOKED** LIKE ... SORT OF

```
Narrow road between lands.
You are stood on a narrow road between The Land and whence you came.
To the north and south are the small foothills of a pair of majestic
mountains, with a large wall running round. To the west the road
continues, where in the distance you can see a thatched cottage
opposite an ancient cemetery. The way out is to the east, where a
shroud of mist covers the secret pass by which you entered The
Land. It is raining.
*w
Narrow road.
You are on a narrow east-west road with a forest to the north and
gorse scrub to the south. It is raining. A splendid necklace lies
on the ground.
*
```

- IT'S A **MOCK-UP** OF A SCREEN, BECAUSE BACK IN 1978 WE HAD **NO SCREENS**
  - WE USED **TELETYPES**

LOG

- HERE'S A **PRINTOUT** OF A 1980 *MUD* LOG...



ARCHITECTURE

- MUD WAS A **TEXTUAL** WORLD
  - MMORPGS WERE **ORIGINALLY** CALLED "GRAPHICAL MUDS"
- IT WAS WRITTEN ON A ROOM-SIZED COMPUTER, THE **DEC-10** (OR **PDP-10**)
  - THE PRIMARY SCIENTIFIC COMPUTER OF ITS ERA
  - **SO** MUCH BETTER THAN THE IBM 360
- THE DEC-10 HAD A **BEAUTIFULLY**-DESIGNED INSTRUCTION SET AND ARCHITECTURE
  - SADLY, ITS **18-BIT** ADDRESS SPACE, WHICH SEEMED **LARGE** AT THE TIME, **WASN'T**

MOTIVATION

- I WON'T EXPLAIN THE **REASONS** THAT ROY AND I DEVELOPED *MUD*, BECAUSE **THAT** WOULD BE A TALK ALL BY **ITSELF**
  - AS INDEED IT **IS**: HTTP://WWW.GDCVAULT.COM/PLAY/1013804/MUD-MESSRS-BARTLE-AND-TRUBSHAW
- I'M GOING TO DISCUSS HOW *MUD* WAS **IMPLEMENTED**
- DEVELOPMENT WENT THROUGH **FOUR** STAGES:
  - MUD VERSION 1, MUD VERSION 2, MUD VERSION 3 ("MUD1"), MUD VERSION 4 ("MUD2")

VERSION 1

- *MUD* VERSION 1 WAS ONLY A **TEST** PROGRAM TO MAKE SURE THAT INTER-PLAYER **COMMUNICATION** WORKED
- *MUD* PROCESSES **COMMUNICATED** THROUGH SHARED, WRITEABLE **MEMORY**
  - **NOT** CLIENT/SERVER, UNLESS YOU COUNT DUMB TERMINALS AS **CLIENTS**
- IT WAS WRITTEN IN **TWO** HOURS OR SO USING MACRO-10 **ASSEMBLER**
- THE PROOF OF CONCEPT **WORKED**, SO ROY **IMMEDIATELY** STARTED ON VERSION 2

VERSION 2

- *MUD* VERSION 2 TOOK SEVERAL **WEEKS** TO REACH A POINT WHERE IT WAS **PLAYABLE**
- IT WAS **ALSO** WRITTEN IN MACRO-10
- THE CODE **ITSELF** WAS COMPACT AND CLEAN, BUT THE PROBLEM WAS **CONTENT**
- IT WASN'T **CALLED** CONTENT BACK THEN, OF COURSE, AS THE TERM HADN'T BEEN **INVENTED**
  - WE'D SAY THERE "WASN'T MUCH **THERE**", OR THERE "WASN'T MUCH TO **DO**"
- WE HAD AN **ENGINE**, BUT NOT A LOT OF **FUEL** FOR THE ENGINE TO **RUN** ON

ADDING CONTENT

- IN VERSION 1, CONTENT WAS **HARD-CODED**
  - THERE WAS HARDLY **ANY** AND IT WAS **THROWAWAY** CODE, SO THIS MADE **SENSE**
- IN VERSION 2, CONTENT HAD TO BE **ADDED**
- COMPUTER **INTERFACES** BACK THEN USED A **COMMAND LINE**
- HMM! *MUD* PLAYERS ISSUED THEIR INSTRUCTIONS TO THE **GAME** AS **COMMANDS**
  - N, GET KEY, E, OPEN DOOR WITH KEY, ...
- ROY THEREFORE ADDED **PLAYER** COMMANDS TO ADD GAME **CONTENT**

META-LANGUAGE

- HIS SOLUTION WAS BASICALLY A **BOOTSTRAP** APPROACH
- HE **HARD-CODED** INTO *MUD* A SET OF COMMANDS THAT COULD BE USED TO ADD **NEW** COMMANDS FROM **WITHIN** *MUD* ITSELF
- IF YOU WANTED TO CREATE A **CREATURE**, FOR EXAMPLE, YOU'D RUN *MUD* AND ISSUE A COMMAND SOMETHING LIKE `create ox`
  - IT WOULD ADD THE NEW **OX** OBJECT TO THE DATA STRUCTURES
  - OTHER COMMANDS COULD THEN **MODIFY** IT

SUCCESS

- THIS **WORKED**, BUT IT HAD **PROBLEMS**
- CONTENT WAS HARD TO **EDIT** AND **REMOVE**
  - AND EVEN HARD TO **LIST**
- AS MORE COMMAND TYPES WERE **ADDED**, THE CODE TO PROCESS THEM BECAME **BIGGER**
  - **SO** BIG THAT IT IMPACTED ON **MEMORY** USE
  - CODE AND DATA SHARED THE SAME MEMORY SEGMENT
- ALSO, WRITING IN ASSEMBLER WAS **SLOW**
- IN 1979, ROY THEREFORE DECIDED TO **REWRITE** *MUD* FROM **SCRATCH** AGAIN AS VERSION 3

VERSION 3

- ROY'S MAIN **AIMS** FOR VERSION 3 WERE:
  - TO WRITE IN A **BETTER** LANGUAGE THAN MACRO-10
  - TO MOVE CONTENT-CREATION **OUTSIDE** THE GAME
- THE BETTER LANGUAGE WAS **BCPL**, THE FORE-RUNNER OF C
  - A **WONDERFUL**, TYPELESS SYSTEMS-PROGRAMMING LANGUAGE, I LOVE IT!
- HE CREATED A **DATA** FILE FOR CONTENT
- HE WROTE A PROGRAM TO **COMPILE** THE DATA FILE INTO **MACRO-10**, WHICH WAS THEN **ASSEMBLED** AND **LOADED** INTO *MUD*

GONE

- WITH COMMAND-LINE PARSING OF CONTENT-CREATION **GONE**, ROY WAS FREE TO USE LESS **ENGLISH-LIKE** SYNTAX FOR CONTENT
  - COMMAND-LINE CONTENT-CREATION WAS LATER REINVENTED FOR **SOCIAL** MUDS (EG. *TINYMUD*)
- HE DESIGNED A **LANGUAGE** FOR DEFINING *MUD* CONTENT, WHICH HE CALLED **MUDDL**
  - "MULTI-USER DUNGEON DEFINITION LANGUAGE"
  - THE NAME WAS A CONSCIOUS NOD TO **MDL**, WHICH WAS USED FOR *ZORK*
- SO HOW DID HE GO ABOUT **DOING** THAT?

ADVENT

- ROY DID THE **SAME** THING PROGRAMMERS **ALWAYS** DO IN SUCH CIRCUMSTANCES: LOOK TO SEE HOW **OTHER** PEOPLE DID IT!
- PROBLEM: THERE **WERE** NO OTHER MUDS HE KNEW OF! *MUD* WAS THE FIRST!
  - *AVATAR* AND *SCEPTRE OF GOTH* **DID** EXIST BY THEN, BUT NONE OF US KNEW OF THE OTHER TWO
- ROY LOOKED AT THE **SINGLE-PLAYER** GAME, *COLOSSAL CAVE*, KNOWN TO US AS *ADVENT*
- HE BASED **SOME** OF **MUDDL** ON *ADVENT*'S (HARD-CODED) **DATA STRUCTURES**

SECTIONS

- MUDDL WAS SPLIT INTO SEVERAL **SECTIONS**, THE **MAIN** ONES BEING:
  - ROOMS
  - VOCABULARY
    - CLASSES
    - OBJECTS
    - ACTIONS
  - TRAVEL
  - TEXT
- FROM **OUR** PERSPECTIVE, THE **VOCABULARY** SECTION IS THE MOST INTERESTING

VOCABULARY

- THE VOCABULARY LISTED THE **WORDS** THAT THE PLAYERS COULD USE AND **DEFINITIONS** OF THOSE WORDS
- ROY HAD A **TWO-LEVEL** STRUCTURE FOR NOUNS
- "CLASSES" WERE **COLLECTIONS** OF "OBJECTS"
- "OBJECTS" WERE ACTUAL GAME **TOKENS**
- CLASSES **WEREN'T** PROPER CLASSES AS WE'D UNDERSTAND THEM **TODAY**
  - ALL OBJECTS HAD TO **HAVE** A CLASS, BUT NO CLASS COULD HAVE **SUBCLASSES**

VOCABULARY OBJECTS

- HERE'S WHAT THE VOCABULARY ENTRIES FOR **OBJECTS** LOOKED LIKE:

```
chain                links       4000       40
mosaic               chip        10         5
stove                oven        0          0
trophy               triumph     1000       35
throne               chair       60000      200
forge                flame       0          0
longsword            killer      2250       0
broadsword           sword       2250       163
sabre                sword       2250       0
```

- OBJECT, CLASS, **WEIGHT** IN GRAMS, **VALUE** IN POINTS
  - NOTE THAT THESE NUMBERS **AREN'T** REALLY WHAT YOU'D CALL "VOCABULARY" ITEMS
- MOST CLASSES ONLY HAD **ONE** OBJECT IN THEM
  - ALTHOUGH `sword` THERE HAS **TWO**

OBJECTS

- MUDDL STARTS TO GET **COMPLICATED** WHEN IT COMES TO OBJECT **DEFINITIONS**
- OBJECTS IN *MUD* VERSION 3 HAD DIFFERENT **STATES** KNOWN AS **PROPERTIES**
  - THEY ALSO HAD **OTHER**, BINARY PROPERTIES...
- HERE'S A RELATIVELY **SIMPLE** OBJECT DEFINITION:

```
longsword sea14     1          1          2          bright    nosummon
0         A murderous, blood-stained longsword lies here.
1         Thrust deep into a rock is a murderous longsword!
```

- THE LONGSWORD STARTS IN SEA14, WITH INITIAL PROPERTY 1, MAX PROPERTY 1, VALUE PROPERTY 2 (SO NOT WORTH POINTS), IT GLOWS IN THE DARK AND BLOCKS SUMMON SPELLS

FORMATS

- *ADVENT* HAD **TWO** FORMATS FOR COMMANDS; ROY HAD ADDED A **THIRD** FOR *MUD*:
  - <VERB>
    - *EG.* QUIT
  - <VERB> <NOUN>
    - *EG.* GET SWORD
  - <VERB> <NOUN> <PREPOSITION> <NOUN>
    - *EG.* OPEN DOOR WITH KEY
- **ALL** GAME COMMAND INTERFACES (EVEN **GRAPHICAL** ONES) ESSENTIALLY REDUCE TO FIND-A-FUNCTION-AND-PARAMETERS

ACTIONS

- ACTION DEFINITIONS ARE THE **MOST** COMPLICATED COMPONENTS OF MUDDL:

```
get       .get      killer    none      ifprop    null      0         0
get       killer    none      unlesslevel         null      5         1049
get       .get      killer    none      set       null      0         1021
```

- SO, **killer** IS THE CLASS FOR **longsword**...
- THE BASIC **FORMAT** IS: VERB SUBJECT OBJECT CONDITION PARAMETER TRUE FALSE
  - THE **.get** IS A **HARD-WIRED** GET FUNCTION
- TRANSLATION (ALL THESE ARE FOR **get longsword**):
  - IF THE LONGSWORD IS IN PROPERTY 0, JUST PICK IT UP
  - OTHERWISE, IF YOU'RE NOT LEVEL 5 PRINT MESSAGE 1049
  - OTHERWISE, SET ITS PROPERTY TO 0, PRINT MESSAGE 1021 AND THEN PICK IT UP

LIMITS

- ALTHOUGH MUDDL WAS **POWERFUL**, IT WASN'T POWERFUL **ENOUGH**
- ITS **ACTION** FORMAT DIDN'T ALLOW FOR **LOOPS** OR MULTIPLE **TESTS**
- THE **SPECIAL** COMMANDS SUCH AS `.get` HAD TO BE HARD-CODED IN, WHICH PUT PRESSURE ON THE **MEMORY** AVAILABLE FOR **OTHER** CODE
  - AND UNDERMINED THE **POINT** OF HAVING A DEFINITION LANGUAGE IN THE FIRST PLACE
- WE HAD **99** SPECIAL FUNCTIONS BY THE END OF V3, BUT THAT'S **NOT** WHAT LED TO V4...

REPETITION

- **THIS** IS WHAT FINALLY DID FOR MUDDL:

```
feed    nanny   pan       null      null      681       0
feed    nanny   victuals  destroy   second    682       0
feed    nanny   antidote  destroy   second    682       0
feed    nanny   flower    destroy   second    682       0
feed    nanny   fungus    destroydestroy      toadstool 683       0
feed    nanny   limb      null      null      684       0
feed    nanny   corpse    null      null      684       0
feed    nanny   sprig     destroydestroy      mistletoe 685       0
feed    nanny   frog      null      null      684       0
feed    nanny   bird      null      null      684       0
feed    nanny   birdofprey          null      null      684       0
feed    nanny   rodents   null      null      684       0
feed    nanny   bunny     null      null      684       0
feed    nanny   vermin    null      null      684       0
feed    nanny   familiar  null      null      684       0
feed    nanny   herring   destroy   second    682       0
feed    nanny   serpent   null      null      684       0
feed    nanny   nut       destroy   second    682       0
feed    nanny   pen       destroy   second    682       0
feed    nanny   parachute destroy   second    682       0
feed    nanny   money     destroy   second    682       0
feed    nanny   gem       destroy   second    682       0
feed    nanny   liquid    destroy   second    682       0
feed    nanny   rum       null      null      930       0
feed    nanny   medication          destroy   second    682       0
feed    nanny   paper     destroy   second    682       0
feed    nanny   map       destroy   second    682       0
feed    nanny   tome      destroy   second    682       0
feed    nanny   adventurer          null      null      1094      0
feed    nanny   book      destroy   second    682       0
```

VERSION 4

- IN ORDER TO **ESCAPE** THIS LIMITATION, I DECIDED TO REWRITE *MUD* FROM **SCRATCH**
  - **VERSION 4**, WHICH BECAME KNOWN AS *MUD2*
- AT THE **CORE** OF IT WOULD HAVE TO BE A NEW **DEFINITION** LANGUAGE
  - WHICH I CALLED **MUDDLE**
  - **M**ULTI-**U**SER **D**UNGEON **D**EFINITION **L**ANGUAGE
- I HAVE **TWO** EXERCISE BOOKS FULL OF **NOTES** ON THE DESIGN OF MUDDLE
- IT'S A FULLY-FLEDGED PROGRAMMING LANGUAGE
  - YOU CAN WRITE A MUDDLE **COMPILER** IN MUDDLE

SEPARATION

- MUDDLE SEPARATED THE **VOCABULARY** FROM THE **PROGRAMMING** OBJECTS:

```
$[        eye
          noun::    ruby1
          verb:     eye
$]
```

- THIS SAYS THAT THERE'S A **WORD**, `eye`, WHICH WHEN IT'S USED AS A **NOUN** REFERS TO THE **ATOM** `ruby1` AND WHEN IT'S A **VERB** REFERS TO THE ATOM `eye`
  - THE `::` MEANS IT'S A ONE-WAY LINK, SO `ruby1` DOESN'T KNOW THAT `eye` IS A SYNONYM FOR IT
  - `ruby1` IS A GAME **TOKEN** (A PARTICULAR RUBY)

PARSING

- I'M NOT GOING TO DESCRIBE *MUD2*'S PARSING IN **DETAIL**, BUT IT WAS VERY **STRONG**
  - **`pick up all the gems except the green one and put them in the smallest box`**
- THE (HARD-WIRED) PARSER GAVE THE MUDDLE **INTERPRETER** A SERIES OF COMMANDS
- COMMANDS WERE **LISTS** OF 1, 2 OR 3 ATOMS
  - OR STRINGS, FOR *EG.* `tell` COMMANDS
- THESE LISTS OF ATOMS WERE PATTERN-MATCHED AGAINST **DEFINITIONS** WRITTEN IN MUDDLE
- **THIS** IS WHERE IT GETS INTERESTING...

PATTERNS

- MUDDLE CODE IS ASSOCIATED WITH **PATTERNS**:

```
{ get longsword }:
{ get longsword room }:
{ get longsword loosener }:
{ get longsword creature }:
{ get longsword container }:
```

- THESE PATTERNS MATCH THE **FUNCTIONS** AND **PARAMETERS** THAT COME FROM COMMANDS
- **IMPORTANT**: ALL THOSE ATOMS THERE REPRESENT **CLASSES**
  - `{ get longsword room }`: MATCHES ANY COMMAND OF TYPE `get` APPLIED TO ANY OBJECT OF TYPE `longsword` AND ANY OBJECT OF TYPE `room`
- **INSIGHT**: THE ATOMS **ARE** THE CLASSES

CLASSES

- IN A LANGUAGE SUCH AS C++ OR JAVA, CLASSES ARE **TEMPLATES** FOR **STAMPING** OUT OBJECT **INSTANCES**
- IN MUDDLE, OBJECTS **AND** CLASSES ARE JUST **ATOMS**
  - AN OBJECT IS MERELY AN ATOM WITH NO CHILDREN
- YOU COULD, IF YOU LIKED, ALLOW PLAYERS TO HOLD THE **CONCEPT** OF A LONGSWORD, RATHER THAN SOME **PARTICULAR** LONGSWORD
  - ALTHOUGH CLASSES-AS-CONCEPTS ARE MAINLY USED FOR COMMANDS SUCH AS `enumerate treasure`

HIERARCHY

- FURTHERMORE, MUDDLE CLASSES CAN HAVE **MULTIPLE** PARENTS

```
longsword:
*+          [sword, undamageable]
            desc:
                        loose(first) ->>
                                    "A murderous longsword glints ahead of you. ",
                                    "Thrust deep into a rock is a murderous longsword! "
            strength: muser(outside(first) 'o') | spellproof(o) ->> 30, 60
            loose: \\
            prop: \\
            luminescent: //
+*
```

- HERE, THE `longsword` IS **BOTH** `sword` **AND** `undamageable`
- SWORD (DEFINED ELSEWHERE) IS ITSELF `metal`, `weapon`, `treasure` AND `loosener`

MATCHING

- WHEN YOU MATCH A **COMMAND** TO A **PATTERN**, YOU MATCH THE MOST LEFT-TO-RIGHT **SPECIFIC**
- FOR EXAMPLE, ROOMS AND CREATURES ARE BOTH **CONTAINERS**
- THE `room` AND `creature` CLASSES ARE THUS **MORE** SPECIFIC THAN THE `container` CLASS
  - `get ls f here` WILL MATCH `{ get longsword room }` BEFORE `{ get longsword container }`
  - `get ls from box` WILL ONLY MATCH `{ get longsword container }`

TANGLED

- MUD2'S OBJECT HIERARCHY WAS SOMETHING LIKE **14** LEVELS DEEP AND HAD **THOUSANDS** OF ATOMS IN IT
- **SOME** ATOMS HAD **50+** CHILDREN
  - TRANSLATION: SOME CLASSES HAD 50+ SUBCLASSES
- YOU MIGHT THINK THIS WOULD BE A HORRIBLE **TANGLE** YOU COULD **NEVER** KEEP TRACK OF
- YOU'D BE RIGHT – IT WAS!
- HOWEVER, YOU DIDN'T **NEED** TO UNDERSTAND IT
- IT HANDLED THE TANGLED MESS **FOR** YOU

CODE

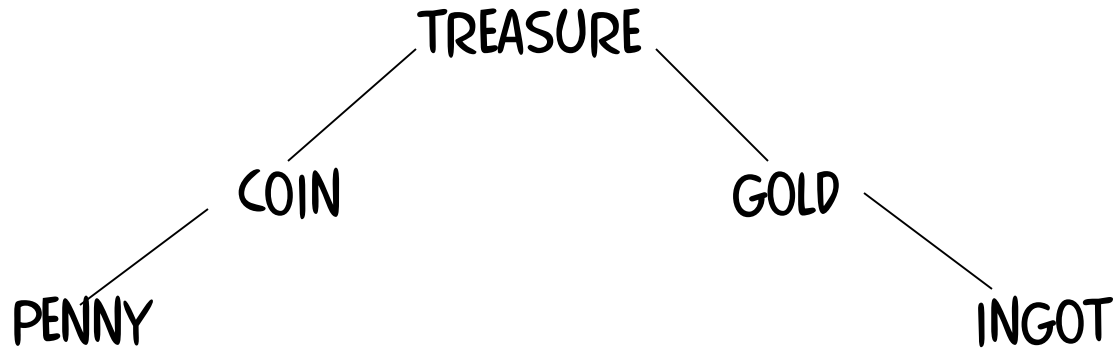
- THE CODE ASSOCIATED WITH PATTERNS **LOOKS** LIKE NORMAL CODE:

```
{ get longsword room }:
(second=outside(me) | checkwiz()) &
$(        the%(first) 'df'
          loose(first) ->> get%(first, second),
          muser(me) ->>
                    !! ("You can't seem to dislodge " + df + ", it won't budge.*N"),
          prop(first) ->>
          $(        checkcanhold(first)
                    loose(first):= //
                    !! ("You easily withdraw " + df + " from the rock.*N")
                    get%(first, second)
          $),
          $(        !! ("You take hold of " + df + " but its magical powers have*
faded, and it disintegrates in your hand.*N")
                    destroy%(first)
          $)
$)
```

- **ALL** THE FUNCTION CALLS IN THERE **ALSO** USE THE PATTERN-MATCHING SYSTEM

DIAMOND PROBLEM

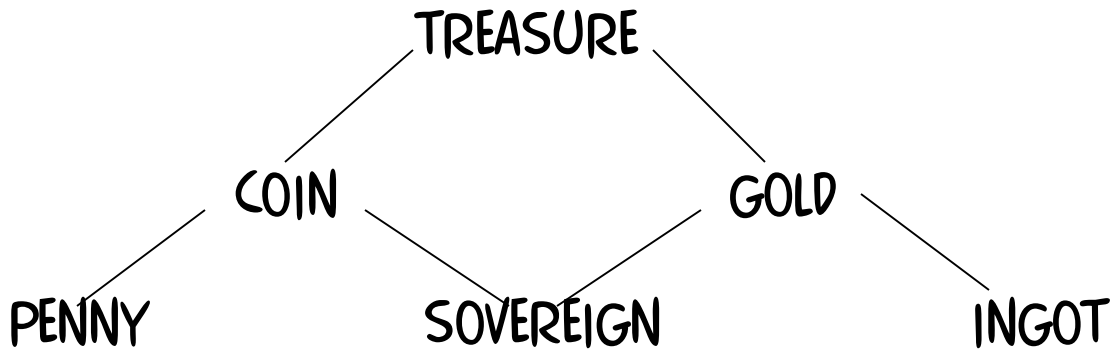
- HERE'S A **SINGLE-INHERITANCE** ATOM HIERARCHY



- IF WE DEFINE
  - `{ value treasure }: 100`
  - `{ value gold }: 200`
- THEN THE PENNY HAS A VALUE OF **100** AND THE INGOT HAS A VALUE OF **200**
- TRY **`get penny`**, **`get coin`** AND **`get treasure`**

MULTIPLE INHERITANCE

- WHAT HAPPENS IF YOU HAVE SOMETHING THAT IS BOTH A COIN **AND** AN ITEM OF GOLD?



- **`get gold`** AND **`get coin`** ARE NOW **IMPRESSIVE**
- HOWEVER, SUPPOSE WE DEFINE
  - `{ value gold }: 200`
  - `{ value coin }: 50`
- WHAT'S THE **VALUE** OF THE **SOVEREIGN**?

ANSWER

- THE ANSWER IS THAT IT **DOESN'T MATTER**!
- SO LONG AS THE PATTERN-MATCHER RETURNS THE **SAME** ANSWER **EVERY** TIME YOU USE IT, IT'S **OK**
  - SO BASICALLY, IF YOU STOP WHEN YOU FIND THE FIRST MATCH, YOU'RE **FINE**
- IT GENUINELY **IS** AMBIGUOUS – SO **EMBRACE** THAT AMBIGUITY!
- MUDDLE, LIKE **BCPL** BEFORE IT, **TRUSTS** THE PROGRAMMER

METHODS

- **ALMOST ALL** VIRTUAL WORLDS ASSOCIATE FUNCTIONALITY ("METHODS") WITH GAME OBJECTS
  - WORKS FOR **SINGLE-PARAMETER** COMMANDS
  - PROBLEMS FOR **MULTI-PARAMETER** COMMANDS
  - "TOUCH CANDLE WITH MATCH"
- SOLUTION: MAKE **VERBS** BE THE PROGRAMMING OBJECTS, NOT **NOUNS**
  - { TOUCH COMBUSTIBLE COMBUSTIBLE }:
- MOST JUST **HACK** IT, C++ OR JAVA STYLE...

CODE AND DATA

- THE **GENERAL** POINT I WANT TO MAKE CONCERNS **CODE** AND **DATA**
- WHAT'S THE **DIFFERENCE**?
- *MUD* VERSION 1 **HARD-CODED** ITS CONTENT
- VERSION 2 **SOFT-CODED** IT
- VERSION 3 COMPILED DATA-DEFINITION FILES INTO **ASSEMBLER**
- VERSION 4 CONVERTED DATA-DEFINITION FILES INTO CODE FOR A **VIRTUAL MACHINE**
- SCRIPTS ARE **DATA** PRESENTED AS **CODE**?

CONTINUUM

- YOU START OUT **HARD-CODING** DATA, THEN YOU MOVE IT OUT TO **FILES** FOR FLEXIBILITY
- THE MORE **CONTROL** YOU MOVE OUT TO FILES, THE MORE YOUR DATA LOOKS LIKE A **SCRIPT**
- THE MORE **POWER** YOUR SCRIPTS HAVE, THE MORE YOU CREATE A STAND-ALONE **LANGUAGE**
- IF YOU TAKE THIS THE **WHOLE** WAY, YOU END UP WITH **EVERYTHING** IN THE SCRIPT AND YOUR ORIGINAL CODE IS AN **INTERPRETER**
- BUT ... YOUR DATA IS NOW **HARD-CODED** IN THE **SCRIPTING** LANGUAGE!

HACK OR REFACTOR

- THIS IS A **GENERAL** PROBLEM WITH PROGRAMMING
- DO YOU **HACK** A SOLUTION, OR DO YOU **REFACTOR** EVERYTHING?
- IT MAKES SENSE TO DO **ONE** OR THE **OTHER**
- IT MAKES **NO** SENSE TO DO ANYTHING IN BETWEEN
  - YOUR ONLY LEGITIMATE **JUSTIFICATION** IS THAT YOU WEREN'T GIVEN ENOUGH **TIME** TO DO A **PROPER** JOB

CONCLUSION

- CODE AND DATA ARE THE **SAME THING**
- **CODE** IS MERELY **DATA** FOR **OTHER** CODE OR FOR HARDWARE
- PLAYING WITH COMPUTER GAME DESIGN FOR **FUN** CAN BE **MORE** THAN JUST FUN
  - A **MULTI-BILLION** POUND/DOLLAR/EURO/YUAN **INDUSTRY** CAME OUT OF ROY'S AND MY FUN!
- **COMPUTERS** TODAY ARE **NOT** AS THEY ONCE WERE, BUT **CREATIVITY** IS
- IF YOU **WANT** TO CODE SOMETHING FOR **FUN**, **CODE** IT FOR FUN!