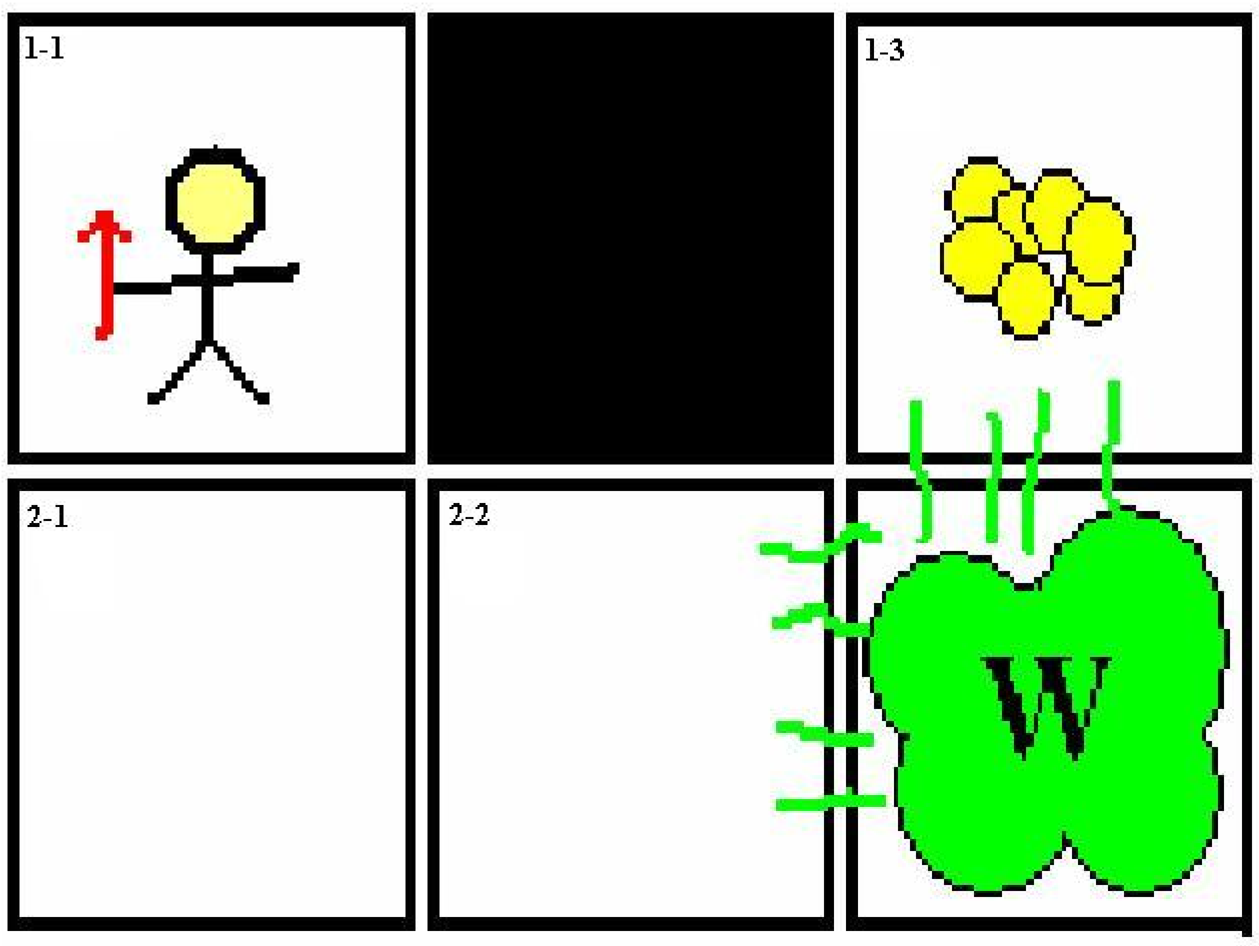
The Wumpus World



Modelling the Wumpus World in PDDL: 1st try...

```
(define (domain wumpus-a)
  (:requirements :strips) ;; maybe not necessary

  (:predicates
   (adj ?square-1 ?square-2)
   (pit ?square)
   (at ?what ?square)
   (have ?who ?what)
   (dead ?who))

  (:action move
    :parameters (?who ?from ?to)
    :precondition (and (adj ?from ?to)
                       (not (pit ?to))
                       (at ?who ?from))
    :effect (and (not (at ?who ?from))
                 (at ?who ?to))
    )

  (:action take
    :parameters (?who ?what ?where)
    :precondition (and (at ?who ?where)
                       (at ?what ?where))
    :effect (and (have ?who ?what)
                 (not (at ?what ?where)))
    )

  (:action shoot
    :parameters (?who ?where ?arrow ?victim ?where-victim)
    :precondition (and (have ?who ?arrow)
                       (at ?who ?where)
                       (at ?victim ?where-victim)
                       (adj ?where ?where-victim))
    :effect (and (dead ?victim)
                 (not (at ?victim ?where-victim))
                 (not (have ?who ?arrow)))
    )
)

(define (problem wumpus-a-1)
  (:domain wumpus-a)
  (:objects
   sq-1-1 sq-1-2 sq-1-3
   sq-2-1 sq-2-2 sq-2-3
   the-gold
   the-arrow
   agent
   wumpus)

  (:init
   (adj sq-1-1 sq-1-2) (adj sq-1-2 sq-1-1)
   (adj sq-1-2 sq-1-3) (adj sq-1-3 sq-1-2)
   (adj sq-2-1 sq-2-2) (adj sq-2-2 sq-2-1)
   (adj sq-2-2 sq-2-3) (adj sq-2-3 sq-2-2)
   (adj sq-1-1 sq-2-1) (adj sq-2-1 sq-1-1)
   (adj sq-1-2 sq-2-2) (adj sq-2-2 sq-1-2)
   (adj sq-1-3 sq-2-3) (adj sq-2-3 sq-1-3)

   (pit sq-1-2)

   (at the-gold sq-1-3)
   (at agent sq-1-1)
   (have agent the-arrow)
   (at wumpus sq-2-3))

  (:goal (and (have agent the-gold) (at agent sq-1-1)))
  )
```

Resulting plan:

```
(MOVE THE-GOLD SQ-1-3 SQ-2-3)
(MOVE THE-GOLD SQ-2-3 SQ-2-2)
(MOVE THE-GOLD SQ-2-2 SQ-2-1)
(MOVE THE-GOLD SQ-2-1 SQ-1-1)
(TAKE AGENT THE-GOLD SQ-1-1)
```

Modelling the Wumpus World in PDDL: 2nd try...

```
(define (domain wumpus-b)
  (:requirements :strips)
  (:predicates
   (adj ?square-1 ?square-2)
   (pit ?square)

   (at ?what ?square)
   (have ?who ?what)

   (takeable ?what)
```

```
     (is-gold ?what)
     (is-arrow ?what)

     (alive ?who)
     (dead ?who))

   (:action move
     :parameters (?who ?from ?to)
     :precondition (and (alive ?who)
                        (at ?who ?from)
                        (adj ?from ?to)
                        (not (pit ?to)))
     :effect (and (not (at ?who ?from))
                  (at ?who ?to))
     )

   (:action take
     :parameters (?who ?what ?where)
     :precondition (and (alive ?who)
                        (takeable ?what)
                        (at ?who ?where)
                        (at ?what ?where))
     :effect (and (have ?who ?what)
                  (not (at ?what ?where)))
     )

   (:action shoot
     :parameters (?who ?where ?arrow ?victim ?where-victim)
     :precondition (and (alive ?who)
                        (have ?who ?arrow)
                        (is-arrow ?arrow)
                        (at ?who ?where)
                        (alive ?victim)
                        (at ?victim ?where-victim)
                        (adj ?where ?where-victim))
     :effect (and (dead ?victim)
                  (not (alive ?victim))
                  (not (at ?victim ?where-victim))
                  (not (have ?who ?arrow)))
     )
 )

 (define (problem wumpus-b-1)
   (:domain wumpus-b)
   (:objects sq-1-1 sq-1-2 sq-1-3
             sq-2-1 sq-2-2 sq-2-3
             the-gold the-arrow
             agent wumpus)
   (:init (adj sq-1-1 sq-1-2) (adj sq-1-2 sq-1-1)
          (adj sq-1-2 sq-1-3) (adj sq-1-3 sq-1-2)
          (adj sq-2-1 sq-2-2) (adj sq-2-2 sq-2-1)
          (adj sq-2-2 sq-2-3) (adj sq-2-3 sq-2-2)
          (adj sq-1-1 sq-2-1) (adj sq-2-1 sq-1-1)
          (adj sq-1-2 sq-2-2) (adj sq-2-2 sq-1-2)
          (adj sq-1-3 sq-2-3) (adj sq-2-3 sq-1-3)
          (pit sq-1-2)
          (at the-gold sq-1-3)
          (is-gold the-gold)
          (takeable the-gold)
          (at agent sq-1-1)
          (alive agent)
          (have agent the-arrow)
          (is-arrow the-arrow)
          (takeable the-arrow)
          (at wumpus sq-2-3)
```

```
         (alive wumpus))
  (:goal (and (have agent the-gold)
              (at agent sq-1-1)
         ))
  )
```

Resulting plan:

```
(MOVE AGENT SQ-1-1 SQ-2-1)
(MOVE AGENT SQ-2-1 SQ-2-2)
(MOVE AGENT SQ-2-2 SQ-2-3)
(MOVE AGENT SQ-2-3 SQ-1-3)
(TAKE AGENT THE-GOLD SQ-1-3)
(MOVE AGENT SQ-1-3 SQ-2-3)
(MOVE AGENT SQ-2-3 SQ-2-2)
(MOVE AGENT SQ-2-2 SQ-2-1)
(MOVE AGENT SQ-2-1 SQ-1-1)
```

Modelling the Wumpus World in PDDL: 3rd time's a charm...

```
(define (domain wumpus-c)
  (:requirements :strips)
  (:predicates
   (at ?what ?square)
   (adj ?square-1 ?square-2)
   (pit ?square)
   (wumpus-in ?square)
     ;; <-> (exists ?x (and (is-wumpus ?x) (at ?x ?square) (not (dead ?x))
   (have ?who ?what)
   (is-agent ?who)
   (is-wumpus ?who)
   (is-gold ?what)
   (is-arrow ?what)
   (dead ?who))

  (:action move-agent
    :parameters (?who ?from ?to)
    :precondition (and (is-agent ?who)
                       (at ?who ?from)
                       (adj ?from ?to)
                       (not (pit ?to))
                       (not (wumpus-in ?to)))
    :effect (and (not (at ?who ?from))
                 (at ?who ?to))
    )

  (:action take
    :parameters (?who ?what ?where)
    :precondition (and (is-agent ?who)
                       (at ?who ?where)
                       (at ?what ?where))
    :effect (and (have ?who ?what)
                 (not (at ?what ?where)))
    )

  (:action shoot
    :parameters (?who ?where ?with-what ?victim ?where-victim)
    :precondition (and (is-agent ?who)
                       (have ?who ?with-what)
                       (is-arrow ?with-what)
                       (at ?who ?where)
                       (is-wumpus ?victim)
                       (at ?victim ?where-victim)
```

```
                          (adj ?where ?where-victim))
     :effect (and (dead ?victim)
                  (not (wumpus-in ?where-victim))
                  (not (have ?who ?with-what)))
     )

   (:action move-wumpus
     :parameters (?who ?from ?to)
     :precondition (and (is-wumpus ?who)
                        (at ?who ?from)
                        (adj ?from ?to)
                        (not (pit ?to))
                        (not (wumpus-in ?to)))
     :effect (and (not (at ?who ?from))
                  (at ?who ?to)
                  (not (wumpus-in ?from))
                  (wumpus-in ?to))
     )

 )

 (define (problem wumpus-c-1)
   (:domain wumpus-c)
   (:objects sq-1-1 sq-1-2 sq-1-3
             sq-2-1 sq-2-2 sq-2-3
             the-gold the-arrow
             agent wumpus)
   (:init (adj sq-1-1 sq-1-2) (adj sq-1-2 sq-1-1)
          (adj sq-1-2 sq-1-3) (adj sq-1-3 sq-1-2)
          (adj sq-2-1 sq-2-2) (adj sq-2-2 sq-2-1)
          (adj sq-2-2 sq-2-3) (adj sq-2-3 sq-2-2)
          (adj sq-1-1 sq-2-1) (adj sq-2-1 sq-1-1)
          (adj sq-1-2 sq-2-2) (adj sq-2-2 sq-1-2)
          (adj sq-1-3 sq-2-3) (adj sq-2-3 sq-1-3)
          (pit sq-1-2)
          (is-gold the-gold)
          (at the-gold sq-1-3)
          (is-agent agent)
          (at agent sq-1-1)
          (is-arrow the-arrow)
          (have agent the-arrow)
          (is-wumpus wumpus)
          (at wumpus sq-2-3)
          (wumpus-in sq-2-3))
   (:goal (and (have agent the-gold) (at agent sq-1-1)))
   )
```

Resulting plan:

```
(MOVE-AGENT AGENT SQ-1-1 SQ-2-1)
(MOVE-AGENT AGENT SQ-2-1 SQ-2-2)
(SHOOT AGENT SQ-2-2 THE-ARROW WUMPUS SQ-2-3)
(MOVE-AGENT AGENT SQ-2-2 SQ-2-3)
(MOVE-AGENT AGENT SQ-2-3 SQ-1-3)
(TAKE AGENT THE-GOLD SQ-1-3)
(MOVE-AGENT AGENT SQ-1-3 SQ-2-3)
(MOVE-AGENT AGENT SQ-2-3 SQ-2-2)
(MOVE-AGENT AGENT SQ-2-2 SQ-2-1)
(MOVE-AGENT AGENT SQ-2-1 SQ-1-1)
```

Modelling the Wumpus World in PDDL: using ADL...

```
(define (domain wumpus-adl)
  (:requirements :adl :typing)

  ;; object types
  (:types agent wumpus gold arrow square)

  (:predicates
   (adj ?square-1 ?square-2 - square)
   (pit ?square - square)
   (at ?what ?square)
   (have ?who ?what)
   (alive ?who))

  (:action move
    :parameters (?who - agent ?from - square ?to - square)
    :precondition (and (alive ?who)
                       (at ?who ?from)
                       (adj ?from ?to)
                        )
    :effect (and (not (at ?who ?from))
                 (at ?who ?to)

                 (when (pit ?to)
                   (and (not (alive ?who))))

                 (when (exists (?w - wumpus) (and (at ?w ?to) (alive ?w)))
                   (and (not (alive ?who)))))
    )

  (:action take
    :parameters (?who - agent ?where - square ?what)
    :precondition (and (alive ?who)
                       (at ?who ?where)
                       (at ?what ?where))
    :effect (and (have ?who ?what)
                 (not (at ?what ?where)))
    )

  (:action shoot
    :parameters (?who - agent ?where - square ?with-arrow - arrow
                 ?victim - wumpus ?where-victim - square)
    :precondition (and (alive ?who)
                       (have ?who ?with-arrow)
                       (at ?who ?where)
                       (alive ?victim)
                       (at ?victim ?where-victim)
                       (adj ?where ?where-victim))
    :effect (and (not (alive ?victim))
                 (not (have ?who ?with-arrow)))
    )
)

(define (problem wumpus-adl-1)
  (:domain wumpus-adl)

  (:objects
   sq-1-1 sq-1-2 sq-1-3 sq-2-1 sq-2-2 sq-2-3 - square
   the-gold - gold
   the-arrow - arrow
   agent-1 - agent
   wumpus-1 - wumpus)

  (:init (adj sq-1-1 sq-1-2) (adj sq-1-2 sq-1-1)
         (adj sq-1-2 sq-1-3) (adj sq-1-3 sq-1-2)
         (adj sq-2-1 sq-2-2) (adj sq-2-2 sq-2-1)
```

```
        (adj sq-2-2 sq-2-3) (adj sq-2-3 sq-2-2)
        (adj sq-1-1 sq-2-1) (adj sq-2-1 sq-1-1)
        (adj sq-1-2 sq-2-2) (adj sq-2-2 sq-1-2)
        (adj sq-1-3 sq-2-3) (adj sq-2-3 sq-1-3)
        (pit sq-1-2)
        (at the-gold sq-1-3)
        (at agent-1 sq-1-1)
        (alive agent-1)
        (have agent-1 the-arrow)
        (at wumpus-1 sq-2-3)
        (alive wumpus-1))

  (:goal (and (have agent-1 the-gold) (at agent-1 sq-1-1) (alive agent-1)))
  )
```