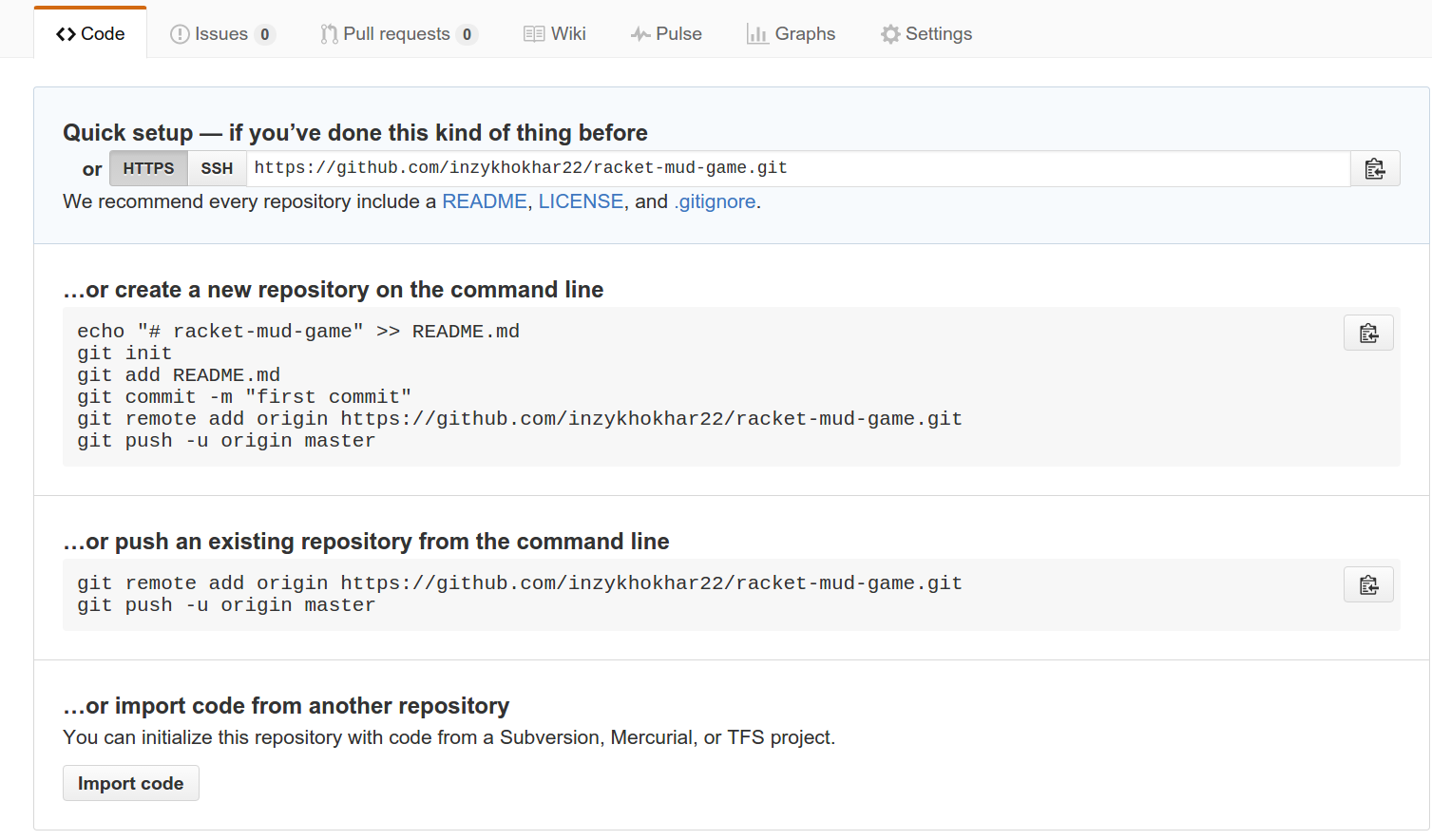
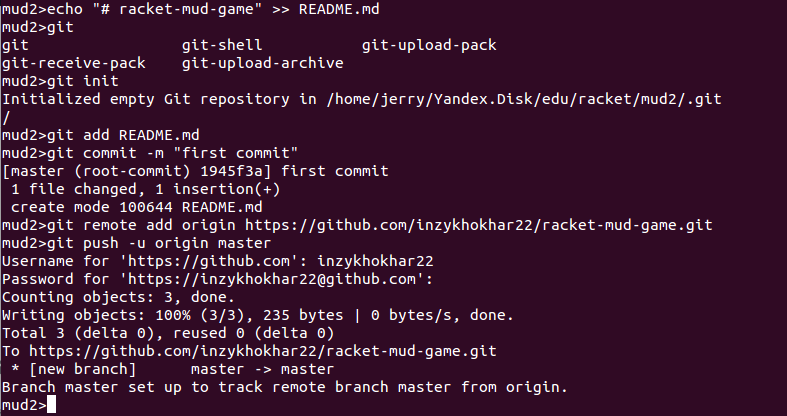
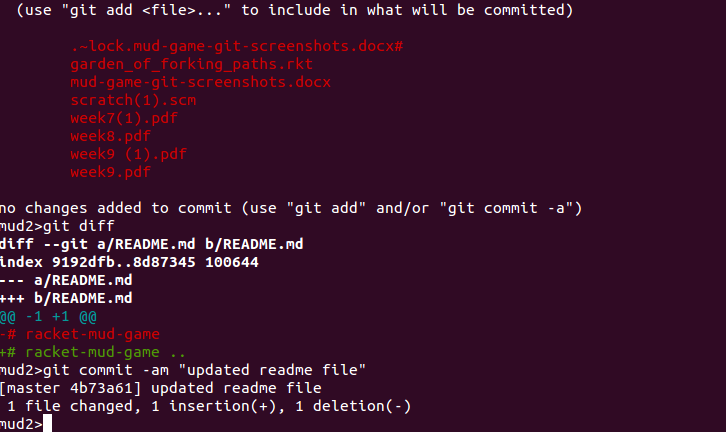
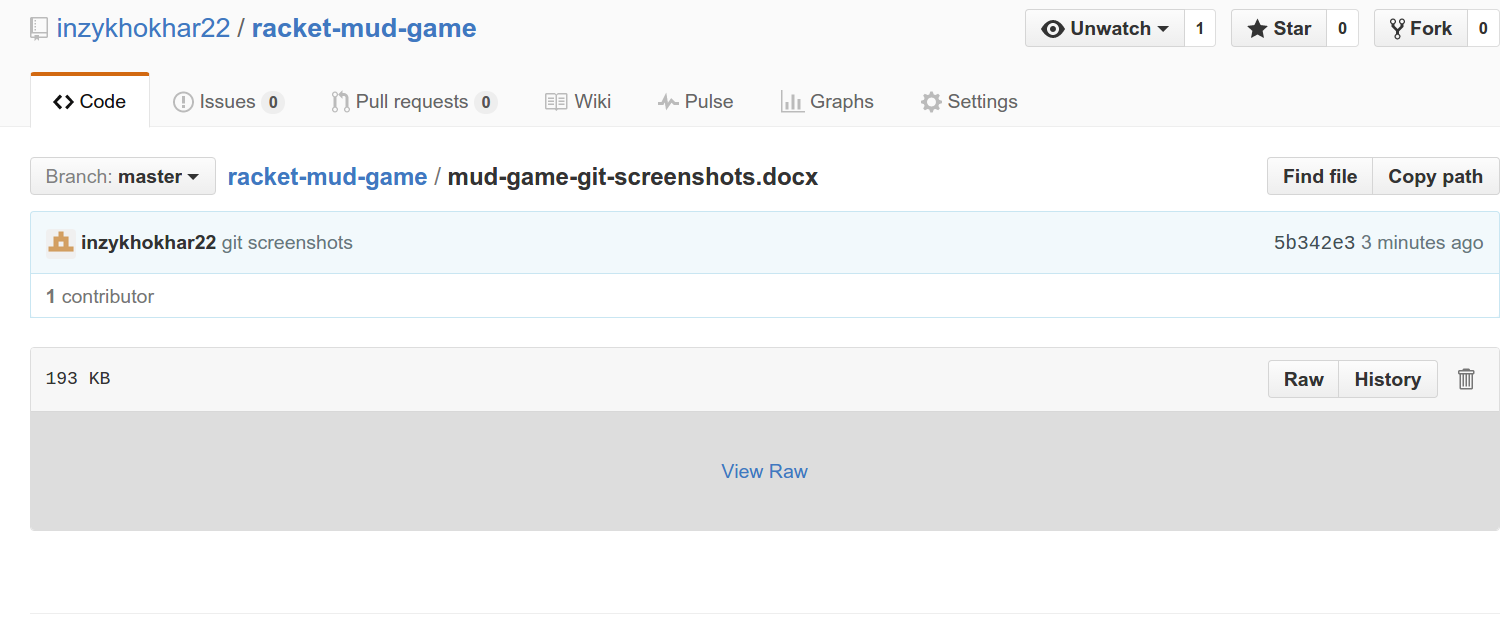
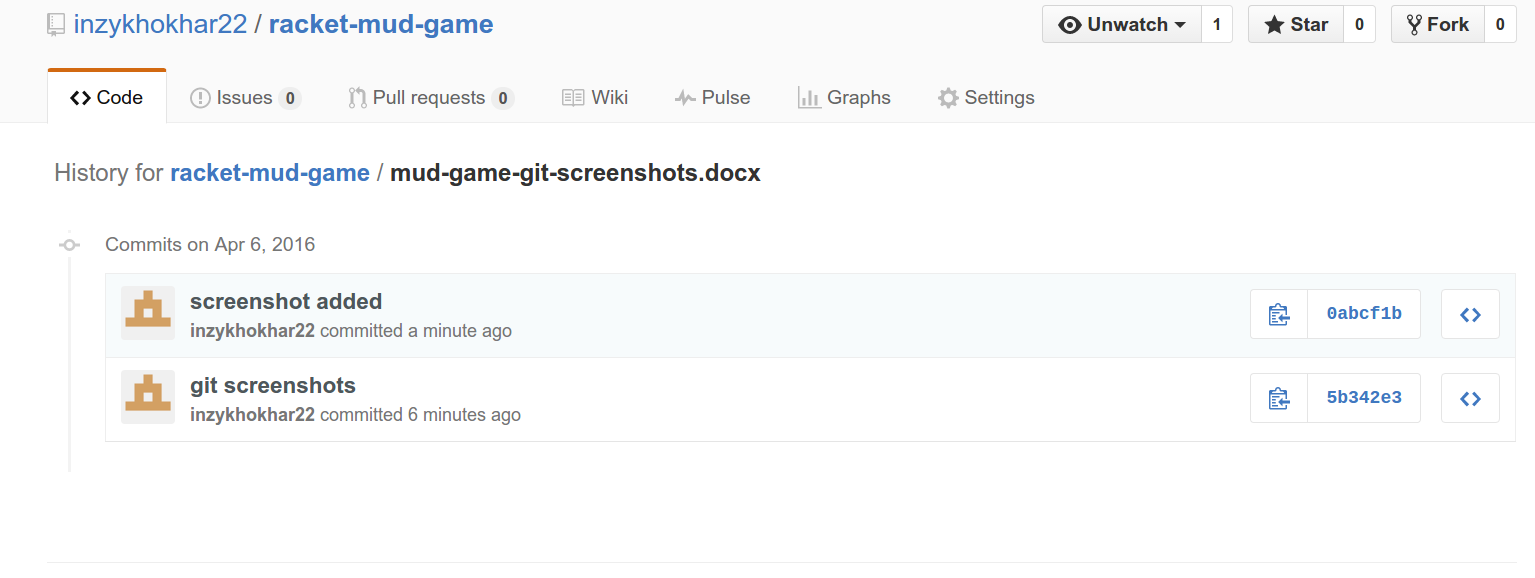
Github part:

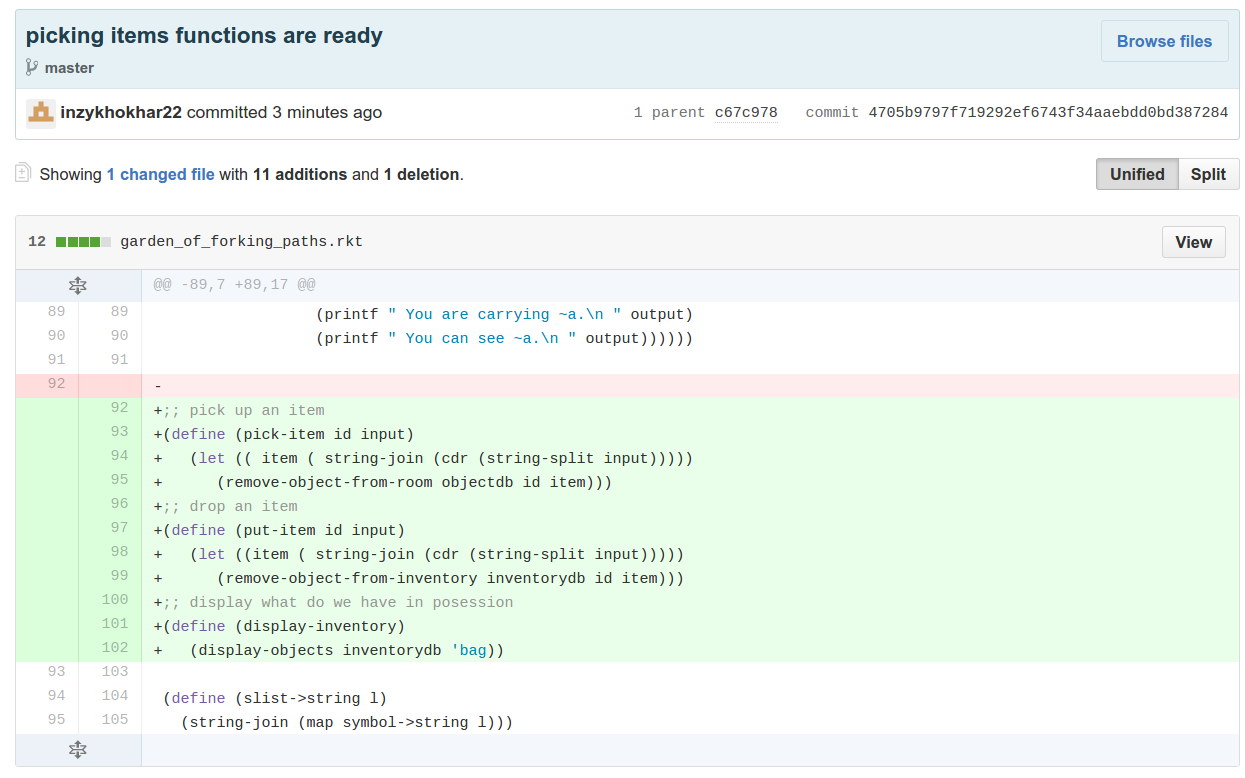


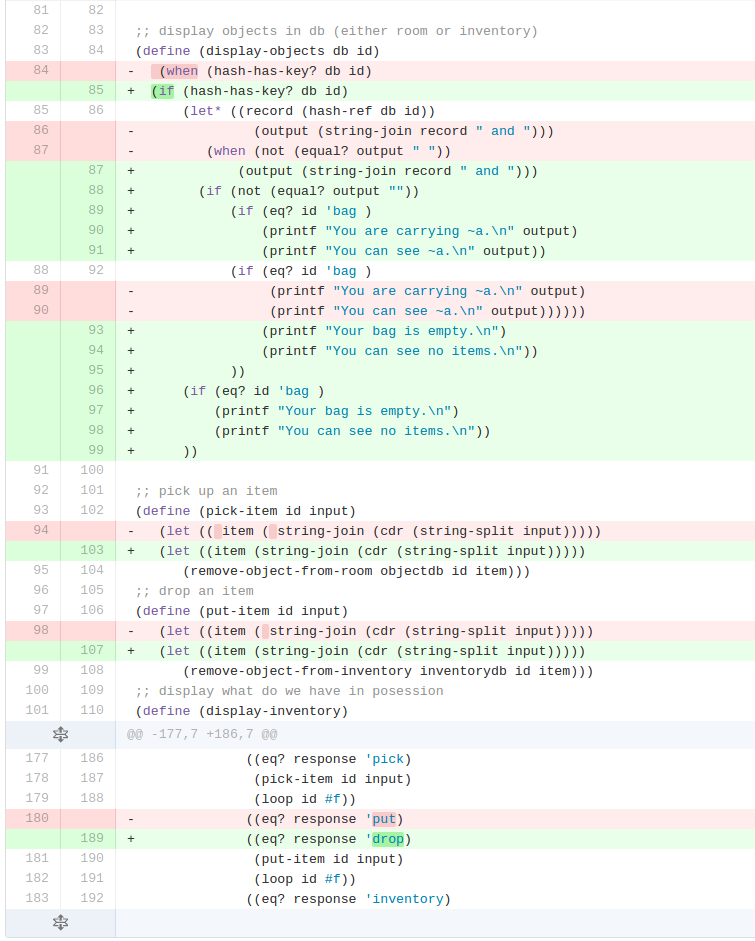
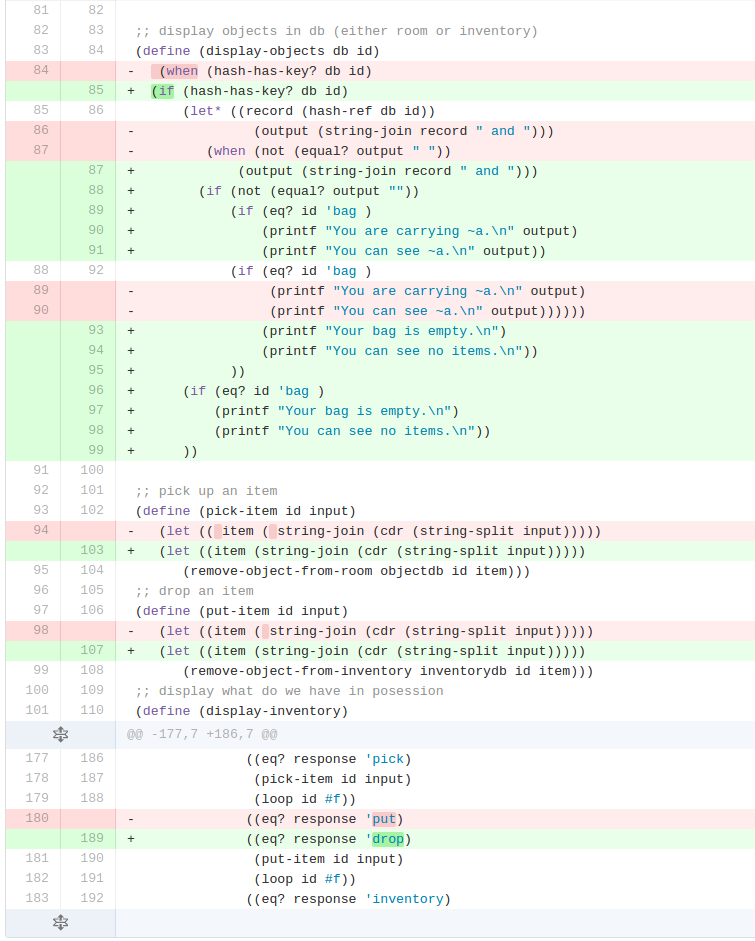












Code explanation:

Describe rooms of the MUD game in descriptions list. Every room description is a list of room id and a text.

;; world map

(define descriptions '((1 "You are in the lobby")

(2 "You are in the hallway")

(3 "You are in a swamp")))

Describe game items.

(define objects '((1 "a silver dagger")

(1 "a gold coin")))

Objects are lists with room id and item description. So if we want to put a silver dagger to the swamp , we need to change (1 “a silver dagger”) to (3 “a silver dagger”).

Describe possible actions.

;; define actions that are possible in rooms

(define look '(((directions) look) ((look) look) ((examine room) look)))

(define pick '(((get) pick) ((pickup) pick) ((pick) pick)))

(define put '(((put) drop) ((drop) drop) ((place) drop) ((remove) drop )))

(define inventory '(((inventory) inventory) ((bag) inventory)))

(define quit '(((exit game) quit) ((quit game) quit) ((exit) quit) ((quit) quit)))

These lists describe actions that are possible in the game as well as their synonyms. So we see that we can drop or put item to free your bag of it.

Now we should join actions list to one actions list. We do this with help of quasiquote (`) and unquote-splicing (,@).

(define actions `(,@look ,@pick ,@put ,@inventory ,@quit))

or

(define actions (quasiquote ((unquote-splicing look) (unquote-splicing pick) (unquote-splicing put) (unquote-splicing inventory) (unquote-splicing quit))))

quasiquote works like quote but includes uncoute and unquote-splicing functions inside of it. What we did in previous define actions is to take items from the lists and join them in new quoted list named actions:

> (pretty-print actions)

'(((directions) look)

((look) look)

((examine room) look)

((get) pick)

((pickup) pick)

((pick) pick)

((put) drop)

((drop) drop)

((place) drop)

((remove) drop)

((inventory) inventory)

((bag) inventory)

((exit game) quit)

((quit game) quit)

((exit) quit)

((quit) quit))

Look we now have plain list with ((synonym) action) structures.

Next we should describe connections between rooms:

;; directions map

(define decisiontable `((1 ((north) 2) ((north west) 3) ,@actions)

(2 ((south) 1) ,@actions)

(3 ,@actions)))

We use quasiquote again to get one big list:

> (pretty-print decisiontable)

'((1

((north) 2)

((north west) 3)

((directions) look)

((look) look)

((examine room) look)

((get) pick)

((pickup) pick)

((pick) pick)

((put) drop)

((drop) drop)

((place) drop)

((remove) drop)

((inventory) inventory)

((bag) inventory)

((exit game) quit)

((quit game) quit)

((exit) quit)

((quit) quit))

(2

((south) 1)

((directions) look)

((look) look)

((examine room) look)

((get) pick)

((pickup) pick)

((pick) pick)

((put) drop)

((drop) drop)

((place) drop)

((remove) drop)

((inventory) inventory)

((bag) inventory)

((exit game) quit)

((quit game) quit)

((exit) quit)

((quit) quit))

(3

((directions) look)

((look) look)

((examine room) look)

((get) pick)

((pickup) pick)

((pick) pick)

((put) drop)

((drop) drop)

((place) drop)

((remove) drop)

((inventory) inventory)

((bag) inventory)

((exit game) quit)

((quit game) quit)

((exit) quit)

((quit) quit)))

>

Every item in this list is a room with possible commands in it. Look the room number 1 has commands:

(1

((north) 2)

((north west) 3)

what means room 6 has exits to rooms 2 and 3 or hallway and swamp.

And see the swamp does not have exits, it is a trap!

So the word is described, now we need set of functions to execute actions in it. See their description in the code please.

Game screenshots:

