Constructor Demo

Adrian Seto, Omar Baranek, Dhruv Mittal

This document contains instructions for running the Constructor program. Whilst it is not an extensive testing suite, it covers instructions to demo all the features required.

Command Line Arguments

We were able to implement all the command line options required for the project.

Load game (-load xxx)

Normal check:

1. ./constructor -load savefile.txt
2. Enter "roll" for Blue builder's turn
3. Input "2"
4. Input "status"
5. Input "residence"
6. Repeat this 3 more times to confirm that the text file was loaded correctly (game board display as expected, houses, roads, resources saved correctly)

Load custom board (-board xxx)

Normal check:

1. ./constructor -board custom_board.txt
2. The board should be correctly laid out as described in layout.txt

Loading an unreadable file:

1. ./constructor -board unreadable_layout.txt
2. The program should tell the user that they can not open the layout file (excuse the "BoardModel::initBoard", we were meant to edit that but ran out of time)

Loading an incorrectly formatted layout (too many tiles):

1. ./constructor -board too_many_tiles.txt
2. Expect an error message

Loading an incorrectly formatted layout (too few tiles):

1. ./constructor -board too_few_tiles.txt
2. Expect an error message

Random Board (-random-board)

Without seed (true random):

1. ./constructor -random-board
2. Expect a random board to be printed in the correct format

With seed (should replicate results):

1. ./constructor -random-board -seed 123456789
2. Expect a random board with a park on tile 11 and geese placed there for a new game
3. Exit the game with control c (we will test save later)
4. Rerun the program with the same command in step 1, you should see the same board

Other tests

No command line options:

1. ./constructor
2. Board should be generated through layout.txt
3. Continue playing the game as usual if you want

Beginning of the Game

Able to build initial 8 houses:

1. ./constructor
2. Enter a number to build the first residence
3. Repeat 7 more times, noting that the builders are rotated in the correct order
4. After all the houses have been built, you should see the game board with all the correct basements build
5. Enter “roll” to roll a loaded dice by default
6. Enter 5
7. Enter “residence” to see all the houses built
8. Enter “status” to see that **no resources were used to build initial houses**
9. Feel free to continue playing the game to check for other builders too

Beginning of Turn

Fair dice generates a random number (true random):

1. ./constructor -load savefile.txt
2. Enter “fair” to set blue builder’s dice to fair
3. Enter “roll” to roll the fair dice
4. Your dice roll should show along with the resources obtained from the roll
5. Enter “next”
6. Repeat steps 2 to 5 3 more times. You should see true random numbers being rolled by the dice

Fair dice generates random number (seed):

1. ./constructor -load savefile.txt -seed 123456789
2. Enter “fair” to set blue builder’s dice to fair
3. Enter “roll” to roll the fair dice
4. Your dice roll should show along with the resources obtained from the roll
5. Enter “next”
6. Repeat steps 2 to 5 2 more times. You should see a the following dice rolls:
    a. Expected dice rolls: 3, 9, 10

Setting dice persists over turns:

1. ./constructor -load savefile.txt -seed 123456789
2. Enter “fair” to set blue builder’s dice to fair
3. Enter “roll” to roll the fair dice
4. Your dice roll should show along with the resources obtained from the roll
5. Enter “next”
6. Repeat steps 2 to 5 3 more times.
7. Once you return to Blue builder, simply enter “roll”, a random dice should roll from the previous time you set Blue builder’s dice to “fair”

Setting dice multiple times before rolling:

1. ./constructor -load savefile.txt -seed 123456789
2. Enter “fair”
3. Enter “load”
4. Enter “roll”, the dice should still be a loaded dice despite being set to fair in the intermediary step.

Range check for loaded dice roll:

1. ./constructor -load savefile.txt
2. Enter “roll”

3. Input “100”, you should get an error message and get asked again
4. Input “-5”, you should get an error message and get asked again

During the Turn Commands

Board (board)

Printing the board has been tested before and will be further tested through other functionalities.

Status (status)

Status has been tested before and will be further tested through other functionalities.

Residences (residences)

Residence has been tested before and will be further tested through other functionalities.

Build Road (build-road <road#>)

Normal test (adjacent residence and then adjacent road):

1. ./constructor -load 100_resources_file.txt
2. Enter “roll” to roll a loaded dice for Blue
3. Input “6” to roll a 6
4. Enter “status” to see the current resource level
5. Enter “build-road 55”
6. Enter “status”, you should see 1 heat and 1 wifi decreased
7. Enter “board”, you should see “BR” on edge 55
8. Enter “build-road 63”
9. Enter “board”, another road should have been built on Edge 63

Bounds check:

1. ./constructor -load savefile.txt
2. Enter “roll”
3. Enter “4”
4. Enter “build-road 100”
5. Enter “build-road -5”

Not enough resources:

1. ./constructor -load savefile.txt
2. Enter “roll” to roll a dice for Blue
3. Enter “2” to roll a 2
4. Enter “status” to see the current resources of Blue builder
5. Enter “build-road 55”, You should see error message

Don’t have adjacent edge or residence:

1. ./constructor -load 100_resources_file.txt
2. Enter “roll”
3. Enter “5”
4. Enter “build-road 0”, you should get error message

Building on an edge with an existing road:

1. ./constructor -load 100_resources_file.txt
2. Enter “roll”
3. Enter “5”
4. Enter “build-road 47”, you should get error message

Trying to build a road through another person’s residence:

1. ./constructor -load build_through_res.txt
2. Enter “roll”
3. Enter “4”
4. Enter “build-road 39”, should give you an error message

Build Residence (build-res <housing#>)

Normal test:

1. ./constructor -load build_res_normal.txt
2. Enter “roll”
3. Enter “9”
4. Enter “status” to see current resources
5. Enter “build-res 48” to build a residence on vertex 48
6. Enter “status” to see resources after building basement
7. Enter “board” to see the changes on the board

Not enough resources:

1. ./constructor -load build_res_no_resources.txt
2. Enter “roll”
3. Enter “4”
4. Enter “build-res 48” to build a res on vertex 48, you should see an error

Bound checks on input:

6. ./constructor -load savefile.txt
7. Enter “roll”
8. Enter “4”
9. Enter “build-res 500”
10. Enter “build-res -5”

Build on vertex with adjacent residence:

1. ./constructor -load build_res_normal.txt
2. Enter “roll”
3. Enter “6”
4. Enter “build-res 31”, you should get error message

No road connecting to the residence:

1. ./constructor -load savefile.txt
2. Enter “roll” to roll a loaded dice for blue builder
3. Enter “4”
4. Enter “build-res 42”, you should get an error message

Improve Residence (improve <housing#>)

Normal test (improve basement -> tower): (show resources subtracting in cost and obtain resources gaining more resources)

8. ./constructor -load build_res_normal.txt
9. Enter “roll”
10. Enter “9”
11. Enter “build-res 48” to build a residence on vertex 48
12. Enter “status” to see resources after building basement
13. Enter “board” to see the changes on the board
14. Enter “improve 48”
15. Enter “status” to see resources were subtracted
16. Enter “board” to see change in UI
17. Enter “next” to go to next builder
18. Enter “roll”
19. Enter “6” to see “BH” obtain 5 WIFI instead of 4 before (Tower + House)
20. Keep rolling 6 and iterating builders until you reach Blue builder again
21. Enter “status” to see resources before improvement
22. Enter “improve 48”
23. Enter “status” to see resources were subtracted
24. Enter “board” to see change in UI

25. Enter “next” to go to next builder
26. Enter “roll”
27. Enter “6” to see that Blue gained 6 WIFI

Improve Tower:

1. ./constructor -load build_res_normal.txt
2. Enter “roll”
3. Enter “9”
4. Enter “build-res 48” to build a residence on vertex 48
5. Enter “improve 48”
6. Enter “improve 48”
7. Enter “improve 48”, you should get error message

Not enough resources:

1. ./constructor -load build_res_no_resources.txt
2. Enter “roll”
3. Enter “9”
4. Enter “improve 30”, you should get error message

Improving residence not owned by builder:

1. ./constructor -load build_res_normal.txt
2. Enter “roll”
3. Enter “9”
4. Enter “improve 32”, you should get an error

Bounds check:

1. ./constructor -load build_res_normal.txt
2. Enter “roll”
3. Enter “9”
4. Enter “improve 100”, you should get an error
5. Enter “improve -5”, you should get an error

Play Goose (roll a loaded dice to 7)

Whilst we implemented all the features of the goose move as needed, we ran out of time to make a demo. Please explore the game to your liking and roll a 7 on loaded dice to check out our lovely geese feature.

Trade (trade <colour> <give> <take>)

Normal (successful trade):

1. ./constructor -load 100_resources_file.txt

2. Enter “roll”
3. Enter “10”
4. Enter “status” to see current resources of all builders
5. Enter “trade Red brick energy”
6. Enter “yes” for Red to accept the trade
7. Enter “status” to see the changes reflected

Normal (unsuccessful trade):

1. ./constructor -load 100_resources_file.txt
2. Enter “roll”
3. Enter “10”
4. Enter “status” to see current resources of all builders
5. Enter “trade Red brick energy”
6. Enter “no” for Red to reject the trade
7. Enter “status” to see the changes reflected

Invalid resources suggested for trade:

1. ./constructor -load 100_resources_file.txt
2. Enter “roll”
3. Enter “10”
4. Enter “trade Red brick steel”, you should see error message

Not enough resources:

1. ./constructor -load savefile.txt
2. Enter “roll”
3. Enter “6”
4. Enter “status” to see resource counts
5. Enter “trade Red brick energy”
6. Enter “yes” for Red to accept the trade, you should see an error message

Next (next)

Next has been tested before and will be further tested through other functionalities.

Save (save)

Save to named file

1. ./constructor -load build_res_normal.txt
2. Enter “roll”
3. Enter “9”
4. Enter “build-res 48” to build a residence on vertex 48
5. Enter “board” to see the changes on the board

6. Enter “save testSave.txt” to save to a text file
7. Enter command c to quit the game or command d to save to backup.sv

Save to backup on

1. ./constructor -load build_res_normal.txt
2. Enter “roll”
3. Enter “9”
4. Enter “build-res 48” to build a residence on vertex 48
5. Enter “board” to see the changes on the board
6. Enter command d to save to backup.sv

Load a backup file after saving to backup

1. ./constructor -load build_res_normal.txt
2. Enter “roll”
3. Enter “9”
4. Enter “build-res 48” to build a residence on vertex 48
5. Enter “board” to see the changes on the board
6. Enter command d to save to backup.sv
7. ./constructor -load backup.sv
8. Check that the board was correctly saved

Help (help)

Help at beginning of turn stage

1. ./constructor -load savefile.txt
2. Enter “help”

Help at during the turn stage

1. ./constructor -load savefile.txt
2. Enter “roll”
3. Enter “6”
4. Enter “help”

End of Game

Play again:

1. ./constructor -load endGame.txt
2. Enter “roll”
3. Enter “9”
4. Enter “build-res 47” to build a 10th residence for builder and win the game

5. Enter “yes” to play again. A new game will start from the beginning

Don’t play again:

1. ./constructor -load endGame.txt
2. Enter “roll”
3. Enter “9”
4. Enter “build-res 47” to build a 10th residence for builder and win the game
5. Enter “no” to end the program