Name: Haris Sohail

Date: 05/26/2018

Assignment 5

CS362

Bug Reports

BUG #1

Title: “Incorrect cards gained when Smithy is played”

Class:
"Serious Bug"

Date: 05/25/2018

Reported By: Haris Sohail

Email: sohailh@oregonstate.edu

Product: Dominion.c

Platform: Linux (Flip3)

Is it reproducible: Yes

Description

===========

Summary: When trying to play the Smithy card from player 1’s hand at the start of the game, the program adds the incorrect amount of cards to player 1’s hand.

What happened: First initiated game state with 5 cards in hand, including Smithy. Then called Smithy through function cardEffect(). This function then called playSmithy(). Game state returned 3 cards less than expected.

What’s the error: No cards are added to player’s hand after Smithy is called, whereas 3 should be.

Supporting Information: According to random test results this occurs on every single call to playSmithy() regardless of game state values.

Steps to Produce/Reproduce

This bug can be reproduced on any call to Smithy. Simply playing Smithy from the player’s hand will produce this bug.

Expected Results

It was expected to see a return of 3 additional cards in the player’s hand.

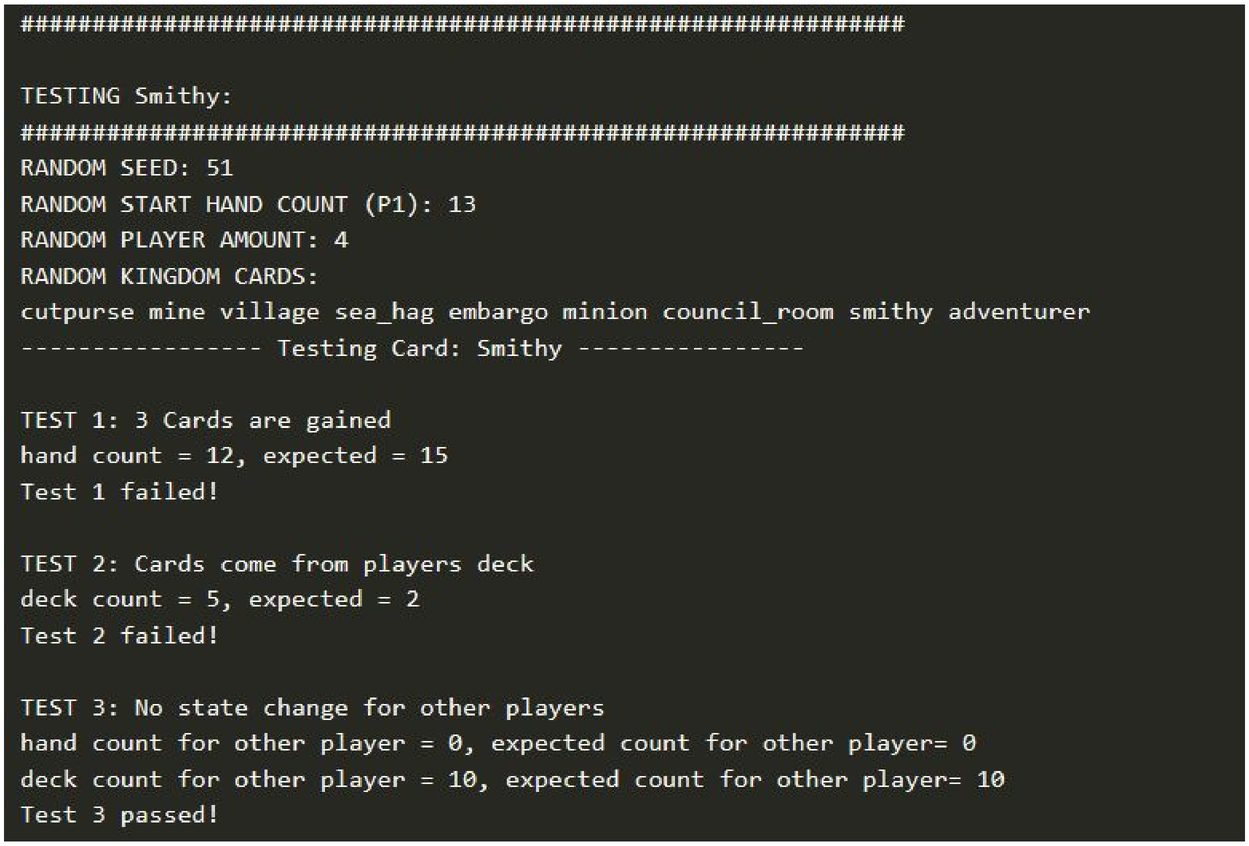
Actual Results

No additional cards were in the player’s hand after Smithy was called. Only 1 was gone which was correct since Smithy was discarded.

Workarounds

No workarounds possible, bug occurs on any call to Smithy.

Attachments



```
################################################################

TESTING Smithy:
################################################################
RANDOM SEED: 51
RANDOM START HAND COUNT (P1): 13
RANDOM PLAYER AMOUNT: 4
RANDOM KINGDOM CARDS:
cutpurse mine village sea_hag embargo minion council_room smithy adventurer
----------------- Testing Card: Smithy ----------------

TEST 1: 3 Cards are gained
hand count = 12, expected = 15
Test 1 failed!

TEST 2: Cards come from players deck
deck count = 5, expected = 2
Test 2 failed!

TEST 3: No state change for other players
hand count for other player = 0, expected count for other player= 0
deck count for other player = 10, expected count for other player= 10
Test 3 passed!
```

Other Information

N/A

BUG #2

Title: “Incorrect actions gained when Village is played”

Class:
"Serious Bug"

Date: 05/25/2018
Reported By: Haris Sohail
Email: sohailh@oregonstate.edu

Product: Dominion.c
Platform: Linux (Flip3)

Is it reproducible: Yes except when player action count equals 2.

Description
===========
Summary: When trying to play the Village card from player 1’s hand at the start of the game, the program adds the incorrect amount of actions back to the player 1’s current turn.

What happened: First initiated game state with 5 cards in hand, including Village. Then called Village through function cardEffect(). This function then called playVillage(). Game state determined player 1 to have only 1 action remaining when 2 were expected.

What’s the error: According to random testing the actions for the player are set to (currentActions * 2 - 1). If the player only has 1 action as in the unit test, this results in only 1 action being present when 2 are expected.

Supporting Information: This bug occurs with a standard game state as seen during unit tests, however during random testing it demonstrated as base player actions increase the actions added by village are nearly doubled each time.

Steps to Produce/Reproduce

This bug can be reproduced on any call to Village. Simply playing Village from the player's hand will produce this bug. However, this bug does not reproduce when player action count is at 2. When plugged into the bug formula this results in (2 * 2 - 1) which is equal to 3. This is also equal to the expected actions to be added to to 2 which is 1 if village worked correctly. This is a false positive.

Expected Results

It was expected to see an action of +1 to be added to the player's starting turn actions.

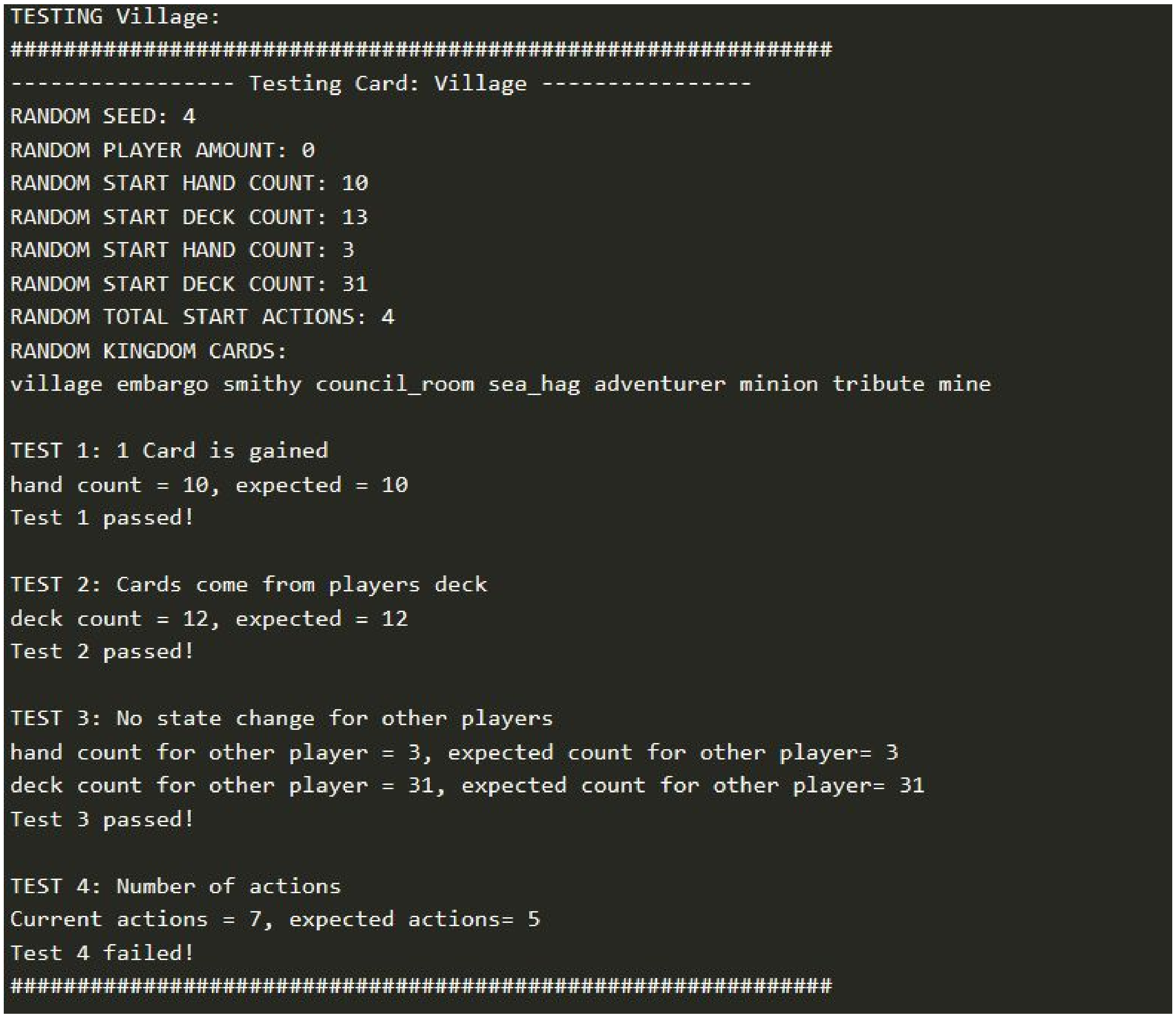
Actual Results

The actions added were based on the formula (currentActions * 2 - 1).

Workarounds

No workarounds possible, bug occurs on any call to Village except for when player actions equal 2.

Attachments



```
TESTING Village:
##############################################################
----------------- Testing Card: Village ----------------
RANDOM SEED: 4
RANDOM PLAYER AMOUNT: 0
RANDOM START HAND COUNT: 10
RANDOM START DECK COUNT: 13
RANDOM START HAND COUNT: 3
RANDOM START DECK COUNT: 31
RANDOM TOTAL START ACTIONS: 4
RANDOM KINGDOM CARDS:
village embargo smithy council_room sea_hag adventurer minion tribute mine

TEST 1: 1 Card is gained
hand count = 10, expected = 10
Test 1 passed!

TEST 2: Cards come from players deck
deck count = 12, expected = 12
Test 2 passed!

TEST 3: No state change for other players
hand count for other player = 3, expected count for other player= 3
deck count for other player = 31, expected count for other player= 31
Test 3 passed!

TEST 4: Number of actions
Current actions = 7, expected actions= 5
Test 4 failed!
##############################################################
```

Other Information

N/A

Interesting Bugs

The most interesting bug is definitely the gain in actions from the Village card. This is mainly due to how the type of testing, unit vs random, truly aided in pinpointing exactly what it was. According to unit testing, the village card was only adding 1 action instead of 2. Going solely on that would result in incorrect bug information. The random testing indicated a far more pressing issue. As actions increased a much larger number of actions were added. This helped in shaping what the bug was. Had we only used the unit tests, we would have incorrectly assumed the action of the bug. The cause of this bug is this line of code in playVillage(): state->numActions = state->numActions * 2; This bug was traceable due to being able to follow the steps taken by the function cardEffect.

The other bug inside Smithy has two defects causing its behavior. Fixing only 1 will cause the problems to remain. From inside the callSmithy() function:

```
int i = 3;

     for(i; i < 3; i--) {
      drawCard(currentPlayer, state);
   }
```

The first problem is i is set to 3. It should be set to 0. Next in the for loop, i-- should be i++. As it stands right nor the for loop never starts and breaks when 3 < 3 is evaluated to be false.

Test Report

Dominion testing results in tests working as expected only a partial amount of the time. There are several moving parts in this program where it can be hard to pinpoint exactly which part isn't working. Is the game class working incorrectly or is it a function incorrectly adding actions to the class? What tests our tests? How can we know for sure we didn't write any errors in our own programs causing false negatives or false positives? Testing Dominion has shown me that a single passing test does not mean that all tests pass. It may pass in one unit but can easily fail another. It has also taught me to unit test parts far removed from the function itself. Such as checking supply pile counts for each unit test to ensure no change has been made to them. This demonstrates a large coverage of testing and ensures a better filter of false positives and negatives.

The code provided from my teammate resulted in low coverage for some functions. playSmithy() for example has parts never executed due to the bug shown above. This coverage is the same despite random or unit testing. There is also a line of code not taken in playAdventurer resulting in lower coverage. This code will compile and run but is indeed buggy and unreliable.

Unit test code coverage:

```
Function 'playVillage'
Lines executed:100.00% of 5
No branches
Calls executed:100.00% of 2

Function 'playAdventurer'
Lines executed:93.75% of 16
Branches executed:100.00% of 12
Taken at least once:75.00% of 12
Calls executed:50.00% of 2
```

Function 'playSmithy'

Lines executed:83.33% of 6

Branches executed:100.00% of 2

Taken at least once:50.00% of 2

Calls executed:50.00% of 2

Random test coverage:

Function 'playVillage'

Lines executed:100.00% of 5

No branches

Calls executed:100.00% of 2

Function 'playAdventurer'

Lines executed:93.75% of 16

Branches executed:100.00% of 12

Taken at least once:75.00% of 12

Calls executed:50.00% of 2

Function 'playSmithy'

Lines executed:83.33% of 6

Branches executed:100.00% of 2

Taken at least once:50.00% of 2

Calls executed:50.00% of 2

Debugging

GDB is a great tool because it allows a live view at your program that you can slowly step through to keep track of all the various states. In order to debug Smithy I ran the command “gdb cardtest1” and then “break playSmithy”. This means once the playSmithy function is reached, GDB allowed me to step through it. I did this utilizing the step command. I was able to use print to print out the hand count before each step so I could visually see what was being changed and what wasn’t. This led me to find the hand count was never updated at all. I found in GDB the for loop where cards are gained was skipped entirely. Thus I found the bug present in playSmithy(). I fixed the code to read:

```
int i = 0;

   for(i; i < 3; i++) {
     drawCard(currentPlayer, state);
  }
```

I used a similar tactic to step through callVIllage. I used the print function to print actions before each step. I found after I stepped through line 698 the actions had doubled. I fixed the Village code on line 698 by:

```
  state->numActions = state->numActions + 2;
```

This ensures 2 actions are added, not multiplied by the current value.