Re-factoring

Using my team's tauqirs' dominion code, I didn't have to change the test itself to run. However, there are many calls that will break if the array structure or data structure change.

For instance, to set up the test for mine card, I have to manually set the hand cards and deck cards in order to run the tests. If tarqirs change the struct or how the array in the hand or deck are set up, the test would not be able to run. This is also the case when I'm assert and check if the test pass or fail.

The best way is to create some interface calls to change the hand card or deck card through some function calls so that the test program does not need to know about the data structure used. It's better to use something like the cardEffect wrapper call which doesn't need to change.

Bug-Reports

`Smithy`: The first bug filed is smithy. The code failed the test to see whether the player has drawn 3 new cards to his hand. For example, with a full hand, after drawing 3 cards and discard the smithy card, the player should end up with 7 cards, instead, it shows 6 cards only. The program did not crash or return with errors so it could be some sort index error and return prematurely. After checking out code, it appears that the for loop condition should be 3 and not 2.

```
int playSmithy(struct gameState* state, int currentPlayer, int handPos){
    //+3 Cards
    int i=0;
      for (i = 0; i < 2; i++)
    {
      drawCard(currentPlayer, state);
    }

      //discard card from hand
      discardCard(handPos, currentPlayer, state, 0);
      return 0;
}
```

```
Noverse Bug Reporting Template
==============================

Title:    smithy card only draws 2 cards instead of 3.

Class: Serious bug
e.g. "Feature Request", "System Error", "Serious Bug"

Date:           25 May 2017
Reported By:   linsh
Email:      linsh@oregonstate.edu


Product: Dominion                       Version: OSU
Platform: All                       Version: n/a
Browser: n/a                        Version:n/a
URL:       n/a
```

[Bug-Reports] continued on next page...

```
Is it reproducible: Yes / Occasionally / One Time / No
Yes.


Description
===========
In my testing, I played the smithy to draw 3 cards but only 2 new cards shows up
in hand and the deck count is incorrect as well.



Steps to Produce/Reproduce
--------------------------

These are the steps for setting up the test:

    /* set hand */
    handpos = 0;
    G.hand[thisPlayer][handpos] = smithy;
    newCards = 3;
    discarded = 1;

    // copy the game state to a test case
    memcpy(&testG, &G, sizeof(struct gameState));
    cardEffect(smithy, choice1, choice2, choice3, &testG, handpos, &bonus);

When compare the new state against the expected state. The test shows 1 card
missing.

Expected Results
----------------
hand count = 6, expected = 7... Error
deck count = 3, expected = 2... Error




Actual Results
--------------
hand count = 6, expected = 7... Error
deck count = 3, expected = 2... Error



Workarounds
-----------
None



Attachments
-----------
```

```
unittestresults.out



Other Information
-----------------
```

mine: This was an interesting bug because it passes some test but failed at others. It appears that when you trade copper or silver for higher value card, it is ok. But whenever it is trade for something of equal value or less, it fails. At first, it looks like maybe that's what the card should do but the official rule does not block trading for equal or less value card. This could be due to some bug when comparing the two card values. In the source code, it appears that the second cannot be less than the first card or it will fail.

```
    if ( (getCost(state->hand[currentPlayer][choice1]) + 3) > getCost(choice2) )
    {
      return -1;
    }
```

```
Noverse Bug Reporting Template
==============================

Title:    mine card cannot trade copper for copper

Class: Annoyance
e.g. "Feature Request", "System Error", "Serious Bug"

Date:           25 May 2017
Reported By:   linsh
Email:      linsh@oregonstate.edu


Product: Dominion                       Version: tauqirsDominion
Platform: All                       Version: n/a
Browser: n/a                        Version:n/a
URL:       n/a

Is it reproducible: Yes / Occasionally / One Time / No
Yes.


Description
===========
I set up the unit test to trade copper, silver, and gold. The
card failed when trading copper for copper. It works if it's copper to silver
and silver to gold. This is a minor annoyance since most poeple use it to get
higher value treasure card anyway.


Steps to Produce/Reproduce
--------------------------
We set the hand so that we can play the mine card with the choice of treasure
card in hand position 0 and 1.

        /* set hand  and deck*/
```

```
        handpos = 0;
        G.hand[thisPlayer][handpos] = mine;
        choice1 = 1;
        G.hand[thisPlayer][choice1] = treasure_cards[i];
        choice2 = treasure_cards[j];

        newCards = 1;
        trashed = 1;
        played = 1;
        xtraCoins = 0;

        // copy the game state to a test case
        memcpy(&testG, &G, sizeof(struct gameState));
        rv = cardEffect(mine, choice1, choice2, choice3, &testG, handpos, &bonus);

When we compare the result, trading for treasure card of the same or less value
will return with error.

Expected Results/  Actual Results
----------------
*** trade treasure card 4 for 4
return status = -1, expected = 0... Error
hand count = 5, expected = 4... Error
new card = 11, expected = 4 ... Error
supply count = 46, expected = 45... Error
mine discarded: card = 11, NOT = 11... Error

*** trade treasure card 4 for 5
return status = 0, expected = 0... OK
hand count = 4, expected = 4... OK
new card = 5, expected = 5 ... OK
supply count = 39, expected = 39... OK
mine discarded: card = 5, NOT = 11... OK

** trade treasure card 4 for 6
return status = 0, expected = 0... OK
hand count = 4, expected = 4... OK
new card = 6, expected = 6 ... OK
supply count = 29, expected = 29... OK
mine discarded: card = 6, NOT = 11... OK

Workarounds
-----------
only trade for higher value treasure cards


Attachments
-----------
unittestresults.out



Other Information
```

```
90 ------------------
```

Debugging

`smithy`: In my own dominion code, I'm debuggint the smithy call as well. The test for this one is simple but the error is different from my teammate. This bug appears not to draw no new card at all, Instead of expected hand count of 7, i have only 4 cards in hand.

```
----------------- Testing Card: smithy ----------------
draw 3 cards
hand count = 4, expected = 7... Error
deck count = 5, expected = 2... Error
5 smithy played: card = 4, NOT = 13... OK
Failed 2 tests
File 'cardtest1.c'
Lines executed:97.62% of 42
Creating 'cardtest1.c.gcov'
```

In order to debug this, I have created several breakpoints and watchpoints. By doing that I can see if how the hand cards change. Other breakpoints include cardEffect and play_smithy, watch for draw card loop index.

```
set logging on
break /home/chewie/Documents/cs362/CS362-004-SP17/projects/linsh/dominion/cardtest1.c
    :60
  commands
    watch testG.handCount[thisPlayer]
5   watch testG.hand[thisPlayer]
  end
break cardEffect
break play_smithy
```

In the case, I can see that the smithy was played and discarded when play_smithy begins by playing the smithy card at handPos=0. The handCount watch point was reflect the new change. But it didn't show any change to the hand from drawing new card.

After rerun the debugger, I have also set watchpoints for index i value. If the loop stops this would help show me why. At first the i integer set to 0 as expected but it did not step into the loop, instead, it proceeds to discarding the smithy card. Looking at line 697 of the code, it becomes clear that the loop condition is wrong and it exits too soon.

```
Breakpoint 3, play_smithy (state=0x7fffffff0ff0, handPos=0) at dominion.c:694
694       int currentPlayer = whoseTurn(state);
(gdb) c
Continuing.
5
Watchpoint 7: testG.hand[thisPlayer]

Old value = {13, 4, 1, 4, 4, 0 <repeats 495 times>}
New value = {-1, 4, 1, 4, 4, 0 <repeats 495 times>}
10 discardCard (handPos=0, currentPlayer=0, state=0x7fffffff0ff0, trashFlag=0)
    at dominion.c:1380
```

```
1380    if ( handPos == (state->handCount[currentPlayer] - 1) )   //last card in hand
    array is played
(gdb) c
Continuing.

Watchpoint 7: testG.hand[thisPlayer]

Old value = {-1, 4, 1, 4, 4, 0 <repeats 495 times>}
New value = {4, 4, 1, 4, 4, 0 <repeats 495 times>}
discardCard (handPos=0, currentPlayer=0, state=0x7fffffff0ff0, trashFlag=0)
    at dominion.c:1395
1395        state->hand[currentPlayer][state->handCount[currentPlayer] - 1] = -1;
(gdb) c
Continuing.

Watchpoint 7: testG.hand[thisPlayer]

Old value = {4, 4, 1, 4, 4, 0 <repeats 495 times>}
New value = {4, 4, 1, 4, -1, 0 <repeats 495 times>}
discardCard (handPos=0, currentPlayer=0, state=0x7fffffff0ff0, trashFlag=0)
    at dominion.c:1397
1397        state->handCount[currentPlayer]--;
(gdb) c
Continuing.

Hardware watchpoint 6: testG.handCount[thisPlayer]

Old value = 5
New value = 4
0x0000000000406383 in discardCard (handPos=0, currentPlayer=0, state=0x7fffffff0ff0,
    trashFlag=0) at dominion.c:1397
1397        state->handCount[currentPlayer]--;


Breakpoint 3, play_smithy (state=0x7fffffff0ff0, handPos=0) at dominion.c:694
694       int currentPlayer = whoseTurn(state);
689
690
691   int play_smithy(struct gameState* state, int handPos)
692   {
693       int i;
694       int currentPlayer = whoseTurn(state);
695
696       //+3 Cards
697       for (i = 0; i > 3; i++) { /* FIXME new bug */
698           drawCard(currentPlayer, state);
Hardware watchpoint 13: i
whoseTurn (state=0x7fffffff0ff0) at dominion.c:347
347     return state->whoseTurn;
348   }
play_smithy (state=0x7fffffff0ff0, handPos=0) at dominion.c:697
697       for (i = 0; i > 3; i++) { /* FIXME new bug */
```

```
Hardware watchpoint 13: i

Old value = 32767
New value = 0
0x000000000040399b in play_smithy (state=0x7fffffff0ff0, handPos=0) at dominion.c:697
697      for (i = 0; i > 3; i++) { /* FIXME new bug */
701      discardCard(handPos, currentPlayer, state, 0);
Continuing.
```

After fixing the loop condition, rerunning the test shows that it has passed the tests. With the debugger, I can see that it has stepped through the loop this time, adding a new card to hand each time.

```
Breakpoint 3, play_smithy (state=0x7fffffff0ff0, handPos=0) at dominion.c:694
694      int currentPlayer = whoseTurn(state);
Hardware watchpoint 6: i
Continuing.

Hardware watchpoint 6: i

Old value = 32767
New value = 0
0x000000000040399b in play_smithy (state=0x7fffffff0ff0, handPos=0) at dominion.c:697
697      for (i = 0; i < 3; i++) { /* FIXME new bug */
Continuing.

Watchpoint 5: testG.hand[thisPlayer]

Old value = {13, 4, 1, 4, 4, 0 <repeats 495 times>}
New value = {13, 4, 1, 4, 4, 4, 0 <repeats 494 times>}
drawCard (player=0, state=0x7fffffff0ff0) at dominion.c:576
576      state->deckCount[player]--;
Continuing.

Hardware watchpoint 4: testG.handCount[thisPlayer]

Old value = 5
New value = 6
0x00000000004033c2 in drawCard (player=0, state=0x7fffffff0ff0) at dominion.c:577
577      state->handCount[player]++;//Increment hand count
Continuing.

Hardware watchpoint 6: i

Old value = 0
New value = 1
0x00000000004039c4 in play_smithy (state=0x7fffffff0ff0, handPos=0) at dominion.c:697
697      for (i = 0; i < 3; i++) { /* FIXME new bug */
Continuing.

Watchpoint 5: testG.hand[thisPlayer]

Old value = {13, 4, 1, 4, 4, 4, 0 <repeats 494 times>}
New value = {13, 4, 1, 4, 4, 4, 4, 0 <repeats 493 times>}
```

```
drawCard (player=0, state=0x7fffffff0ff0) at dominion.c:576
576       state->deckCount[player]--;
Continuing.

Hardware watchpoint 4: testG.handCount[thisPlayer]

Old value = 6
New value = 7
0x00000000004033c2 in drawCard (player=0, state=0x7fffffff0ff0) at dominion.c:577
577       state->handCount[player]++;//Increment hand count
Continuing.

Hardware watchpoint 6: i

Old value = 1
New value = 2
0x00000000004039c4 in play_smithy (state=0x7fffffff0ff0, handPos=0) at dominion.c:697
697       for (i = 0; i < 3; i++) { /* FIXME new bug */
Continuing.

Watchpoint 5: testG.hand[thisPlayer]

Old value = {13, 4, 1, 4, 4, 4, 4, 0 <repeats 493 times>}
New value = {13, 4, 1, 4, 4, 4, 4, 4, 0 <repeats 492 times>}
drawCard (player=0, state=0x7fffffff0ff0) at dominion.c:576
576       state->deckCount[player]--;
Continuing.

Hardware watchpoint 4: testG.handCount[thisPlayer]

Old value = 7
New value = 8
0x00000000004033c2 in drawCard (player=0, state=0x7fffffff0ff0) at dominion.c:577
577       state->handCount[player]++;//Increment hand count
Continuing.

Hardware watchpoint 6: i

Old value = 2
New value = 3
0x00000000004039c4 in play_smithy (state=0x7fffffff0ff0, handPos=0) at dominion.c:697
697       for (i = 0; i < 3; i++) { /* FIXME new bug */
Continuing.

Watchpoint 5: testG.hand[thisPlayer]

Old value = {13, 4, 1, 4, 4, 4, 4, 4, 0 <repeats 492 times>}
New value = {-1, 4, 1, 4, 4, 4, 4, 4, 0 <repeats 492 times>}
discardCard (handPos=0, currentPlayer=0, state=0x7fffffff0ff0, trashFlag=0)
    at dominion.c:1380
1380    if ( handPos == (state->handCount[currentPlayer] - 1) )   //last card in hand
    array is played
Continuing.
```

[Debugging] continued on next page...

```
Watchpoint 5: testG.hand[thisPlayer]

Old value = {-1, 4, 1, 4, 4, 4, 4, 4, 0 <repeats 492 times>}
New value = {4, 4, 1, 4, 4, 4, 4, 4, 0 <repeats 492 times>}
discardCard (handPos=0, currentPlayer=0, state=0x7fffffff0ff0, trashFlag=0)
    at dominion.c:1395
1395        state->hand[currentPlayer][state->handCount[currentPlayer] - 1] = -1;
Continuing.

Watchpoint 5: testG.hand[thisPlayer]

Old value = {4, 4, 1, 4, 4, 4, 4, 4, 0 <repeats 492 times>}
New value = {4, 4, 1, 4, 4, 4, 4, -1, 0 <repeats 492 times>}
discardCard (handPos=0, currentPlayer=0, state=0x7fffffff0ff0, trashFlag=0)
    at dominion.c:1397
1397        state->handCount[currentPlayer]--;
Continuing.

Hardware watchpoint 4: testG.handCount[thisPlayer]

Old value = 8
New value = 7
0x0000000000406383 in discardCard (handPos=0, currentPlayer=0, state=0x7fffffff0ff0,
    trashFlag=0) at dominion.c:1397
1397        state->handCount[currentPlayer]--;
Continuing.
```