Dominion-Base Architecture

Team Number: 13
Team Members: Jimmy Brewer, Crystal Chan

What are the files/components? What are their roles?

The Dominion-Base directory has 18 files, which includes 12 C source files, 4 C header files, 1 makefile, and 1 .gcno file. All file names and their role are listed below.

Source File

1. dominion.c
- It defines all the functions to build a dominion game object and its contents (e.g., deck of cards, card effect, score, action, etc.).

2. interface.c
- It is the user interface file. It defines several functions to print player, card, deck, action, and score in console.

3. playdom.c
- It has a main() function and defines how two players play the dominion game with the Smithy and Adventure action card.

4. player.c
- It has 2 main() functions. The main() function executes the dominion game and gives instructions players during the game. The main2() function is used to initialize a game.

5. rngs.c
- This is an ASCII C library for multi-stream random number generation. It contains several functions to create and generate a random number and set its state,

6. rt.c
- It has a main() function. It tests the state of the random number generator from rngs.c

7. testBuyerCard.c
- It defines a checkDrawCard function. It performs random tests on the buyCard function with the checkDrawCard function.

8. testDrawCard.c
- It defines a checkDrawCard function. It performs random tests on the drawCard function with the checkDrawCard function.

9. badTestDrawCard.c

- It defines a checkDrawCard function. It performs random tests on the drawCard function with the checkDrawCard function. It seems to be the oldest testDawCard file and has been replaced by testDrawCard.c.

10. betterTestDrawCard.c
- It defines a checkDrawCard function. It performs random tests on the drawCard function with the checkDrawCard function. It seems to be an improved version of the oldest testDawCard file and has been replaced by testDrawCard.c.

11. testInit.c
- It has a main() function. It tests the initializeGame function

12. testShuffle.c
- It has a main() function. It tests the shuffle() function.

Header File

1. dominion_helper.h
- It is one of two header files of dominion.c. It imports the dominion.h header file and contains the following function prototypes: drawCard(), updateCoins(), discardCard(), gainCard(), getCost(), and cardEffect().

2. dominion.h
- It is one of two header files of dominion.c. It contains most of the function prototypes except for those in dominion_helper.h.

3. interface.h
- It is the header file of interface.c. It imports dominion.h and contains several macros and all the function prototypes from interface.c.

4. rngs.h
- It is the header file of rngs.c. It contains all the function prototypes from rngs.c.

Makefie

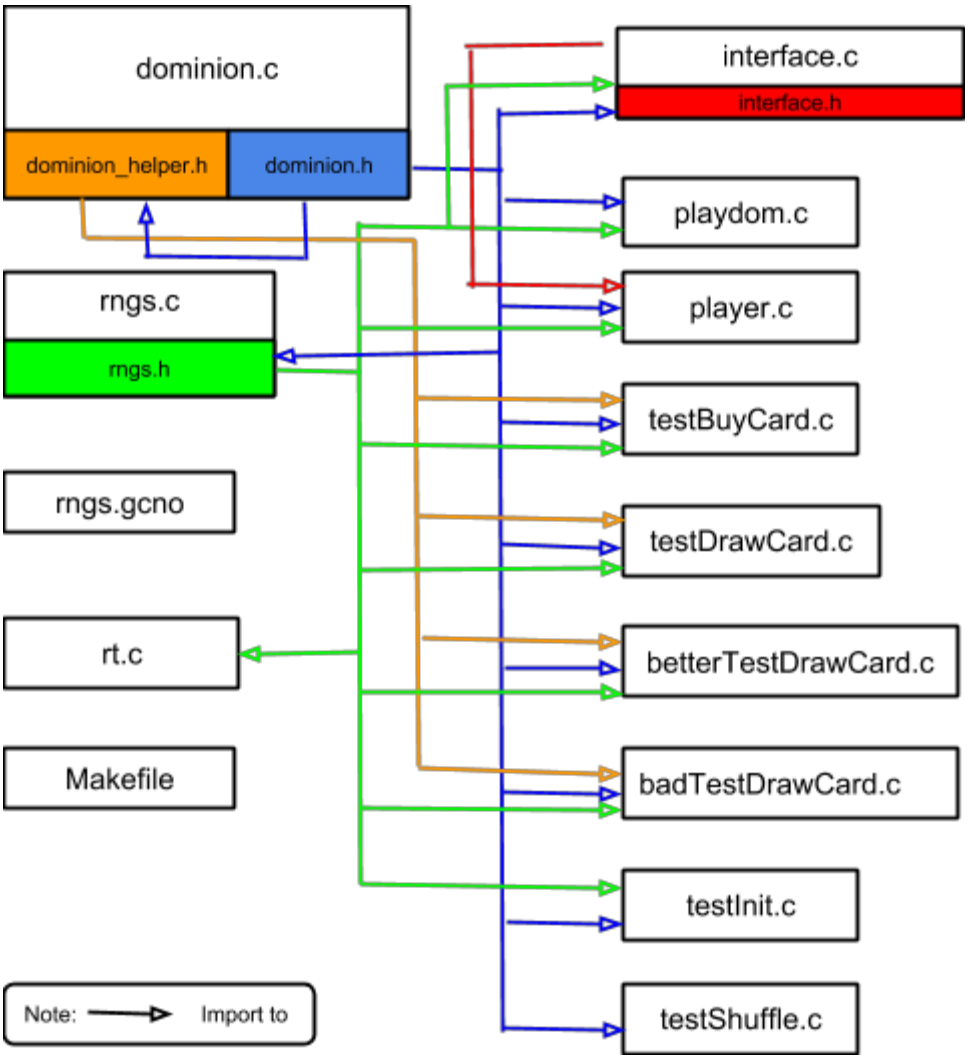
1. Makefile
- It tells the compiler which source files to compile into object files. The compiler compiles each one individually, and give the linker the list of object files to combine into the executable.

***.gcno**

1. rngs.gcno
- It was generated when rngs.c was compiled with the GCC –ftest-coverage option. It contains information to reconstruct the basic block graphs and assign source line numbers to blocks.

How do they interact?

-Show how files interact in a diagram



Function chains

See appendix for function details.

Game start functions

- main()
    - intializeGame()
        - SelectStream()
        - PutSeed()
        - shuffle()

                - qsort()
                - Random()
                - floor()
            - drawCard()
                - shuffle()
                    - qsort()
                    - Random()
                    - floor()
            - updateCoins()

Play a card functions

- playCard()
    - handCard()
        - whosTurn()
    - cardEffect()
        - shuffle()
            - qsort()
            - Random()
            - floor()
        - drawCard()
            - shuffle()
                - qsort()
                - Random()
                - floor()
        - discardCard()
        - updateCoins()
        - supplyCount()
        - getCost()
        - gainCard()
            - supplyCount()
        - isGameOver()
        - handCard()
            - whoseTurn()
    - updateCoins()

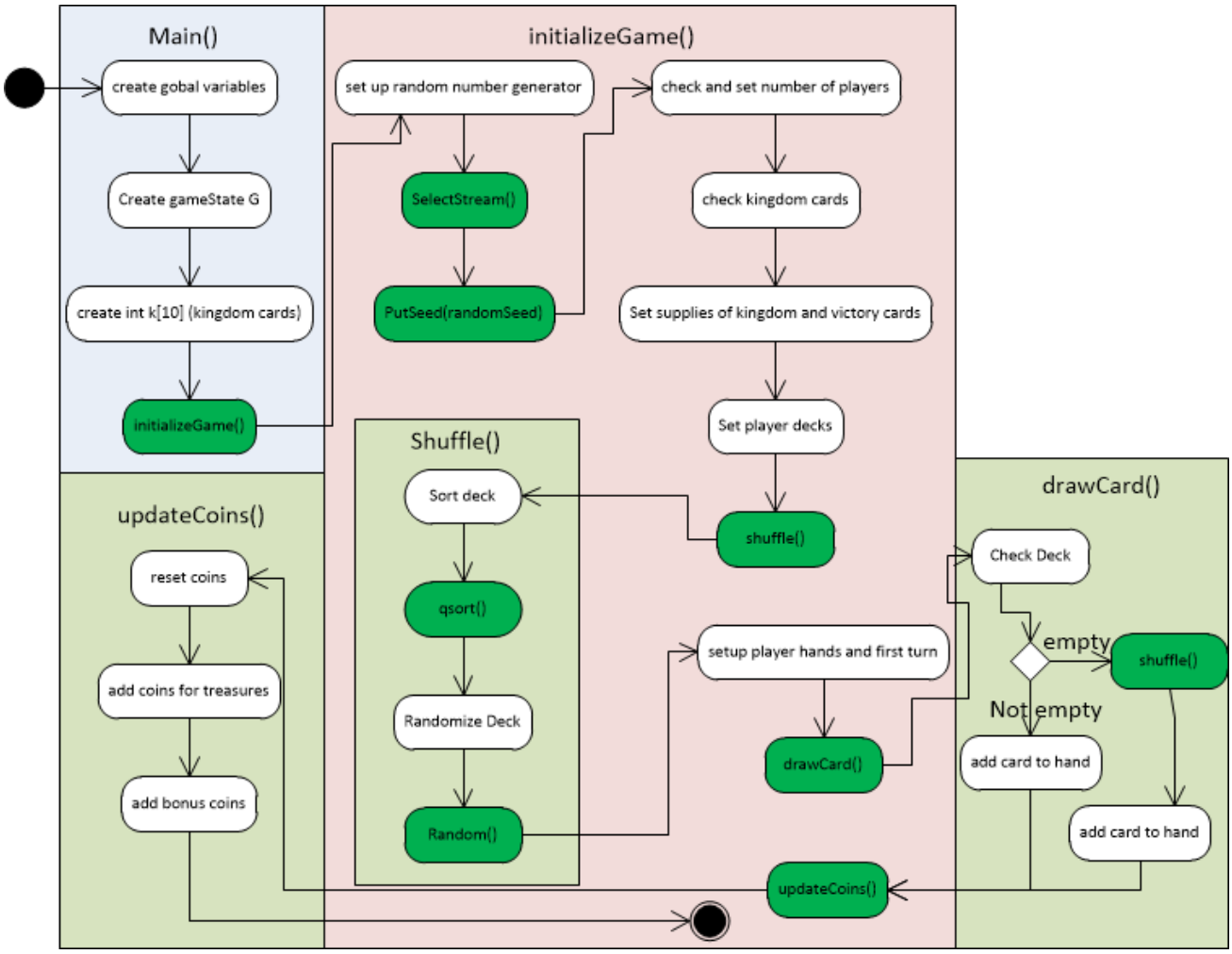
End turn functions

- endTurn()
    - whoseTurn()
    - drawCard()
        - shuffle()
            - qsort()
            - Random()
            - floor()

- updateCoins()

UML Activity Diagrams

Game Initialization

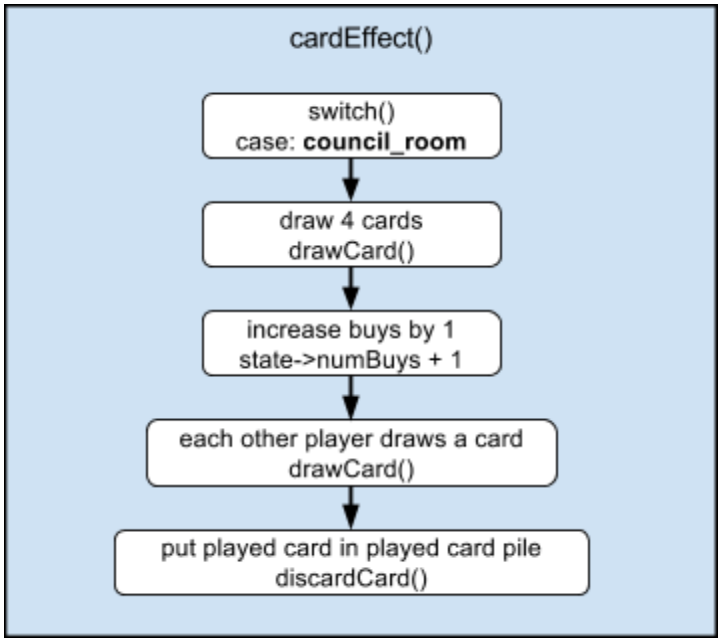


Player's Turn

A Few Diagrams for Playing Some Specific Cards

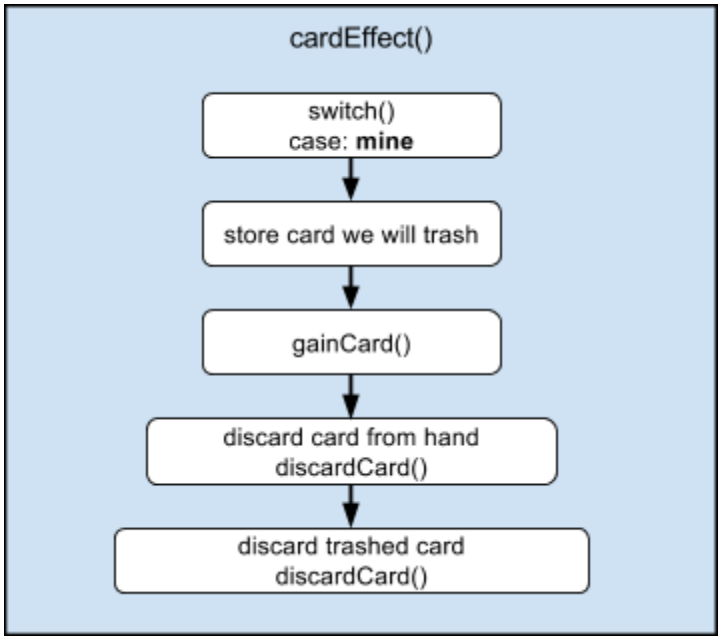
1. Play Council Room

Rule: draw 4 cards, increase buys by 1, and each other player draws a card

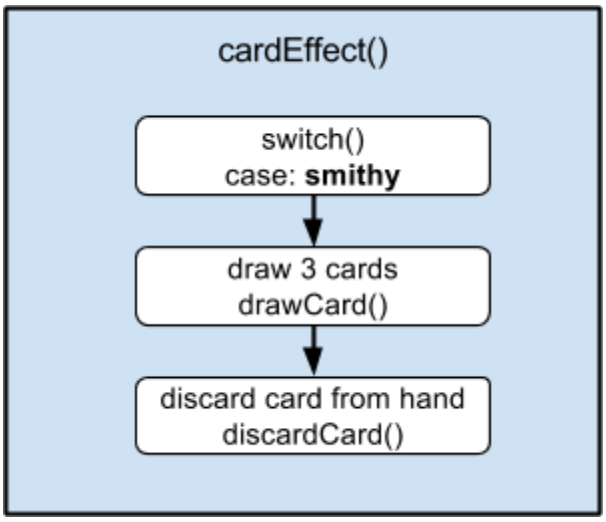


2. Play Mine

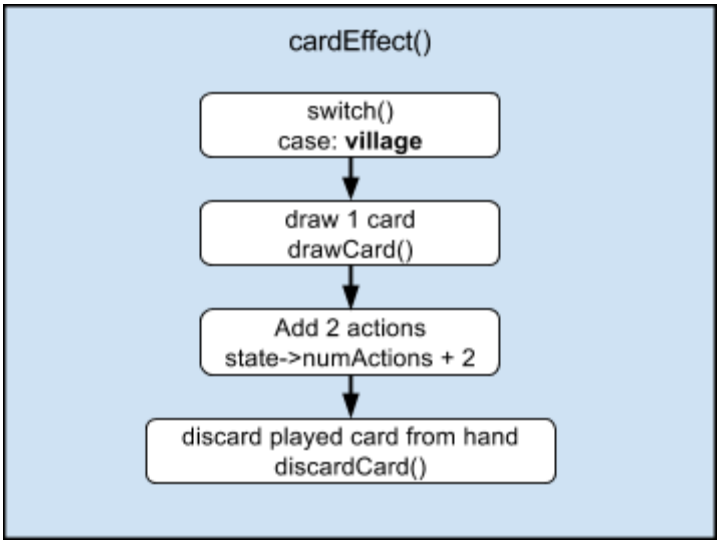
Rule: Trash a Treasure card from your hand, gain a treasure card and put it into your hand



3. Play Smithy
Rule: draw 3 cards

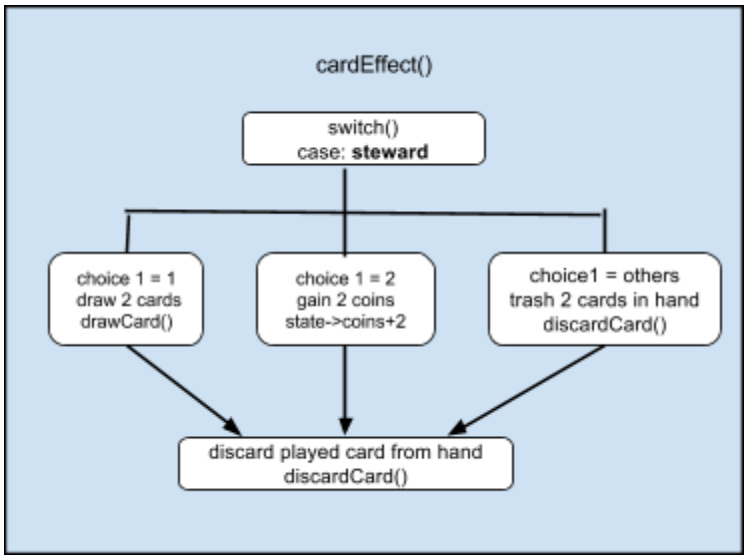


4. Play Village
Rule: draw 1 card and gain 2 actions



5. Play Steward

Rule: choose one from the following: draw 2 cards, gain 2 coins, or trash 2 cards from your hand



Appendix 1 - function descriptions

Main()

Description:

Main program loop. Accepts an input of a random seed used for initialization then runs through 2 simple AI routines until one player wins. The first AI plays the smithy card if able, and the second AI plays the adventure card if able. Both AIs play and buy treasure and provinces when able. Outputs the player actions and the final score.

Input:

Random Seed from the command line (argv[1])

Return:

Returns 0 to the system on success

Data structures:

- Used globally
    - Structure GameState G – holds all the player data and deck information for the

game
    - int array K[10] – kingdom cards available for purchase
- Used locally
    - int money
    - int smithyPos
    - int adventurePos
    - int i – used for index counter
    - int numSmithies
    - int numAdventurers

Dependencies:

- stdio
    - printf()
- stdlib
    - atoi()
- dominion.h
    - Enumerations
        - CARD
- dominion.c
    - intializeGame()
    - isGameOver()
    - numHandCards()
    - handCard()
    - whoseTurn()
    - playCard()
    - buyCard()
    - endTurn()
    - scoreFor()

initializeGame ()

Description:

Sets up struct gameState with the initial game information modifying the data in a struct gameState. Finishes with setting up the first player and drawing the first player's cards.

Input:

- int numPlayers – number of players
- int kingdomCards[10] – the 10 kingdom cards to be available for purchase
- int randomSeed – used to randomize the result for shuffling
- struct gameState state – holds a pointer to a gameState variable

Return:

Returns -1 on error

Returns 0 on success

Data structures:

- Used locally
    - int i
    - int j
    - int it

Dependencies:

- Dominion.h
    - Definitions
        - MAX_PLAYERS
    - Enumerations
        - CARD
- rngs
        - SelectStream()
    - PutSeed()
- local
    - shuffle()
    - drawCard()
    - updateCoins()

isGameOver()

Description:

Checks the number of provinces or if three supply piles are empty to determine if the game is over.

Input:

- struct gameState state – holds a pointer to a gameState variable

Return:

Returns 1 if the game is over
Returns 0 on if the game is not over

Data structures:

- Used locally
    - int i
    - int j

Dependencies:

- dominion.h
    - Enumerations

        - CARD

numHandCards()

Description:
Getter for gameState handCount.

Input:
- struct gameState state – holds a pointer to a gameState variable

Return:
Returns number of cards in the current player's hand

Data structures:
- none

Dependencies:
- local
    - whoseTurn()

whoseTurn()
Description:
Getter for gameState whoseTurn.

Input:
- struct gameState state – holds a pointer to a gameState variable

Return:
Returns the integer of whose turn it is now.

Data structures:
- none

Dependencies:
- none

handCard ()
Description:
Getter for gameState returning the card in a specific position of the hand.

Input:

- struct gameState state – holds a pointer to a gameState variable
- int handPos – the position of the hand of the card to return

Return:

Returns the integer of the card in the handPos of the current player.

Data structures:

- int currentPlayer

Dependencies:

- local
- whoseTurn()

supplyCount()

Description:

Getter for gameState returning the supply count of a specific card.

Input:

- struct gameState state – holds a pointer to a gameState variable
- int card

Return:

Returns the count of supply for a card

Data structures:

- none

Dependencies:

- none

shuffle()

Description:

Getter for gameState returning the card in a specific position of the hand.

Input:

- struct gameState state – holds a pointer to a gameState variable
- int player

Return:

Returns the integer of the card in the handPos of the current player.

Data structures:

- int newDeck[MAX_DECK]
- int newDeckPos
- int card
- int i

Dependencies:

- dominion.h
    - definition
        - MAX_DECK
- stdlib
    - qsort()
- rngs
    - Random()
- math
    - floor()

playCard ()

Description:

Validates the card being played can be played then calls cardEffect to play the card.

Input:

- struct gameState state – holds a pointer to a gameState variable
- int handPos – the position of the hand of the card to return
- int choice1
- int choice2
- int choice3

Return:

Returns -1 on error
Returns 0 on success

Data structures:

- int card
- int coin_bonus

Dependencies:

- dominion.h
    - enumeration
        - CARD

- local
    - handCard()
    - cardEffect()
    - updateCoins()

discardCard ()

Description:

Validates the card being played can be played then calls cardEffect to play the card.

Input:

- struct gameState state – holds a pointer to a gameState variable
- int handPos – the position of the hand of the card to return
- int currentPlayer
- int trashFlag

Return:

Returns 0 on success

Data structures:

- none

Dependencies:

- none

buyCard()

Description:

Validates the card being played can be played then calls cardEffect to play the card.

Input:

- struct gameState state – holds a pointer to a gameState variable
- int supplyPos – the position of the supply

Return:

Returns -1 on error
Returns 0 on success

Data structures:

- int who – current player

Dependencies:

- local
    - supplyCount()
    - getCost()
    - gainCard()
- stdio
    - printf()

endTurn()

Description:

Changes the current player to the next player or the first player if the last player has finished their turn. Resets the player gameState and draws cards for the next player.

Input:
struct gameState state – holds a pointer to a gameState variable

Return:

Returns 0 on success

Data structures:

- int k
- int i
- int currentPlayer – current player from gameState

Dependencies:

- local
    - whoseTurn()
    - drawCard()
    - updateCoins()

scoreFor()

Description:

Calculates a specific player's current score.

Input:

- struct gameState state – holds a pointer to a gameState variable
- int player

Return:

Returns score

Data structures:
- int i
- int score

Dependencies:
- dominion.h
    - enumeration
        - CARD
- local
    - fullDeckCount()

fullDeckCount()

Description:
Returns the count of a specific card in a specific player's discard deck and hand.

Input:
- struct gameState state – holds a pointer to a gameState variable
- int card
- int player

Return:
Returns count

Data structures:
- int i
- int count

Dependencies:
- none

drawCard()

Description:
Adds a card from the deck to the player's hand. If the deck is empty the deck is shuffled, and if still empty returns -1.

Input:
- struct gameState state – holds a pointer to a gameState variable
- int player

Return:

Returns -1 if the deck is empty after shuffling
Returns 0 on success

Data structures:

- int count
- int deckCounter

Dependencies:

- local
    - shuffle()
- stdio
    - printf()

updateCoins()

Description:

Returns the count of a specific card in a specific player's discard deck and hand.

Input:

- struct gameState state – holds a pointer to a gameState variable
- int player
- int bonus

Return:

Returns 0 on success

Data structures:

- int i

Dependencies:

- dominon.h
    - enumeration
        - CARD

getCost()

Description:

Getter for the cost of cards (hard coded into this function).

Input:

- int cardNumber

Return:

Returns the cost of the card on success
Returns -1 on failure

Data structures:

- none

Dependencies:

- dominion.h
    - enumeration
        - CARD

gainCard()

Description:

Adds a specific card to a players hand, deck, or trash. Checks for enough supply of the card then puts the card in the directed location.

Input:

- struct gameState state – holds a pointer to a gameState variable
- int supplyPos – enumerated card
- int toFlag – where to put the card
- int player –who to give it to

Return:

Returns -1 if not enough supply
Returns 0 on success

Data structures:

- none

Dependencies:

- local
    - supplyCount()

cardEffect()

Description:

Executes a card based on the provided card and choices.

Input:

- struct gameState state – holds a pointer to a gameState variable
- int handPos

- int bonus – pointer to a int
- int card
- int choice1
- int choice2
- int choice3

Return:

Returns -1 if the card failed to play
Returns 0 on success

Data structures:

- int i
- int j
- int k
- int index
- int x
- int currentPlayer
- int nextPlayer – set to currentPlayer +1
- int tributeRevealedCards[2] – set to -1, -1
- int temphand[MAX_HAND]
- int drawntreasure
- int cardDrawn
- int z

Dependencies:

- dominion.h
    - enumeration
    - CARD
- local
    - shuffle()
    - drawCard()
    - discardCard()
    - updateCoins()
    - supplyCount()
    - getCost()
    - gainCard()
    - isGameOver()
    - handCard()
- stdio
    - printf()

Appendix 2 - functions in each source file

- dominion.c
  - int compare(const void* a, const void* b);
  - struct gameState* newGame()
  - int* kingdomCards(int k1, int k2, int k3, int k4, int k5, int k6, int k7,int k8, int k9, int k10)
  - int initializeGame(int numPlayers, int kingdomCards[10], int randomSeed, struct gameState *state)
  - int shuffle(int player, struct gameState *state)
  - int playCard(int handPos, int choice1, int choice2, int choice3, struct gameState *state)
  - int buyCard(int supplyPos, struct gameState *state)
  - int numHandCards(struct gameState *state)
  - int handCard(int handPos, struct gameState *state)
  - int supplyCount(int card, struct gameState *state)
  - int fullDeckCount(int player, int card, struct gameState*state)
  - int whoseTurn(struct gameState *state)
  - int endTurn(struct gameState *state)
  - int isGameOver(struct gameState *state)
  - int scoreFor (int player, struct gameState *state)
  - int getWinners(int players[MAX_PLAYERS], struct gameState*state)
  - int drawCard(int player, struct gameState *state)
  - int getCost(int cardNumber)
  - int cardEffect(int card, int choice1, int choice2, int choice3, struct gameState *state, int handPos, int *bonus)
  - int discardCard(int handPos, int currentPlayer, struct gameState *state, int trashFlag)
  - int gainCard(int supplyPos, struct gameState *state, int toFlag, int player)
  - int updateCoins(int player, struct gameState *state, int bonus)

- interface.c
  - void cardNumToName(int card, char *name)
  - int getCardCost(int card)
  - void printHand(int player, struct gameState *game)
  - void printDeck(int player, struct gameState *game)
  - void printPlayed(int player, struct gameState *game)
  - void printDiscard(int player, struct gameState *game)
  - void printSupply(struct gameState *game)
  - void printState(struct gameState *game)
  - void printScores(struct gameState *game)
  - void printHelp(void)
  - void phaseNumToName(int phase, char *name)
  - int addCardToHand(int player, int card, struct gameState *game)

    - void selectKingdomCards(int randomSeed, int kingCards[NUM_K_CARDS])
    - int countHandCoins(int player, struct gameState *game)
    - void executeBotTurn(int player, int *turnNum, struct gameState *game)

- rngs.c
    - double Random(void)
    - void PlantSeeds(long x)
    - void PutSeed(long x)
    - void GetSeed(long *x)
    - void SelectStream(int index)
    - void TestRandom(void)

- playdom.c
    - int main (int argc, char** argv)

- player.c
    - int main (int argc, char** argv)

- rt.c
    - int main2(int argc, char *argv[])
    - int main(int argc, char** argv)

- testBuyCard.c
    - int checkDrawCard(int p, struct gameState *post)
    - int main ()

- testDrawCard.c
    - int checkDrawCard(int p, struct gameState *post)
    - int main ()

- badTestDrawCard.c
    - int checkDrawCard(int p, struct gameState *post)
    - int main ()

- betterTestDrawCard.c
    - int checkDrawCard(int p, struct gameState *post)
    - int main ()

- testInit.c
    - int main (int argc, char** argv)

- testShuffle.c
    - int compare(const int* a, const int* b);

- int main ()