The following sections will lead you through the different elements of the game that we have implemented. All the load files will be included in the same folder. Thank you for looking at our demo!

Saving the Game:

Unless the current player owes money to the bank or another player, the game can be saved at any point during their turn. All they would need to do is enter the “save [filename]” command.

You can start a new game by running “. /watopoly” and playing a few turns of the game. You do not need to play those turns, but it may help show that the game can save at different stages. Once you stop playing, you can call “save [filename]” and the game will save the details of the players and ownable properties. It will save players starting from the current player so that the current player starts first when the game is loaded in. Once you call the save command, the game will exit, and you can see the save file in the same folder as executable watopoly.

Note: When the save command is called the player’s turn will end, so when the game is loaded back in, it will be the next player’s turn.

The code for this is found in board.cc in the overloaded operator<<.

Loading the Game:

Loading the game is done using the command line options before running the game. You start by running “. /watopoly -load [filename]”, assuming that the file you are using is in the correct format, the game will be loaded at the point of save from that load file.

Start by loading the “loadtest1.txt” using the “. /watopoly -load [filename]”, or alternatively play some turns of a new game and save, then load it back up. “loadtest1.txt” will load you into the game where 2 players have begun playing but have not finished. Once the game is loaded, you can call the “all” command to display the assets of all the players to see that the saved game has been loaded properly.

Note: when loading the game, it will start with the player that was after the person who saved the game, as saving the game will end the person’s turn.

If a filename is not entered in the command line arguments, then the -load argument will be ignored, and a new game will start.

The code for this is found in board.cc in the overloaded operator>>.

STARTING THE GAME:

If starting a game from scratch, players are asked to enter their name and character one by one, and a full version of the board is printed. To try this out, run the game without the load option and enter the necessary information. At the start of each turn, a smaller, scoped version of the board is displayed as a strip, and the player's balance is displayed. Using the 'help' command, a list of possible commands is displayed.

ROLLING:

Using the 'roll' command will cause the player to move forward on the board by the amount rolled. If doubles are rolled, the player automatically rolls again. If they roll doubles three times in a row, they are sent to Tim's.

Property: Upgradable, Residence, Gym:

The propertydemo.txt game will be used to demonstrate the possible player actions on different properties. Make sure to use the testing option. If the player lands on an owned property, they pay the appropriate amount to the property owner. In this example, Christina owns the Arts2 monopoly, as well as V1, UWP, MKV, CIF, and MC, with 2 improvements on ECH. Moving Isabel 3 spaces (roll 1 2) to ECH makes her pay Christina $90, while moving Paul to 5 spaces (roll 2 3) to PAS only makes him pay $12. Similarly, moving Wayne to UWP (roll 2 1) makes him pay $100, moving Colin to CIF (roll 3 2) makes him pay 4x the sum of a dice roll, and moving Beatrix to MC (roll 2 3) makes her pay $35. If the property is owned by the player, nothing happens (move Christina to HH (roll 2 1)). If the property is unowned and the player has enough money to buy it, they will be given the option to purchase it (move Isabel 5 to RCH (roll 3 2)). If they choose not to, an auction occurs for the property. If the property is unowned and the player does not have enough money to purchase it (move Paul to DWE (roll 3 2)), an auction also occurs.

NonProperty:

To demonstrate what happens when a player lands on a non-property square, load the game nonpropertydemo.txt with the testing option. If a player lands on TUITION, they have the option of paying $300 of a tenth of their total worth to the bank (move Isabel 3 spaces (roll 1 2)). If a player lands of COOP FEE, they pay $150 to the bank (move Paul 4 spaces (roll 1 3)). If a player lands of GOOSE NESTING, nothing happens (move Wayne 4 spaces (roll 1 3)). Similarly, if a player lands on DC TIMS LINE, nothing happens (move Colin 3 spaces (roll 2 1)). However, if a player lands on GO TO TIMS, they are stuck in the Tim's line until they get out. To explore this, load the game timslinedemo.txt with the testing option. Kelly and Xiaoting have just gotten sent to the line, while this is Caitlin's third turn. Kelly has one cup, and Xiaoting has no money. Before their third turn in Tim's they have the option of paying $50 to get out, trying to roll doubles, or using a cup (if they have one). On their third turn, if they don't roll doubles, they are forced to pay $50. To test out SLC and NH, load slcdemo.txt and nhdemo.txt respectively, each with the testing mode. Moving any of the 6 players 3 squares (roll 1 2) will land them on SLC/NH. Whenever a player passes COLLECT OSAP, they receive $200. To see this in action, load osapdemo.txt and move Isabel 3 squares (roll 1 2) and Beatrix 4 squares (roll 1 3).

The code for these can be found in the playerEffect function in their respective class .cc files.

Display:

During one's turn, a player can enter "display" to print out the full gameboard.

Code can be found in board.cc: void displayBoard().

Bankruptcy:

The "bankrupt" command can be declared when the player owes more money than they currently possess to either a play or the bank. The player can call the "bankrupt" command to drop out of the game or try to raise enough money by selling improvements and selling/trading properties for money. When a player owes money, they can only sell improvements, trade, mortgage and declare bankruptcy. Once they have gathered enough money, the game will automatically pay the money owed and the player will be able to continue.

Start by loading the file "bankrupttest1.txt" into the game. Then roll until player vishal must pay rent to either david or the Bank. Now you can use trade or mortgage (you can't sell improvements because in this load file, there are no improvement for vishal to sell) to raise money to pay the rent, once you have enough money, the rent will be paid right away. Start by trading david any property vishal owns for less than the amount owed for rent. You will see that once the trade action is completed, the game will remind vishal that he still has rent to pay. Now either trade with david or mortgage a property so vishal has enough money, you will see that the game automatically pays the rent for vishal and shows his remaining balance.

Next, you can start again by loading "bankrupttest1.txt" again. Keep rolling for each player until vishal must pay rent to david or the Bank. This time rather than trying to raise money, declare bankruptcy for vishal by calling "bankrupt", and selecting option (a). Then all assets of vishal will be transferred to david; he will have a chance to choose if he wants to unmortgage properties that he is receiving if they are mortgaged. You can check if the assets have been transferred by calling the "all" command on the next player's turn.

Next, we want to demo going bankrupt to a bank. First, you want to start the game using the following arguments: “./watopoly -load bankrupttest2.txt -testing”. Once loaded in, use the “roll 1 1” command for david, then go to the next turn using the “next” command. This turn will be vishal’s turn, please check to confirm. On vishal’s turn roll 12 12, to get to the Coop Fee space on the board. At this space you will owe the bank money; you want to call the “bankrupt” command and select bankrupt rather than trying to raise money. All the assets from vishal will be given up to the bank, and the properties are auctioned off as unmortgaged.

Code found in board.cc in void startAuction(Property*).

Winning:

The winner is decided if there is 1 player remaining who has not declared bankruptcy. Load “winnertest1.txt” into the game using the following command “./watopoly -load winnertest1.txt”. Play david’s and vishal’s turns until vishal has to pay rent to either bank or david. Once vishal has to pay, declare bankruptcy by using the “bankrupt” command and select “declare bankruptcy” option. Once vishal is bankrupt he will be removed from the game, and since david is the only player remaining that has not declared bankruptcy, he will be the winner. If there were more players then the game would continue.

Assets:

Calling command all and assets, essentially do the same thing: they either print the assets of all players (all) or the assets of the player whose turn it is currently (assets).

You can start a new game or load in the following file: “assetstest1.txt”. Once the game has started, on whoever’s turn you can call “all” to see the assets of all players, and you can all “assets” to see the assets of the player whose turn it is.

Note: assets of any player cannot be accessed if a player needs to pay tuition.

Code for this is found in board.cc in getAllAssets() and printAssets(std::shared_ptr<Player> p).

Improve:

The improve command is used to either buy or sell improvements of any academic buildings. The general style of the improve commands follows “improve <property_name> buy/sell”.

For example, to buy an improvement for EV1, the player can enter “improve EV1 buy”. To sell an improvement on AL, the player can enter “improve AL sell”.

<u>-Exception-property finding</u>

If the players misspell the name of the academic buildings, or the input <property_name> is not a property that can be owned by players, the program will output "I cannot find this property." to standard output. The player can re-enter the command or enter other commands to continue the game.

<u>-Exception-buy improvements</u>

-If players try to improve a property that is not an academic building, they will get the error message:

"This building cannot be improved".

-If the player does not own the input property, he/she will get the error message: "This building is not yours :(".

-If the player mortgaged the input property, he/she will get the error message: "You have mortgaged this building.".

-If the player does not own an monopoly, he/she will get the error message: "You do not have enough improvements to sell".

-If the player already has 5 improvements on the property, he/she will get the error message: "You have reached the maximum improvement number.".

-If the player does not have enough money to improve the property, he/she will get the error message: "You don't have enough money.", and prevent him/her from purchasing the improvement.

<u>-Exception-sell improvements</u>

-If players try to sell improvement on a property that is not an academic building, they will get the error message:"This building cannot be improved".

-If the player does not own the input property, he/she will get the error message: "This building is not yours :(".

-If the player mortgaged the input property, he/she will get the error message: "You have mortgaged this building.".

-If the player already has no improvement on the property, he/she will get the error message: "You do not have enough improvements to sell".

<u>-Demo – improve</u>

Try loading the file “demo_improve.txt”. On joe’s turn, since joe already owns the Arts1 monopoly, if you enter “improve AL buy”, you will purchase an improvement for AL. However, if you enter “improve AL buy” again, you will get an error message since the improvement number is at maximum. To sell an improvement on AL, enter “improve AL sell”. If you are trying to sell improvements on a property that does not have any improvements, such as entering “improve ML sell”, an error message will be thrown. You cannot improve properties that are not upgradable. For example, entering “improve UWP buy/sell” will give an error message. You cannot buy or sell improvements on properties if you do not own the corresponding monopoly, or you mortgaged one of the properties in your monopoly. Entering “improve PAS buy” will lead to such an error since joe has mortgaged HH.

Related function calls are void sellImprovement(Property * up) and void buyImprovement(Property * up) in player.cc, void improve(Player * player) void sellimprove(Player * player) in upgradable.cc, property.cc, residence.cc and gym.cc.

Mortgage/Unmortgage:

During one’s turn, a player can mortgage any properties he/she owns to the bank and get half of the price of the property back. To mortgage a property, the player has to sell all possible improvements on that property. A player can also unmortgage any properties he/she has mortgaged to the bank by paying 60% of the price of the property. The general command style is “mortgage/unmortgage <property_name>”.

For example, to mortgage UWP, the player can enter “mortgage UWP”. To unmortgage “CIF”, the player can enter “unmortgage CIF”.

-Exception-property finding

If the players misspell the name of the academic buildings, or the input <property_name> is not a property that can be owned by players, the program will output "I cannot find this property." to standard output. The player can re-enter the command or enter other commands to continue the game.

-Exception-mortgage

-If the player does not own the property, he/she will get the error message: "You are not the owner of this property."

-If the player has already mortgaged the property, he/she will get the error message: "You have already mortgaged the property.".

-If the property is going to be mortgaged is an academic building and the player has improvements on the property, he/she will get the error message: "You need to sell all the improvements."

-Exception-unmortgage

-If the player does not own the property, he/she will get the error message: "You are not the owner of this property."

-If the player did not mortgage the property before or the property is already unmortgaged, he/she will get the error message: "You have already unmortgaged the property.".

-If the player does not have enough money to unmortgage the property, he/she will get the error message: "You don't have enough money.", and prevent him/her from unmortgaging the property.

-Demo – mortgage

Try loading “demo_mortgage.txt”. Joe can mortgage UWP by entering “mortgage UWP”. If the property is not owned by joe, for example, if you enter “mortgage RCH”, an error message will be thrown. You cannot mortgage properties that have improvements on them. Therefore “mortgage AL” is forbidden. To unmortgage a building, for example NH, enter “unmortgage NH”.

Related function calls are void getMortgage(Property * up) and void getUnmortgage(Property * up) in player.cc, void mortgageBy(Player * player) and void unmortgageBy(Player * player) in upgradable.cc, property.cc, residence.cc and gym.cc.

Trading:

Trading in watopoly is called by the following command “trade <name> <give> <receive>”, where give is the name of the player the current player is offering a trade to, give and receive are properties or money that the current player is giving and receiving respectively.

Start by loading in “tradetest1.txt” and input the following command: “trade vishal PAS CPH”, the trade will go through as it is a valid trade option.

A player can trade money for property, property for property and property for money. A player cannot do the following with the trade command: they cannot trade money for money; they cannot trade give or take money if there isn’t enough to give or take; a player cannot trade themselves; a property not owned cannot be given or asked for either. One more non-valid trade is trying to trade properties that have an improvement on it or the properties from its monopoly set have improvements on it.

For e.g. input the following command “trade vishal RCH EV2”, you will get a statement saying “You cannot trade a property that has improvements on it!

And this makes sense as monopolies where there are improvements on any of the properties cannot be trade until the improvements are sold.

Try this: input the following command “trade vishal RCH EV3”, you will get a statement saying: "You can't trade a property where the other properties in its monopoly have improvements on them!".

This makes sense also because as you saw earlier, EV2 which is part of the ENV monopoly has an improvement on it. So, any property in a monopoly, where the properties in the monopoly have improvements cannot be traded.

Code for trading found in board.cc in: void trade(const std::string &from, const std::string &to, const std::string &give, const std::string &receive).