Property Tycoon

https://www.property-tycoon.com/

Team Number: 1

Team Name: Team 💯

Team Members:

Peter Lloyd

Liam Scriven

Guy McClenahan

Elliot Massen

Sam Kennard

Problem definition

Our client, Watson Games, came to us with a brief requiring the creation of a digital version of their game: *Property Tycoon*. The client has 50 years of experience making board games, however has no experience with electronic games. The owner, Quentin Raffles, asked our company to digitise the game in the same format, stating that the game should be "fun to play," and have a "colourful and intuitive interface" that reflects the "spirit and character of the original board game".

Watson games are the UK leaders in board games, having produced many popular board games since their inception in 1963. They have produced many classics over the years; however, they have come into some hardship due to "Mousetrapgate". Following this, and their direct competition with Hasbro's Monopoly, they have decided this electronic game is the way forward.

In order for our team to complete this task, we began by creating a set of formal requirements for the development team including functional, non-functional and domain requirements. This allows us to see exactly what needs to be researched (domain requirements), what needs to be coded in (functional requirements), and the criteria for the game (non-functional requirements). Following the completion of this document we created a PERT and GANTT chart along with a Trello board to keep up to date with outstanding tasks.

As you can see from the PERT Chart below, there is no defined critical path. The reason for this is that all defined functions created by the team were critical to the next stages of development, as we implemented a waterfall methodology to complete the project. We did however suspect that problems would arise in the development stage as the coding of the game can hit barriers far faster than any other section. We saw that the primary issue would be the game engine and user interface development, hence we knew that by that point in the project we could not be using any extra buffer time.

We expected the project to take from the 5th of February to the 17th of April (10 weeks), including time to complete all documentation of the program. The project closely followed waterfall sections, using standard sections commonly seen in this methodology.

GANTT CHART

ORGANISATION

Team structure
Meeting organisation
Task Delegation
GitHub Repo
Trello Board
Google Drive for minutes & agenda's

12th Feb - 19th Feb

REQUIREMENT DOCUMENTATION

User requirements
Domain requirements
Functional requirements
Non-functional requirements
Interface requirements
Game requirements
Testing plan

12th Feb - 19th Feb

MODELLING DOCUMENTATION

Software:
High level:
Flow chart
Use Case Diagrams
Graphics:
UI wireframes
Game objects

Low level:
UML diagrams
Sequence diagrams

19th Feb - 5th Mar

DEVELOPMENT

Back-end:
Database
Game engine

Front-end:
Interface
Graphics

Unit testing

5th Mar - 5th Apr

SOFTWARE DOCUMENTATION

5th Mar - 5th Apr

TESTING DOCUMENTATION

5th Mar - 5th Apr

TESTING

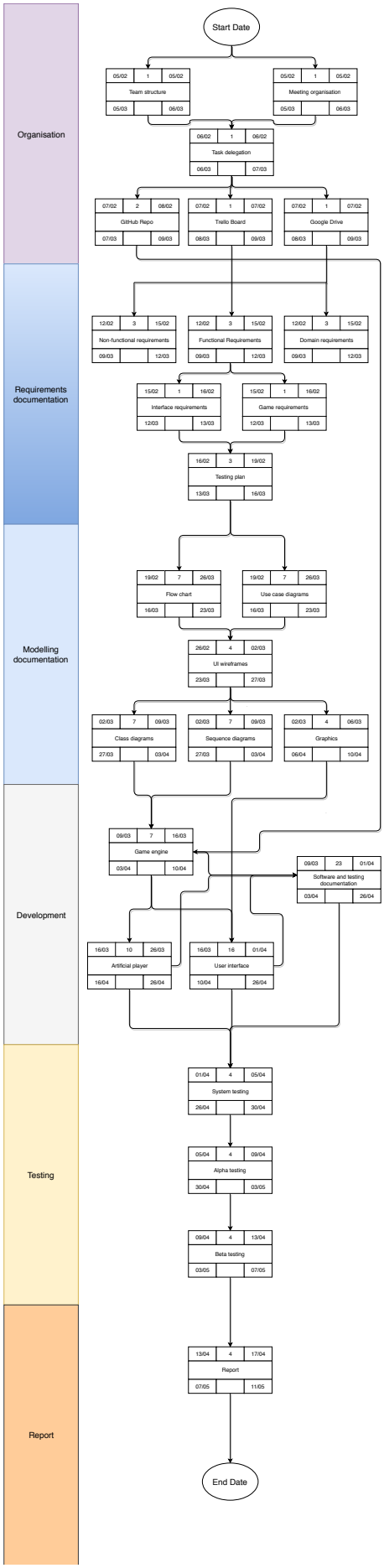
System testing
User testing
A|B testing

5th Apr - 12th Apr

REPORT

12th Apr - 4th May

documentation created on the 20/02/2018


Organisation
Requirements documentation
Modelling documentation
Development
Testing
Report
Start Date
05/02 1 05/02
Team structure
05/03 06/03
05/02 1 05/02
Meeting organisation
05/03 06/03
06/02 1 06/02
Task delegation
06/03 07/03
07/02 2 08/02
GitHub Repo
07/03 09/03
07/02 1 07/02
Trello Board
08/03 09/03
07/02 1 07/02
Google Drive
08/03 09/03
12/02 3 15/02
Non-functional requirements
09/03 12/03
12/02 3 15/02
Functional Requirements
09/03 12/03
12/02 3 15/02
Domain requirements
09/03 12/03
15/02 1 16/02
Interface requirements
12/03 13/03
15/02 1 16/02
Game requirements
12/03 13/03
16/02 3 19/02
Testing plan
13/03 16/03
19/02 7 26/03
Flow chart
16/03 23/03
19/02 7 26/03
Use case diagrams
16/03 23/03
26/02 4 02/03
UI wireframes
23/03 27/03
02/03 7 09/03
Class diagrams
27/03 03/04
02/03 7 09/03
Sequence diagrams
27/03 03/04
02/03 4 06/03
Graphics
06/04 10/04
09/03 7 16/03
Game engine
03/04 10/04
09/03 23 01/04
Software and testing documentation
03/04 26/04
16/03 10 26/03
Artificial player
16/04 26/04
16/03 16 01/04
User interface
10/04 26/04
01/04 4 05/04
System testing
26/04 30/04
05/04 4 09/04
Alpha testing
30/04 03/05
09/04 4 13/04
Beta testing
03/05 07/05
13/04 4 17/04
Report
07/05 11/05
End Date

Organisation

During the organisation of the project, we discussed different methodologies for the development. We decided upon using waterfall with some parts of agile, such as updating previous documentation or progressing with other parts of the project when previous parts have not yet been completed. The reason for this was to ensure timekeeping was effective.
Requirements Documentation

The requirements document was the integral part of the development, as reviewing the project aims given by the client allowed us to set clear defined goals for development achievement and helped us outline the end result of the project. We split the project up into functional, non-functional and domain requirements: functional requirements could be given to the development team to begin development and non-functional requirements would help to outline the project aim. Finding any domain requirements, we would need to clarify with the client and get assistance with from outside sources.
Modelling documentation

The modelling documentation consists of high-level and low-level diagrams. The low-level diagrams consisted of class diagrams and sequence diagrams. The high-level consisted of flowchart and use case diagrams. These diagrams outlined to the development team exactly what was expected from the project and what deliverables were required in a clear format, showing exactly how the system should act at any given moment.
Development

For the development of the software, the group was split into three teams that worked on the game engine, the user interface and the AI player respectively. We also set up a GitHub repository to keep track of changes to the project whilst it was being developed. This allowed the whole team to have access to the current version at all times.
Testing

In order to test the game, we incorporated unit testing, testing every class created in the game engine. We also used a library called TestFX to test the user interface.

Trello

We then created a Trello board to delegate tasks. Trello is a web-based project management application that allows individual tasks to be posted and assigned to members. These tasks are stored on boards and allow for headings such as “To do”, “In-Progress” or “Complete”. After each team meeting the board was used to make notes of outstanding actions and also as a place to discuss specific pieces of work remotely.

https://trello.com/b/nkADfSPk/monopoly

Requirement Analysis

Domain requirements

	Requirement	**Reason/how**	**Specification quote**
1	Fun to play	This is speculative and will require insight. However, to do this we will use an intuitive interface that ensures the game is "fun"	"The game should be fun to play and have a colourful and intuitive interface that reflects the spirit and character of the original board game."
2	Reflects the spirit and character of the game	This will need to be clarified, however this is based around the colours and interface thus we will confirm that our idea for the interface matches what the client is aiming for.	"The game should be fun to play and have a colourful and intuitive interface that reflects the spirit and character of the original board game."

Non-functional requirements

	Requirement	**Reason/how**	**Specification quote**
3	Able to run on both windows and mac OS.	This gives wide compatibility allowing the game to be played on the majority of computers. To do this we will choose a language which will compile on either operating system.	“ The electronic version should be for desktop machines, and ideally should be playable on both Mac and PCs. If this is difficult, then PC development should be preferred. “
4	Possible app development and reskin of game.	The code must be highly ambiguous. The user interface will be made separate to the core game code, and the language used will reflect possible app development.	“ There are no plans for a mobile version at this stage. “
5	Use java to program the game mechanics.	This is a language well known to the development team which is versatile and allows us to use layers of ambiguity, split tasks and fit with any future modifications the client wishes to make.	“ The electronic version should be for desktop machines, and ideally should be playable on both Mac and PCs. If this is difficult, then PC development should be preferred. There are no plans for a mobile version at this stage “

Game requirements

	Requirement	**Reason/how**	**Specification quote**
6	Two different game modes, "full game" and an abridge version.	There will be a "game controller" class that controls the overall running of the game, here we will implement the time limit for the abridged version and allow the full version to be played.	" The game can be played in two versions: **The full game:** In the full version, the game is played until there is only one player left and all other players have retired from the game due to bankruptcy, or because they have decided to leave the game with the agreement of the other players. In the latter case, all of the player's property and funds are returned to, and become the property of, the bank. **The abridged game:** In the abridged version, a time limit is agreed at the outset by all players. When the time limit is reached, and the players have all taken the same number of turns, the game ends. Each player then calculated the value of their game assets. The player with the greatest value of game assets is declared the winner. "

7	The full game must play until only one player remains.	This is controlled by the game controller.	“ game is played until there is only one player left “
8	A player can choose to withdraw from the game, their assets go to the bank.	Should a player withdraw, the game controller will call a method that removed the player and moves their assets to the bank, then returns control to the game controller.	“ They have decided to leave the game with the agreement of the other players. In the latter case, all of the player’s property and funds are returned to, and become the property of, the bank. “
9	In the abridged version, a time limit is agreed at the start.	This will be done in a menu screen. “All players” will not include the computer opponent.	“ In the abridged version, a time limit is agreed at the outset by all players. “
10	After the time is reached a winner is found by asset value.	This will be determined by a constant running total of each players worth being compared.	“ The player with the greatest value of game assets is declared the winner. “

Functional requirements

Money

	Requirement	**Reason/how**	**Specification quote**
11	Each player begins with £1,500, allow for possibility of more or less.	Allowing for future expansion, this can be changed on the input file.	“ At the outset of the game, each player has £1,500 in cash. “
12	A player receives £200 when they pass go.	This will be attached to the amount of tiles the player has moved over. In the standard 40 tile game, the player will receive £200 every time they pass an increment of 40 tiled passed.	“ When a player passes Go, they receive £200 from the bank. “
13	If a player cannot pay the rent, they must sell their assets until they can afford the price of the rent.	Players can choose which assets they sell, if their player value is less that the rent, they are automatically disqualified.	“ If a player is unable to pay the rent for a property they have landed on, they must sell game assets to make good on the rent. “
14	If there are houses on a property, a player that lands on the property must pay the corresponding price for the number of houses on the property.	The price will be a part of the property object, the object will know how many houses are on it and can charge accordingly.	“ If a property is improved with houses or hotels, then the rent to be paid is as shown on the card. “

15	The price to buy a house or hotel is as shown on the card for the specific property in question.	This is an attribute of the property object.	“ Houses and hotels are purchased for the amount shown on the game card. “
16	A property can be sold back to the bank for its original card value if no houses/hotels are present.	There will be a sell function in the property object.	“ If a player needs to raise funds, they can sell a property back to the bank for its original value “
17	If a player lands on a property, and the owner possesses the whole street, the rent is double that shown on the card.	The rent will be taken from the initialisation file on game start up, when a property is purchased, it will check the ownership of the other properties in the street and mark a Boolean value in each as true if they are owned by one player. **This needs to change if trading is brought in!**	“ If a player owns all of the properties in a colour coded group, but the properties are otherwise not developed further with houses and hotels, then the rent due is doubled. “

18	A property may be mortgaged to the bank for half of its original card value, while mortgaged, the owner cannot collect rent.	There will be a mortgage function in the property object.	“ If a player needs to raise funds, they may mortgage a property with the bank. The bank will pay the player one half of the value of the property as shown on the game card. No rents may be collected for that property whilst it is under mortgage. “
19	Should a player land on free parking, they gain the money from the unclaimed fines of all players currently on the free parking square.	The property object for free parking will contain its value, this will be transferred to the player upon landing on the square.	“ When a player lands on free parking, they collect all of the funds currently on the free parking space. “
20	A player in jail may choose to get out, by paying £50.	Jail will have a charge function that takes precedent over the dice roll function, the fine is put in free parking.	“ If a player is sent to the jail, they may pay £50 to be released from jail. “
21	A player can trade game items with the bank, not including get out of jail free cards or their player token	After dice roll, this is offered along with buying houses.	
22	A player may not borrow money from other players or the bank.	This could make the game go on exponentially, thus it is not aloud.	“ Players may not borrow or lend money from each other, and may not borrow money from the bank. “

23	All fines are paid to free parking.	Fines are transferred to the free parking property object, they can then be collected by the player who lands on the property.	“ Where fines are to be paid, the proceeds accumulate on the free parking space in the centre of the board. When a player lands on free parking, they collect all of the funds currently on the free parking space. “
24	Property cards are transferred to a player upon payment.	The player will choose to buy a property, provided they have the funds they will be taken and the card ownership will move to that player.	“ When a player purchases a property, the card is transferred from the bank to that player and the amount shown on the card is paid to the bank. “
25	A players assets are calculated by: Cash Property value (Original set card purchase value, not auctioned price) Mortgaged property value (Half of original value) Houses and hotels at original price Get out of jail free cards.	This calculation will happen every time a player has a turn, for them, and any player they have interacted with. A get out of jail free card is worth £50.	“In the abridged version, the value of a player’s assets is calculated by adding up: Cash held The value of properties as shown on the game card, unless the property is mortgaged, in which case the value is half the value shown on the game card. The value of houses and hotels purchased for each property. “Get out of jail” free any other card items have no cash value. “

26	Bank has unlimited funds.	The bank will start with a very large number of money that will stop it from running out, or use functions that avoid the need for the bank to have any numerical value of money.	“ The bank has a total of £50,000 cash. “
27	Property and houses must be paid for upfront.	provided they have the funds they will be taken and the house will move to that players property.	“ All assets procured from the bank must be paid for in cash. The bank does not provide credit. ”

Dice

	Requirement	Reason/how	Specification quote
28	If the dice rolled are the same number, the player rolls again, this is called a "double".	This will be a part of the dice roll method.	" If a player throws a double, then they take another turn. If a player throws another double at the third turn, then they "go to jail". When a player goes to jail, they go directly and do not pass Go. "
29	Should this happen multiple times, if the third roll is a double, the player goes directly to jail.	This will be a part of the dice roll method, then hand back a value telling the game controller to move the player to jail, or call a go to jail function elsewhere.	" If a player throws a double, then they take another turn. If a player throws another double at the third turn, then they "go to jail". When a player goes to jail, they go directly and do not pass Go. "
30	After a turn (Dice roll) a player may choose to buy houses/hotels for their property, given that they own the whole street. This cannot be done at any other time.	The game controller will offer this after a dice roll.	" When a player has finished moving their token, and has completed any property purchase activity, they have the option to buy houses and hotels to improve their properties. "

31	Two fair dice.	Dice must be fair for all player to have an equal chance of winning.	“ The dice used in the game must be fair with each dice have an equal probability of landing on one of its six sides. ”

Unclassified rules

	Requirement	**Reason/how**	**Specification quote**
32	Game is designed for 2 players minimum and has a default of 6 maximum.	The maximum of 6 can be removed at any time in the input file, provided a player piece is also given. This allows future game expansion.	“ The game is for 2-6 players. “
33	Currently 6 pieces, but possibility to add more – “Boot, smartphone, goblet, hatstand, cat and spoon**”.**	This allows for future game expansion. The pieces can be added to a game file, and max players can be changed on the game start up files.	“ The tokens are: boot, smartphone, goblet, hatstand, cat and spoon. “
34	Corner pieces could be in any order, and anything other than the Go square could change, the changes must be within the set parameters that they must be either a “go to” task, a “pay” task or a “receive” task.	The corner pieces will be decided in the order given in the input file, the only square that must exist is the go square.	“ Board spaces may consist of properties, a “pot luck” space, an “opportunity knocks” space, “free parking”, the jail/just visiting space or a space with specific instructions that must be followed by the player. “

35	Should a player remain in jail, they forfeit their next two turns (They have the option to pay the £50 on these turns but cannot roll after paying).	Player objects have an attribute to check if they are in jail.	“ If a player opts to stay in jail, they give up their turn for the next 2 rounds. Whilst in jail, a player may not collect any rents from other players. At the end of the next 2 rounds, the player token is moved to “just visiting” and the player turn ends. The player takes a normal turn in the next round. “
36	If a get out of jail free card is owned by the player they may use it to get out of jail on that turn, to move on the next turn	Player is moved out of jail, player object attribute set to out of jail.	“ If a player has a “get out of jail free” card, then they place the care at the bottom of the “pot luck” or “opportunity knocks” pile as appropriate, the player token is moved to “just visiting” and the players turn ends. The player takes a normal turn in the next round. “
37	Players start on the go square.	Players will start on the go square, this must not automatically give £200.	“ At the outset, all players start on the board space labelled Go and move clockwise around the board. “

38	Players move clockwise.	The dice function will move the players forward.	“ At the outset, all players start on the board space labelled Go and move clockwise around the board. “
39	Used “opportunity knocks” and “Pot luck” cards (Or anything they are renamed to) must be placed at the end of the queue.	Queue data structure used to allow for cards to be picked up the same way as in the game.	“ At the outset of the game, the two packs of cards labelled “pot luck” or “opportunity knocks” are shuffled and placed on the board. When cards are takne, they must be replaced at the bottom of the corresponding pile. “
40	“Opportunity knocks” and “Pot luck” cards (Or anything they are renamed to) must be randomised in order at the start of the game.	To ensure fairness for the players. This can be done in the initialisation function.	“ At the outset of the game, the two packs of cards labelled “pot luck” or “opportunity knocks” are shuffled and placed on the board. “

41	“Opportunity knocks” and “Pot luck” cards can be renamed, each card can be renamed, and the cards have the ability to do one or multiple of a predefined list of functions.	Allows for the game to be updated at any point, the data for these and the predefined function list will be in the initialisation file.	“ At the outset of the game, the two packs of cards labelled “pot luck” or “opportunity knocks” are shuffled and placed on the board. “
42	All property originally belongs to the bank.	All property will belong to the bank object at the start of the game.	“ All properties are initially the property of the bank. “

Property

	Requirement	**Reason/how**	**Specification quote**
43	Players must complete a circuit of the board before buying property.	This will be a Boolean in the player class.	“ Players may not purchase property until they have completed one complete circuit of the board by passing the Go space. “

44	Should a player land on a property on their second lap of the board and choose not to buy it, the property is auctioned. If there are no bids, the property is unsold.	There will be a property buy method called if a player lands on a property owned by the bank, this will offer the player to buy the property, or begin the auction.	“ Once a player has made their move, if they land on a property that has not yet been purchased, they have the opportunity to buy that property. If they decide not to buy that property, then the property is auctioned by the bank. “
45	Any bidding player must have completed a lap of the board.	Any player that has passed 40 tiles (Unless the size of the game board has been changed).	“ All bidding players must have completed one circuit of the board. “
46	If a player lands on an opponent’s property, they must pay the rent as shown on the card.	The rent is retrieved from the game initialisation file when the game is launched, there will be a check landed space method as part of the player class that checks the ownership of a property and can check the rent price and pay accordingly.	“ If a player lands on a property owned by another player, they must pay the player who owns the property the value of the rent shown on the card. “
47	There can be a difference of no more than one house on the individual properties of a street.	When buying a new house, the object will check the house amounts on the other properties in the street.	“ there may never be a difference of more than 1 house between the properties in that set. “

48	A player in jail may not collect rent.	Properties must check the position of their owner when rent is requested, thus a getRent() function must be created in the property object.	“ Whilst in jail, a player may not collect any rents from other players. “
49	Maximum development for a property is one hotel. The purchasing order is houses up to 4, then these are swapped for a hotel (Hotel is still paid for).	This can be changed in the initialisation document.	“ The maximum development permitted on any one property is one hotel. “

AI Requirements

	Requirement	**Reason/how**	**Specification quote**
50	Computer opponent (AI Player).	Extra players in the game, allows for single player game. Simple rules will be made to allow the opponent to make decisions to buy sell, trade, etc. This can then be replaced with a machine learning algorithm to make a better player.	“ **A game player agent:** An agent that can take the role of 1 or more of the players. This would allow for a limited number of human players to enjoy a richer gaming experience. “

51	One or more Artificial player can be in a game.	This helps with machine learning training and is also a desired functionality from the client.	“ An agent that can take the role of 1 or more of the players. “
52	Able to play to the same extent as a human player.	The initial version will use a rule-based system as to buy and sell, then may progress into a machine learning algorithm.	“ The game player agent should be able to play the game to the same extent that a human player would. ”
53	Able to have a fully autonomous game.	This helps with machine learning training and is also a desired functionality from the client.	“ it also provides the possibility for fully autonomous play when all of the players are provided by the program “
54	It can thus buy and sell property, bid for property and choose to buy houses.	This will be a part of the initial rules, and a key part of the algorithm used for machine learning.	“ The game player agent would roll moves and buy/bid for property. “

55	The autonomous player cannot withdraw from the game.		“ A game player agent may not opt to retire from the game. A game player grant only leaves the game when they are bankrupt. ”
56	“Fun to play”	Should pause between plays to make it seem like you’re playing a human	“ The game should be fun to play and have a colourful and intuitive interface that reflects the spirit and character of the original board game. “

Input files

These should contain all values for game aspects that have the capacity to be changed, this includes but is not limited to –

- Opportunity knocks and pot luck card names and functions
- Property names and prices
- Game board size
- Max players
- Player “pieces” (Character Images)

This allows for maximum ability to modify and update the game

Interface Requirements

	Requirement	**Reason/how**	**Specification quote**
57	The board must have 40 squares.	This can be modified in the initialisation file of the game, but the default will be 40.	This is never written formally, however the original game is 40 squares as seen by the given property titles and prices.
58	The game must be “Fun and intuitive”.	Can be made with a simple easy to use interface. The designs will be sent to the client for assessment and will be easy to change.	“ The game should be fun to play and have a colourful and intuitive interface that reflects the spirit and character of the original board game. “

59	Ability to see the current worth of a player in terms of property and cash value.	The calculation for this will already be done per term, so this value will be displayed on the screen or a sub screen.	“ **A means of monitoring the performance of the simulation:** including the current worth of each of the players and the property assets that they own. “
60	The ownership of all properties must be available to see by all players at all times.	This will be displayed on the screen or a sub screen for anyone to see at any time.	“ The range of properties available for sale by the bank, and owned by players, is a matter of public record and that information must be available to all players at all times. ”

Expansions

	Requirements	**Reason/how**
61	Machine learning for computer opponent.	Create a better player for humans to play against, this can be done with a machine learning algorithm which can learn by playing itself until it plays the game well. This could also allow us to make different difficulties of player.

Testing

To test the game, we will be implementing test driven development. This means we will create unit tests before we create any classes or methods designed to test the validity of the code we have created. By doing this we can eliminate most logical errors made.

We will also implement systems testing, in which after the base code is created, we will test the system as a whole to ensure it meets all of the brief requirements and no “bugs” are present in the code.

Classes

Updates to functional requirements made through emails with the client

	Requirements	**Question**	**Email response quote**
62	Player pays appropriate fee shown to unmortgage a property.	“Will there be a penalty for players un-mortgaging previously mortgaged property? “	“A: No. The player will need to pay the appropriate fee to unmortgage the property. That fee is one half of the property value as shown on the property card. That fee is paid to the bank. Remember that no rents can be collected from a property whilst it is mortgaged. “
63	Player must make any bids in an auction using existing cash.	“Q) During bidding can a player sell some of their properties in order to make a higher bid? “	“A) A player cannot sell assets outside of their turn. Bids have to be made using cash assets in their possession at the time the bid is made. “
64	Maximum amount of players may change in the future after the release, thus must be made as easy to amend as possible.	“Q) Amount of players. The game was limited to 6 players as the bank had a finite amount of money, are we following the 6-player maximum rule? If so, would expansion of player quantity limit be something you may look at in the future? “	“A) We did some thinking on this. The current game has a limit of 6 players imposed by the player tokens. There is no particular reason why it can't be more than 6, but in practice we found that with more than 6, the chances of any player actually winning was too low and made the gameplay boring. So, 6 will remain the maximum number of players. “

65	Player pieces should be imported on launch, as to be able to implement this feature in the future.	“Q) If there is the ability to add more players, could you suggest more player piece objects; further to the hat-stand, spoon, etc. Alternatively, we can incorporate a way of adding pieces at any time. “	“A) Whilst there are no more players, it might be nice if the pieces could be customizable. “
66	An auction begins at £1.	“Q) Is there a starting price for an auction, or does the price begin at 0? “	“A) The starting price should be £1. “
67	The bank can buy houses, hotels and property that the player owns.	“Q) What "Assets" can the bank buy? “	“A) Any property asset, house or hotel. Get out of jail cards and the tokens have no value. “
68	If a player cannot afford to pay rent, everything they own is given to the bank, and its worth given to the player whom is owed rent.	“Q) What happens to a player's assets if they can't pay another player? Are they mortgaged, then passed on to the player that they owe money? “	“A) Assets are sold to the bank to raise cash to pay the player. If a player cannot raise enough cash, all the cash proceeds from sale of assets to the bank is paid to the player and the bankrupt player retires from the game. “
69	A property sold to the bank becomes available for sale when a player lands on that tile.	“Q) Is a property that goes back to the bank auctioned or goes back to being for sale when someone lands on the property? “	“A) The property becomes available for sale when someone next lands on that property. “

70	There should be a free parking spot in the centre of the board showing the value currently on the free parking tile; in addition to the tile the player can land on.	“Q) It says there is a free parking square in the centre of the board, is this alongside a free parking square on the playable board (The squares the player will circulate)? “	“A) In the original game, the fines collected for free parking area placed on a space in the centre of the board. This is n addition to the space on the main board. “
71	A player can pay the get out of jail fee as soon as they are sent to jail, then move on the next turn.	“Q) Does a player have the option to pay the £50 to get out of jail the second they go to jail, or on the next turn? “	“A) Yes, they can opt to pay the fine immediately. Their token is then moved to free parking and they move on their next turn. “
72	A player who does not pay the fee must stay in jail for 3 turns, then are able to roll and move on the fourth.	“Q) The specification says, "Get out of jail on two turns", do you roll on the third and move along the board or get out on the third? “	“A) You get out on the third. So, your token moves to just visiting and then you make a move as normal on the next turn.
73	Player has the option to use their get out of jail free card	“Q) Do you have to use a get out of jail free card? “	“A) No. But it has no resale value. “
74	A player can withdraw from the game, regardless of game type at any time.	“Q) . Can you withdraw from the abridged version? “	“A) A human player may withdraw from either version of the game at any time. If they withdraw for any reason, all their assets are returned to the bank. A player game agent may not retire of its own choosing for any reason whatsoever. “

75	To choose who starts the game, each player rolls the dice and whoever rolls highest will start. Following this, the player order will be in order in which the players were entered into the system.	“Q) Another question is who starts the game, at the moment we are operating on the basis that each player rolls dice and the highest score starts, is this acceptable? “	“A) That would be in the spirit of the original game. “
76	A player turn ends when they are sent to jail, however they still have the option to pay to get out of jail before the turn ends.	“Q) While testing an instance came up that needs clarification, this is if you roll doubles and land on a pot luck card where you are sent to jail, are you allowed to take the extra turn from the doubles? (either from within jail or just visiting if the player paid). “	“A) A player's turn ends upon being sent to jail. “
77	Amount of tiles will not change from the original game.	“Q) Can the user customise the amount of tiles, i.e. removing tiles wholesale from the spreadsheet or adding their own tiles? “	“A) The amount of tiles should remain fixed as in the original game. “
78	The external configuration file could be changed at the beginning of a game, changing all prices of everything on the board.	“Q) Can the user modify the 'rent' of the utilities, stations and property tiles? If so what standard input should we expect to receive from the utilities and stations notes section in order to know how to modify their rent? “	“A) The rent is set at the time the game is initialized. The rents are initialized at whatever value was contained in the external configuration file. “
79	A random option should be available for selecting a player piece for players that don’t mind what they play as; in addition to the selection.	“Q) Are tokens allocated to players randomly or can players choose? “	“A) It would be nice to be able to choose (you can in real life). But equally a random option is useful when players don't care. “

80	Time limits for the length of a turn should be set for the abridged version of the game.	“Q) Should we implement a timer/turn limit for the players on the abridged version? - A timer specifically so people cannot get into the lead then waste time. “	“A) A timer would be in the spirit of the original game. "Filibustering" is for cads … “
81	If the property being sold back to the bank is part of a full street, it, and its fellow street properties must have no houses.	“Q) Can players sell back properties to the bank or only mortgage them? “	“A) Good point. A player is free to sell a property to the bank for the price paid originally for it. A property may only be sold when it is free of houses and hotels, when it is the player's turn. When a property is sold, no improvements are possible in that colour group. A player must own all of the properties in a colour group before any improvements can take place. “
82	A player makes 1 bid for a property. They can choose to remain anonymous if they wish, however all players will be playing on the same device, as such all players could be anonymous. Bids should not be displayed on the screen, so order of bidding is not unfair.	“Q: When a property goes to auction, the rules say that "Each player makes a bid to the bank." Does this mean that each player may only make a single bid at any one auction, or should the players be able to make multiple bids, much like how an auction would work in real life? “	“A) It is intended to work as a sealed bid system. Each player makes a single bid to the bank. A player may may the bid public or may choose to keep it private between them and the bank. Whether a bid is public or not is entirely down to the player and does not influence the bank's decision. The bank sells to the highest bidder. “
83	A player cannot start an auction if they are the only person who has completed a lap of the game board. Additionally, a player who starts an auction cannot bid for the property in question. Thus, for auctions to occur, 3 players must have	“Q) My colleague has brought to my attention that the first player that goes around the board could auction the first property they land on, then as the only bidding player, purchase the property for £1. We	“A) If only one player is able to purchase, then there cannot be an auction. That player simply has the option to purchase or not to purchase. “ “A) No, the player has already decided that they did not wish to purchase

	completed a lap of the board in order for a fair auction and to be within the spirit of the original game.	would like clarification on whether the property should not be able to go to auction when there is only one player able to bid. “ “Q) If a player decides not to buy a property, are they still allowed to be part of the auction for that property? “ “Q) In regard to passing go, does 2 or more people have to have passed go to trigger an auction or all players to have passed go. “	the property, so that would not be in the spirit of the original game. “ "A) To take part in an auction, a player must have passed go. So, to have a valid auction, 3 players would have need to have passed go. “
84	A player in jail cannot be a part of an auction.	“Q) Can the player join the auction if he/she is in the jail? “	“A) No. “
85	Should a player mortgage one property from a street but still own the others, the rent is still doubled on the non-mortgaged tiles.	“Q) If a player owns all of the properties with the same colour, and he mortgaged one of it, will the rent of them still be doubled? “	“A) Yes but remember that no rent is payable when a player lands on a mortgaged property. “
86	Trading will be implemented as an option at the beginning of a game. This type of trading allows players to trade their game assets mid-game, excluding player pieces and get out of jail free cards. Players in jail cannot initiate or be involved with a trade.	“After careful consideration, and consultation with representatives of the Property Tycoon Players Club (PTPC), we have decided to amend our original user requirements. The PTPC noted that with large numbers of players, the full game has a tendency to reach a stalemate as it can be hard for any individual player to acquire ownership of a complete coloured property group. They have suggested that there should be an option to allow players to trade properties. We have considered this suggestion and we believe that such an option would benefit the	

		playability of both the full and abridged versions of the game. However, we don't want to upset our loyal players who have been playing the classic rules for many years. We have therefore decided that for both the full and abridged game, we would like a user selectable option to allow players to trade properties. This would be a simple trading system where one player can swap with another player. These trades can be on any basis that the players see fit. A player may offer a trade at the end of their turn, provided that they have paid any debts or fines that they might have incurred on that turn. Trade offers do not have to be accepted by other players. Trades must not be for cash. Properties must be traded for properties. It is up to the players to determine whether the trade is fair and equitable. “

Feasibility study

In order to determine the feasibility of the project, we needed to look at three sections; technical, operational and economic feasibility. For the technical side, we discussed using different software distribution packages and programming languages that satisfied the requirements.

In the requirements of the game, it was stated that the game must be multi-platform desktop and allow for expansion in every possible area. Given these requirements, we shortlisted using Unity (in order to allow for the widest platform distribution) or Java (as the team was far more comfortable with the syntax).

As a team, we came to the conclusion that the time factor alone to perfect our Unity skills would outweigh the time to program and complete the game in Java to the same specification. We decided that the game would be built in Java using the IntelliJ IDE, as the compatibility with GitHub was intuitive.

In terms of the economic cost for the game, we aimed to ensure the completion of the project did not cost the client any unseen fees. As such, we utilised GitHub's private repositories to store the game code in order for the team to have full access; we then hosted the website allowing for the game download and documentation using Netlify. Netlify is a free service that ensures there will be no fees.

The operational concerns for the project were primarily in terms of a time factor. Following the creation of the Pert and Gantt chart we saw that it was feasible to complete tasks within the given time frame; however, we completed a risk analysis to ensure that given any situation the client would get the finished product on time.

Risk management plan

Risk	Probability	Effects	Plan	Redundancy	Monitoring
Lack of knowledge: the team does not have the collective knowledge to properly execute the project.	Medium	Big	Anticipate complex features in project plan and research their implementation.	Use researched resources to help obtain knowledge.	Hold weekly team meetings and ensure each member is comfortable with their tasks.
Permanent loss of member: a member of the team becomes permanently unable to continue contributing to the project.	Low	Big	Ensure well-being of all team members.	Reduce the scope of the project and re-distribute tasks.	Keep good communication channels and take attendance each meeting.
Temporarily loss of member: a member of the team becomes temporarily unable to contributing to the project.	High	Medium	Find out in advance of the temporary break and plan for it.	Reduce the scope of the project and re-distribute tasks.	In weekly meetings always ask about any upcoming absences.
Loss/overwriting of code/data.	Medium	Big	Use version control to collaboratively work on the project.	Re-pull the code from the version control system.	Ensure each member is confident in commiting code.
Project requirements change.	High	Medium	Design and build the project in a modular and abstract fashion that easily allows for changes. Allow for a buffer time in	Re-distribute tasks using the buffer time.	In weekly meetings make note of any comments from the client and discuss their effect.

			the plan to account for changes.		
Project requirements are misunderstood by the team.	Medium	Medium	Create requirements analysis and each check against the specification. Allow for a buffer time in the plan to account for changes.	Re-distribute tasks using the buffer time.	Consistently check our work against the project requirements.
The underlying technologies/languages do not perform as expected.	Low	Big	Anticipate complex features in project plan and research their implementation.	Use researched resources to help solve the issue. If not try an alternative.	Research any unknown technologies we want to use and find potential alternatives.
Team member(s) fall behind the plan.	High	Medium	Identify critical tasks and ensure those involved have resources they need.	Re-distribute tasks using the buffer time.	Hold weekly team meetings to keep accountability and offer help if needed.
Elements of the codebase do not work as expected.	Medium	Medium	Enstate practice of test-driven development (no code should be added if it doesn't have tests!)	Using buffer time, other members can help resolve issues with the codebase.	Each class should be compared to the low-level designs to ensure it meets the criteria.
Members of the team disagree on proposed solutions.	Medium	Big	Ensure all big decisions are made democratically in a public	Hold a section of the meeting to discuss the disagreement	Ensure that everyone is given the option and feels

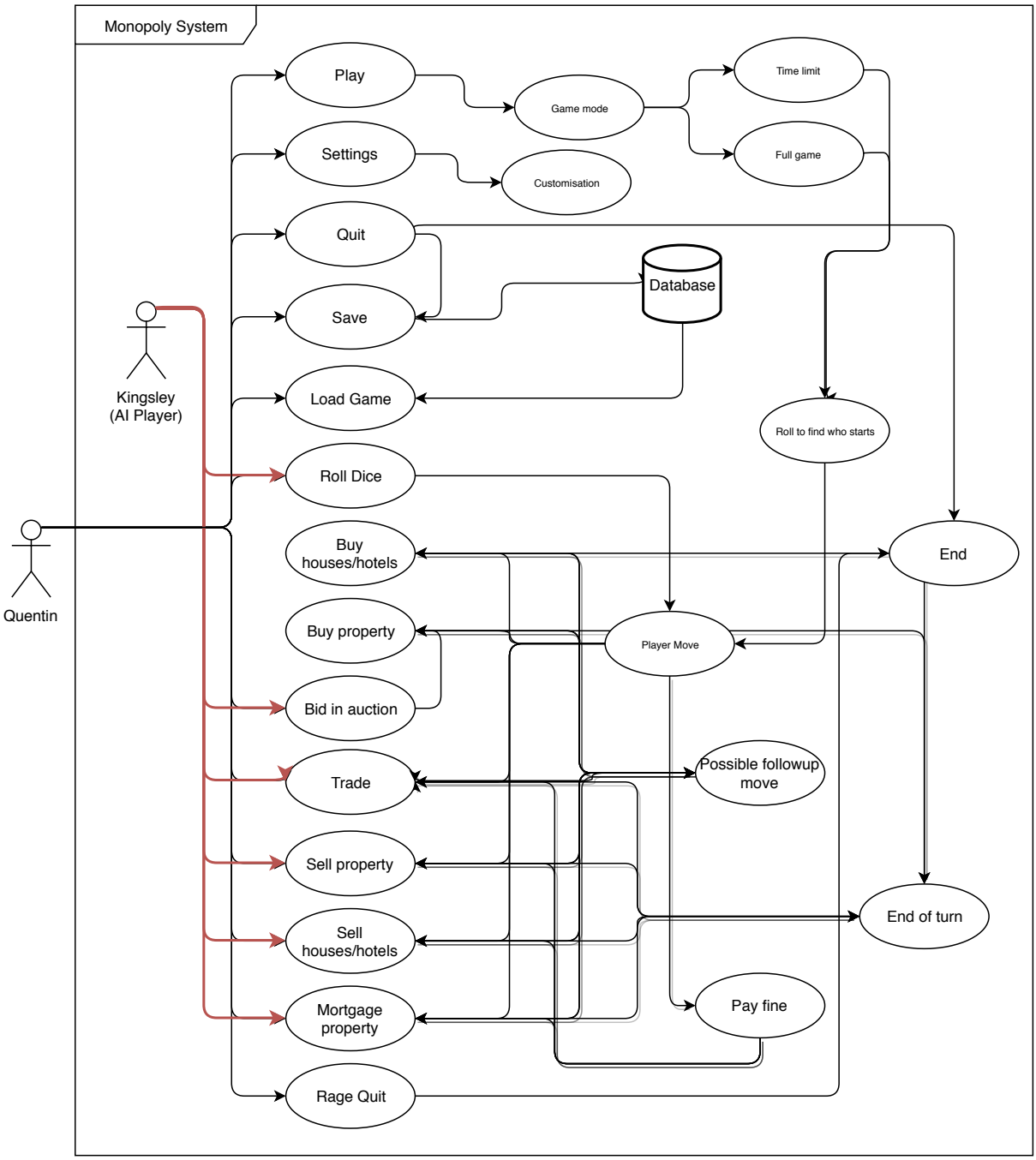
			space (group chat or meeting).	and come to a definitive agreement.	comfortable to present their solution to the team.
Lack of time for testing and QA	High	Big	Ensure requirements are descriptive and clearly state expectations of the software. Enstate practice of test-driven development (no code should be added if it doesn't have tests!)	Ensure any additional time remaining is used to every just one bit of testing and QA (some is better than nothing).	Hold weekly team meetings to keep accountability and offer help if needed.

Modelling

High level

Use case

Pre-trading

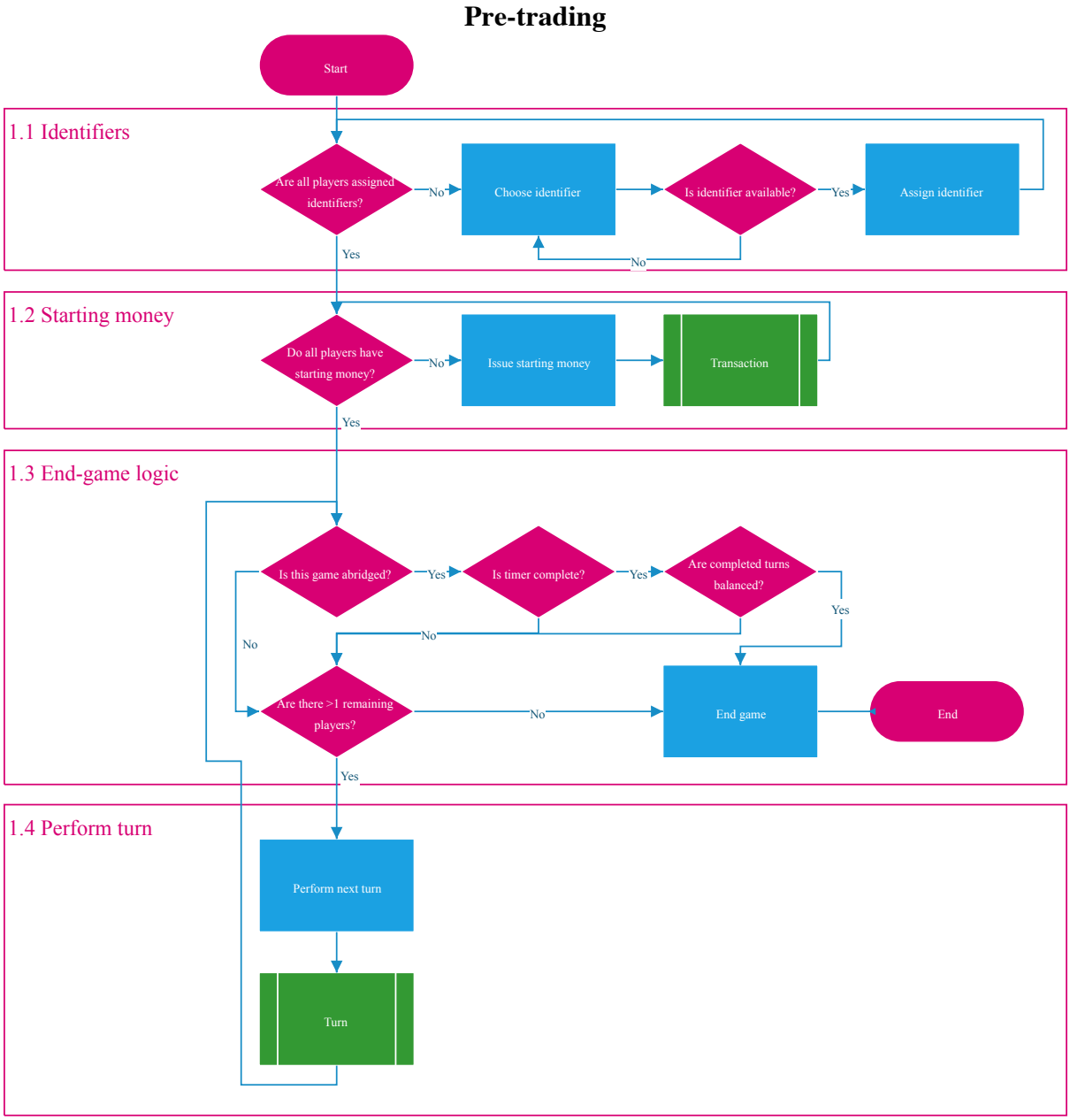


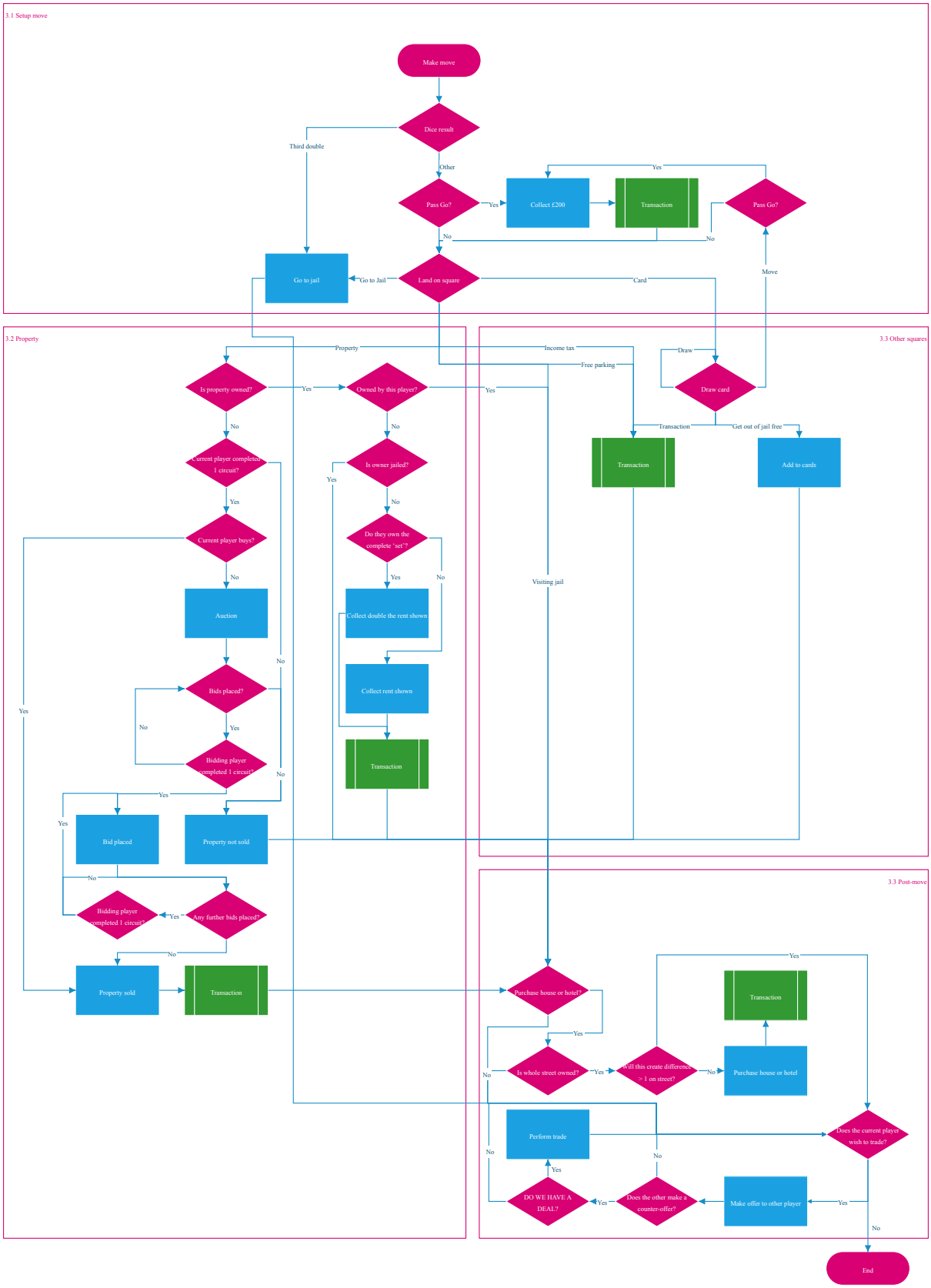
The use case diagram shows us exactly how a user will interact with the game platform. This diagram shows exactly all functionality the user will utilise when playing the game, and how certain functionality leads to one another.

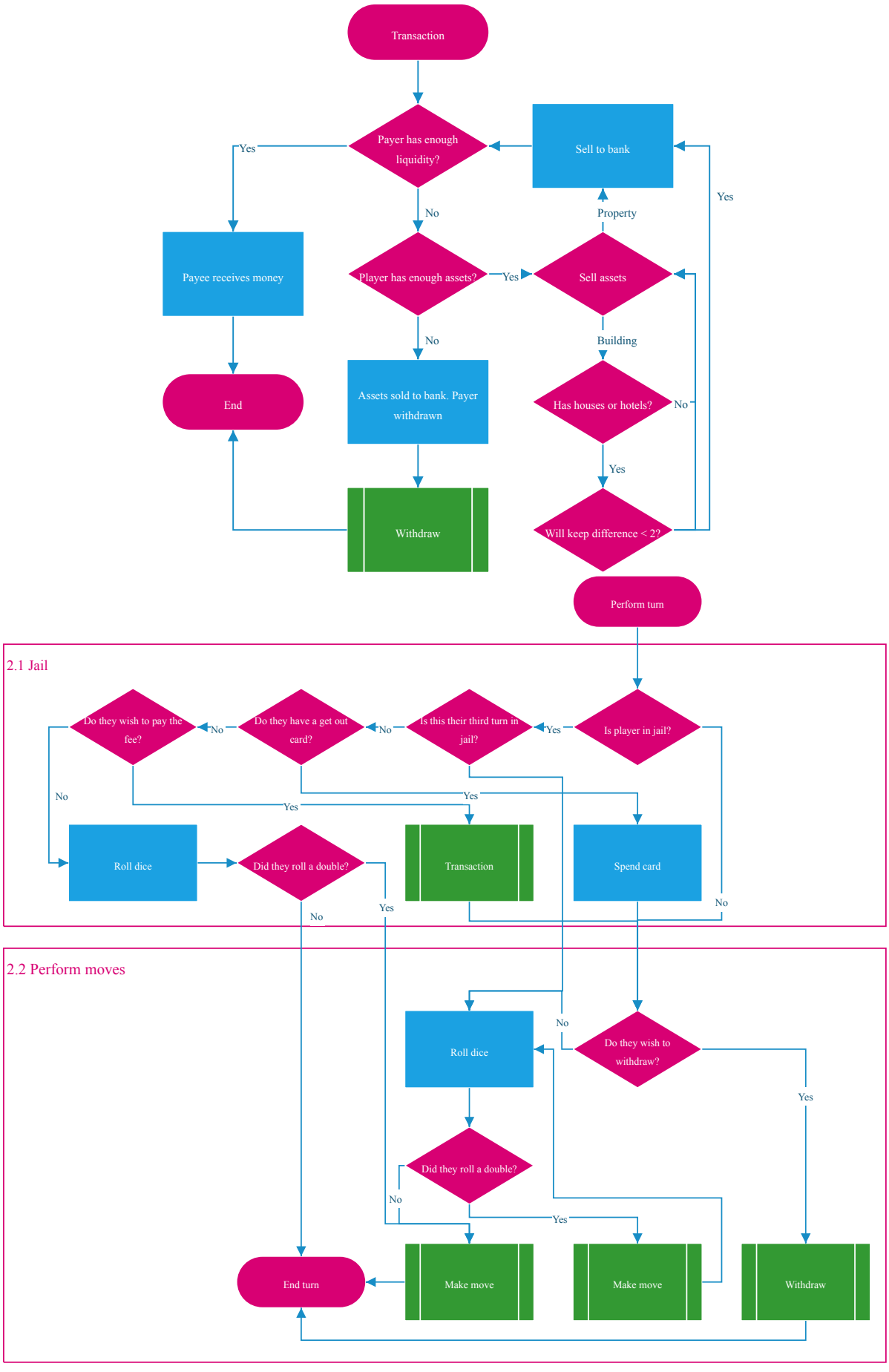
Post trading

Following clarifications from the client, the game now incorporates trading functionality. Above is the class diagram showing the changes; however, as the diagram incorporated trading as a function to be used with the bank, this change is simply an addition of the functionality of that method.

Flowchart




3.1 Setup move
Make move
Dice result
Third double
Other
Pass Go?
Yes
Collect £200
Transaction
Yes
Pass Go?
No
No
Go to jail
Go to Jail
Land on square
Card
Move
3.2 Property
Property
Is property owned?
Yes
Owned by this player?
No
Current player completed 1 circuit?
Yes
Current player buys?
No
Auction
Bids placed?
No
Yes
Bidding player completed 1 circuit?
No
Yes
Bid placed
Property not sold
No
Bidding player completed 1 circuit?
Yes
Any further bids placed?
No
Property sold
Transaction
Yes
Yes
No
Yes
No
Is owner jailed?
Yes
No
Do they own the complete 'set'?
Yes
No
Collect double the rent shown
Collect rent shown
Transaction
3.3 Other squares
Income tax
Free parking
Visiting jail
Draw
Draw card
Transaction
Get out of jail free
Transaction
Add to cards
3.3 Post-move
Purchase house or hotel?
Yes
Is whole street owned?
Yes
Will this create difference > 1 on street?
No
Purchase house or hotel
Transaction
Yes
No
Perform trade
Does the current player wish to trade?
No
Yes
DO WE HAVE A DEAL?
Yes
Does the other make a counter-offer?
Make offer to other player
Yes
No
No
End


Transaction
Payer has enough liquidity?
Yes
Sell to bank
Yes
No
Property
Payee receives money
Player has enough assets?
Yes
Sell assets
No
Building
End
Assets sold to bank. Payer withdrawn
Has houses or hotels?
No
Yes
Withdraw
Will keep difference < 2?
Perform turn
2.1 Jail
Do they wish to pay the fee?
No
Do they have a get out card?
No
Is this their third turn in jail?
Yes
Is player in jail?
No
Yes
Yes
Roll dice
Did they roll a double?
Transaction
Spend card
No
Yes
No
2.2 Perform moves
Roll dice
No
Do they wish to withdraw?
Yes
Did they roll a double?
No
Yes
End turn
Make move
Make move
Withdraw

The flow chart is used to show step by step the interaction with the system. This allows us to view the control flow and see exactly how the user can step through each small section of the game. For example, rolling the dice could cause three doubles sending the player to jail. It is useful to see this for the development team to see how the functionality links together.

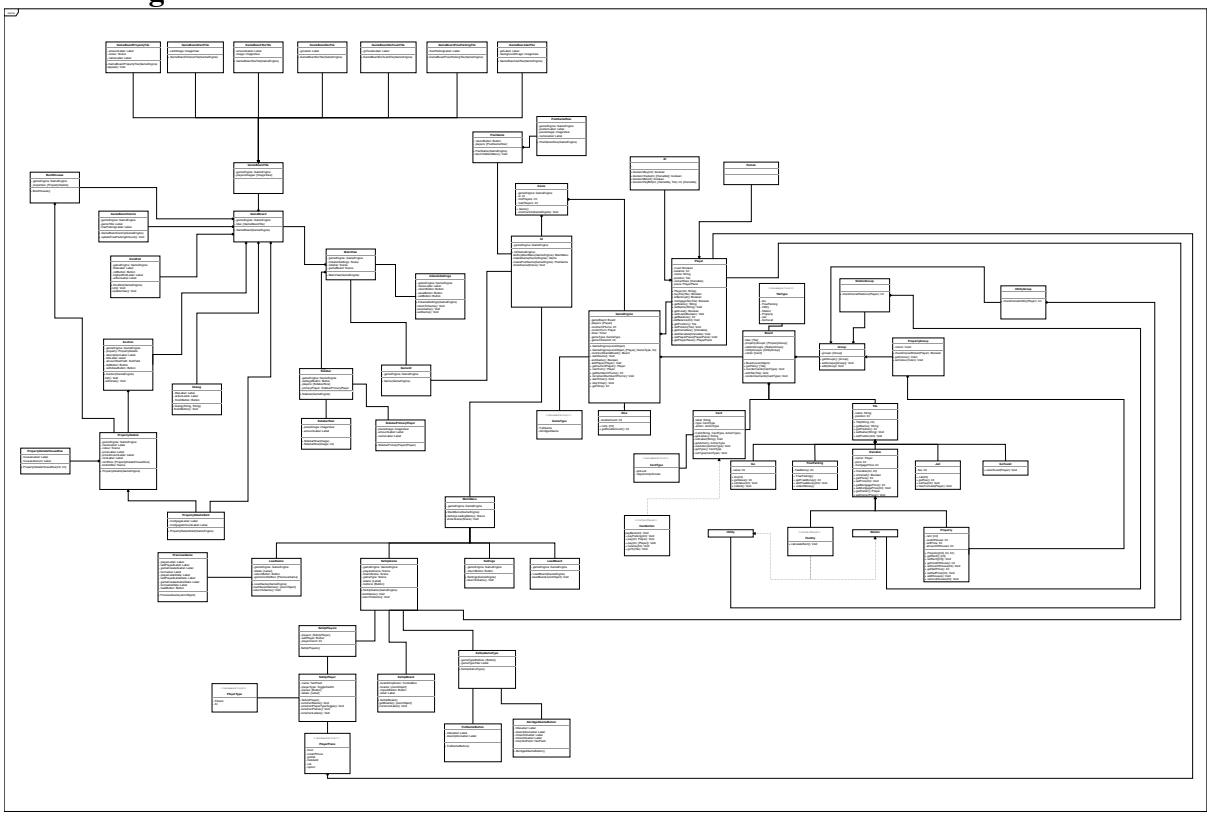
Post trading

Following the client clarifications and update to trading, the flowchart has been updated to show how this fits into the system.

Low Level

Class

Pre-trading



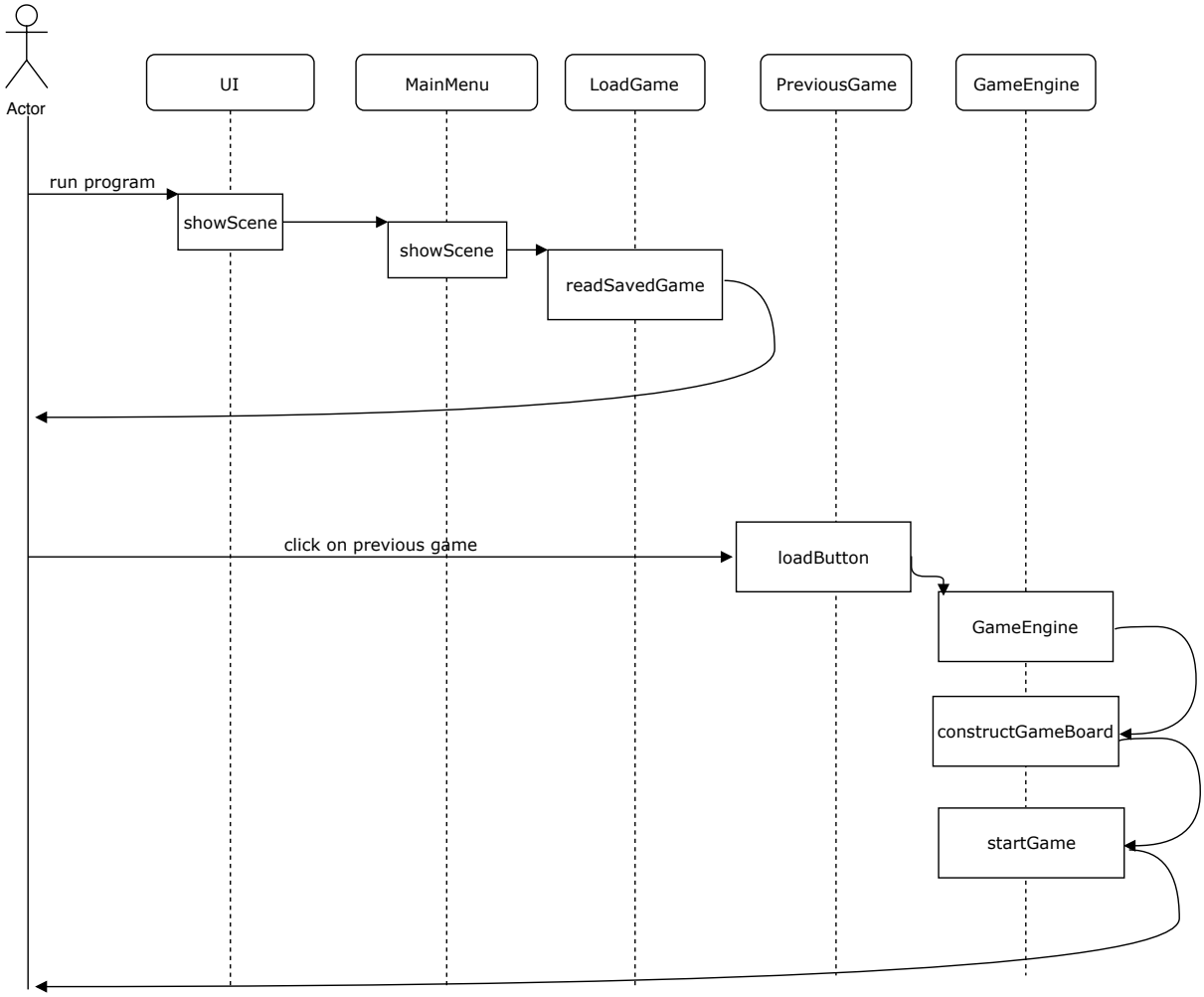
The class diagram shows how the system needs to operate programmatically. It allows us to see all of the objects that need to exist and in what sense they need to connect for the game to operate smoothly. By completing this diagram, we can see how we expect the game to link and remove any non-abstracted, poorly designed code that may slow down system operation and prohibit updating.

Post trading

Following the update to the trading, we added a trading class and also refactored the object flow design. For this, we removed any non-essential classes and cleaned up the methods by which the game engine and user interface operated.

Sequence

Pre-trading and post trading



The sequence diagram shows object interactions arranged in chronological order. This shows how objects and classes are involved in the exchanging data and what entity is responsible in the flow.

Design

User research

Summary

Of the surveyed respondents, most preferred a traditional square game board. Some felt it was important for some aspects of gameplay that they could zoom in on certain tiles. Most preferred a traditional game board format for most aspects of gameplay.

We undertook research to find out which method of displaying the Property Tycoon board was most preferred by our users. Similar games already exist on the market, and so we drew inspiration from how these were implemented before reaching our own, independent, conclusion.

Monopoly (Board)

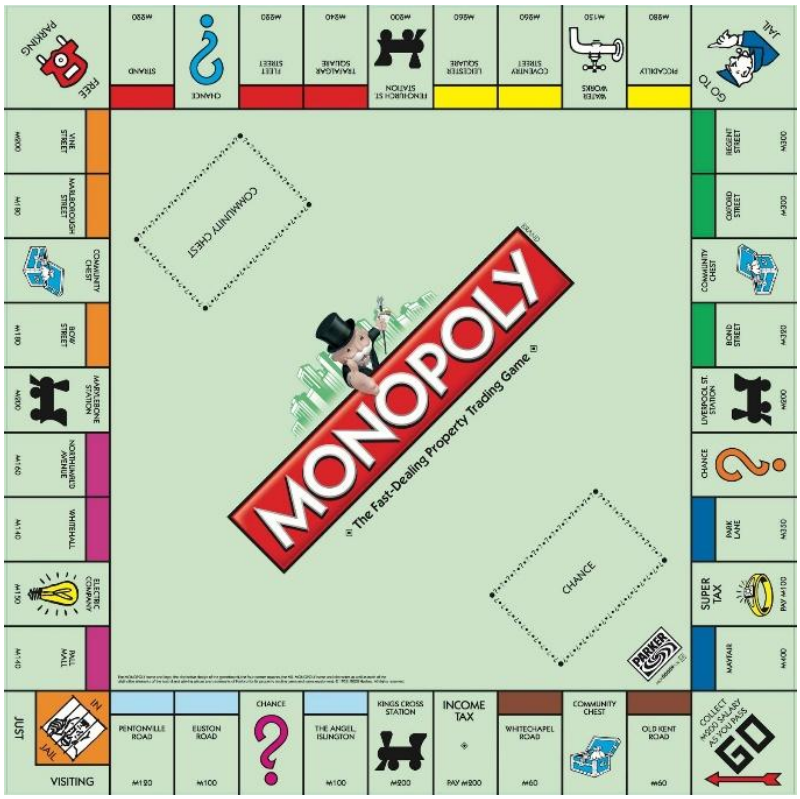


Figure 1: Traditional physical Monopoly board (Hasbro)

Hasbro produce their game Monopoly, which uses the recognisable board game structure of a square board with a circular playing track. In the post-survey interviews we conducted with our user group, concerns were raised about the readability of the top row of squares. Respondents also liked the familiarity of this design:

“I like how I can see the whole state of play at once.”
“This is simple, if it ain’t broke don’t fix it.”

Clearly, if we use this format as the inspiration for our (fundamentally different) design, the learning curve would not be steep for both new Property Tycoon players and those moving from the physical board game. Most of our respondents identified this format as being that of Hasbro's game Monopoly without being prompted in the post-survey interview, and so it may be important to consider using other options in order to differentiate our offering from Hasbro's.

Monopoly Minis (PSP)

Figure 2: Monopoly Minis game screen (Hasbro, Electronic Arts)

Monopoly Minis was published by Hasbro and Electronic Arts, released in 2008. This game, while still being technically 3D, uses a ribbon-like format, with only part of the board being visible in the main play screen. We wanted to explore the option of using a similar format, as there's no reason for the board to be 'circular' on our display – you can simply scroll left and right, which makes user interaction a lot simpler (once they are familiar with this UI).

Respondents praised the simplicity of this format:
"It's kind of nice for everything to be clear on the screen and not small."
"I think it's called a HUD, like a Heads Up Display in a shooter game, showing the money info? That's really useful I think, not something you can do in real life!"

Some were less complimentary:

"The fact this looks so 2005 makes me think they literally couldn't display the whole board – this must be from a Nintendo DS or something."
"If this was on a computer it would be silly to have it so zoomed in."

Interestingly, only one respondent noticed the option at the top of the screen to adjust the view at the player's discretion. They liked this feature, and this is something that it is worth considering adding to our implementation, if time allows.

Conclusion

In the table below, Example 1 refers to Monopoly (physical board game), and Example 2 refers to Monopoly Minis (PSP). 6 respondents were surveyed in total. The survey was conducted using an online form.

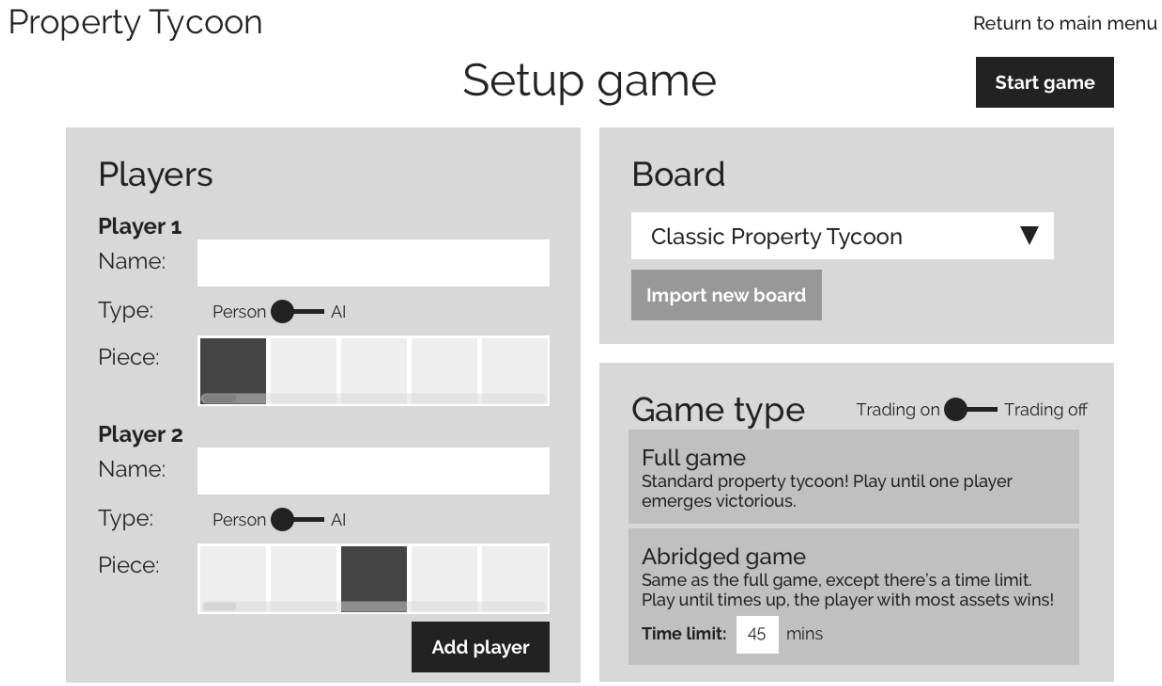
Comparing the two views of the popular board game Monopoly (Hasbro) above, which do you prefer for the following aspects of gameplay?	**Strongly prefer Example 1**	**Slightly prefer Example 1**	**Indifferent**	**Slightly prefer Example 2**	**Strongly prefer example 2**
Seeing the entire state of play	5	1	-	-	-
Viewing data about one tile		1	1	2	2
Forming a strategy for my future play	4	-	1	1	-
Overall visual appeal	2	1	2	1	-

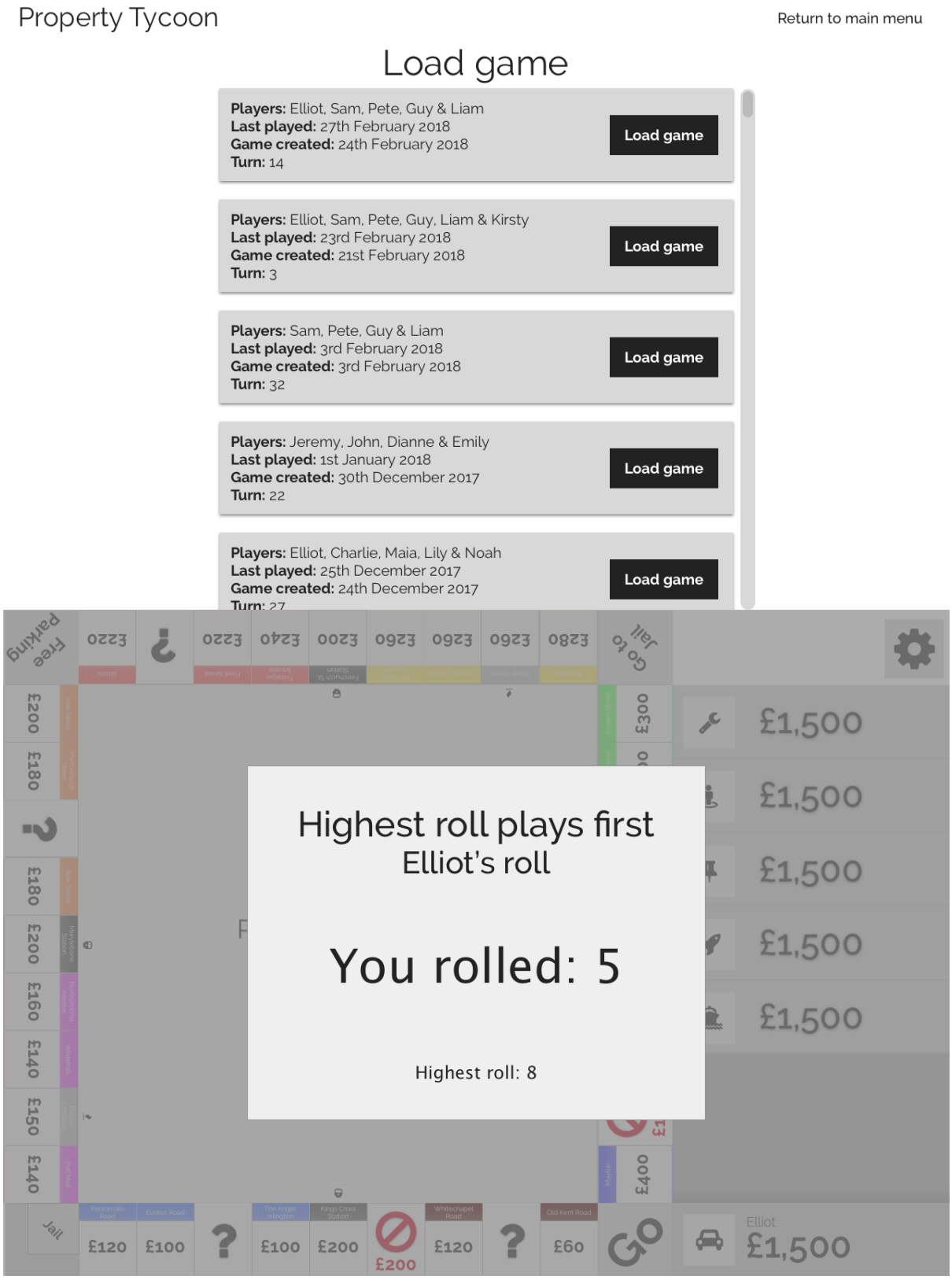
The results above show that broadly, respondents prefer Example 1 for most surveyed aspects of gameplay. However, Example 2 was preferred for viewing metadata about one single tile.

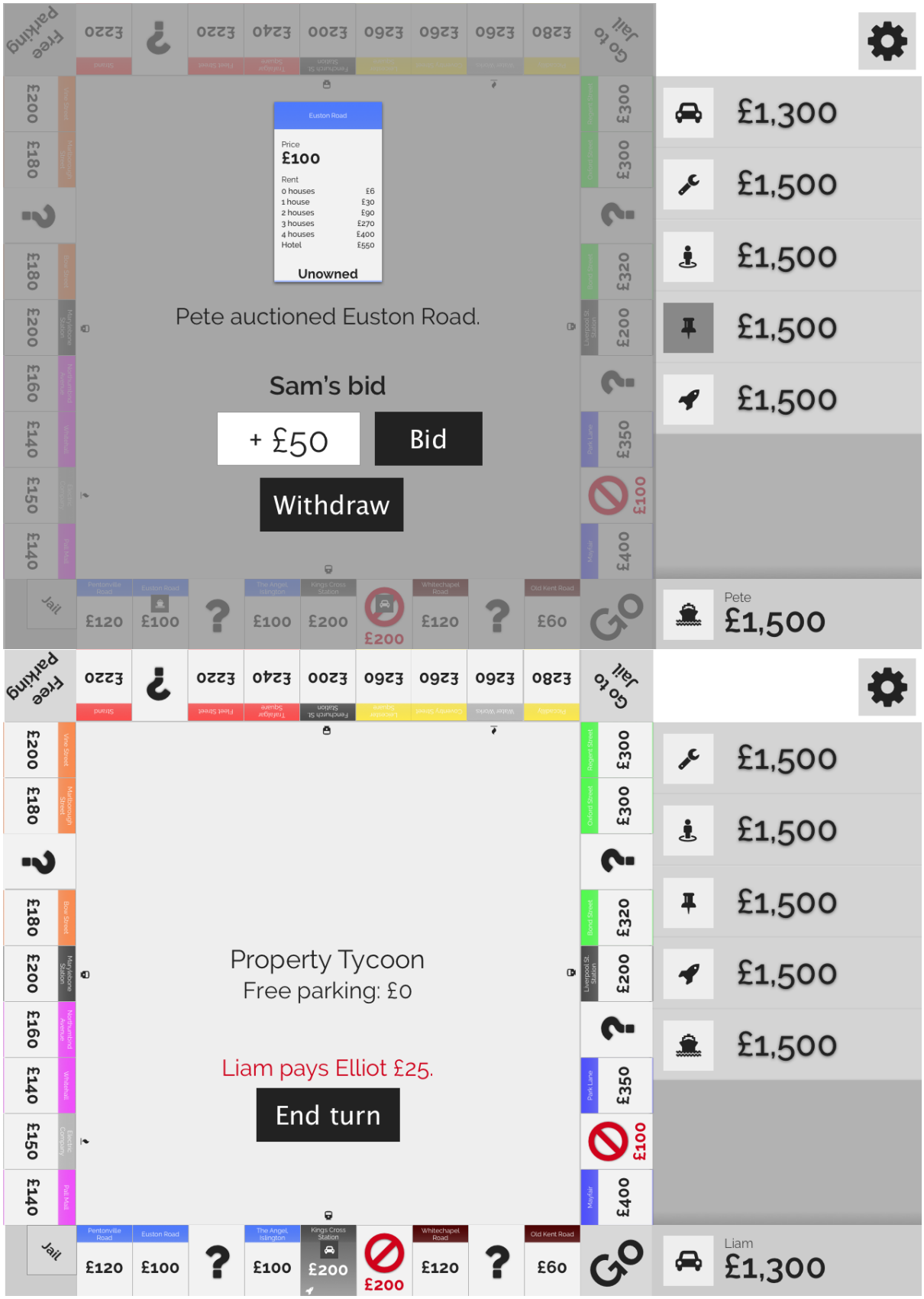
It is important for us to consider the user group's conclusions regarding a zoomed-in view. One respondent recommended that players could decide how 'close' the view was to the game board. It is clear that this functionality would provide significant qualitative benefit, if this is able to scheduled into our development time – 67% of respondents preferred a zoomed-in 'ribbon' view of the board at times when they wished to closely inspect the state of a specific tile.

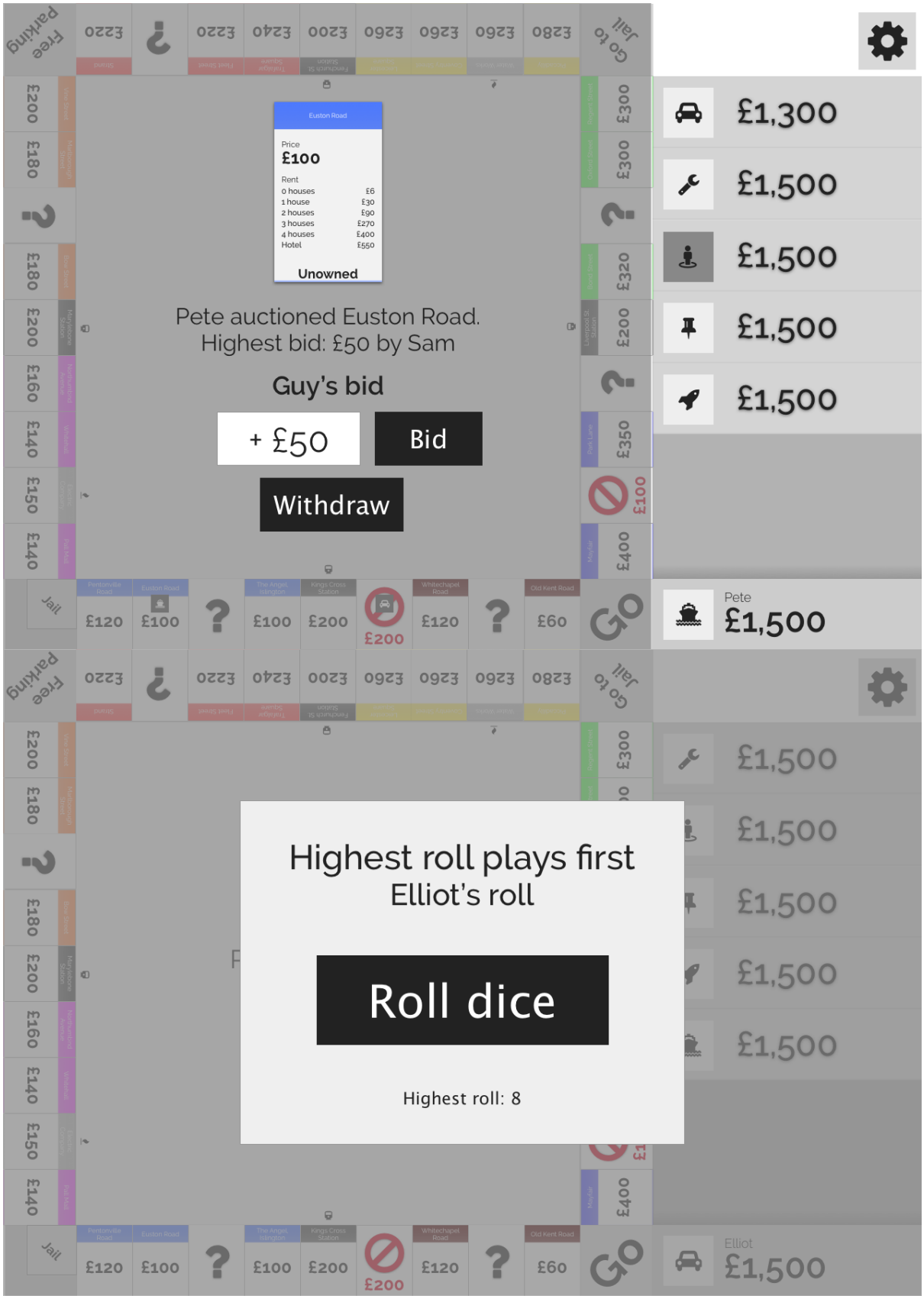
Wireframes

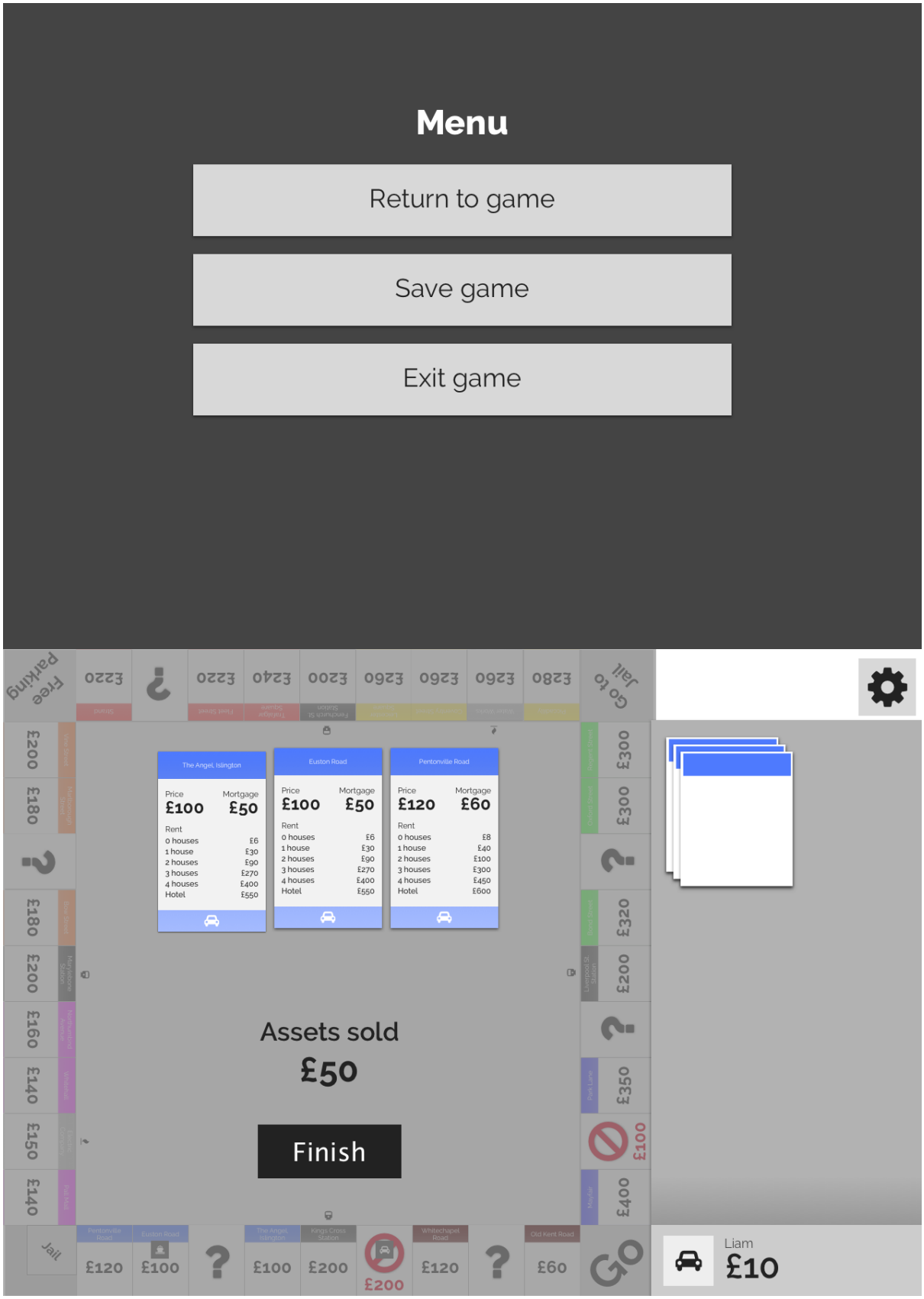

Property Tycoon
New game
Load game
Import board
Settings
Exit

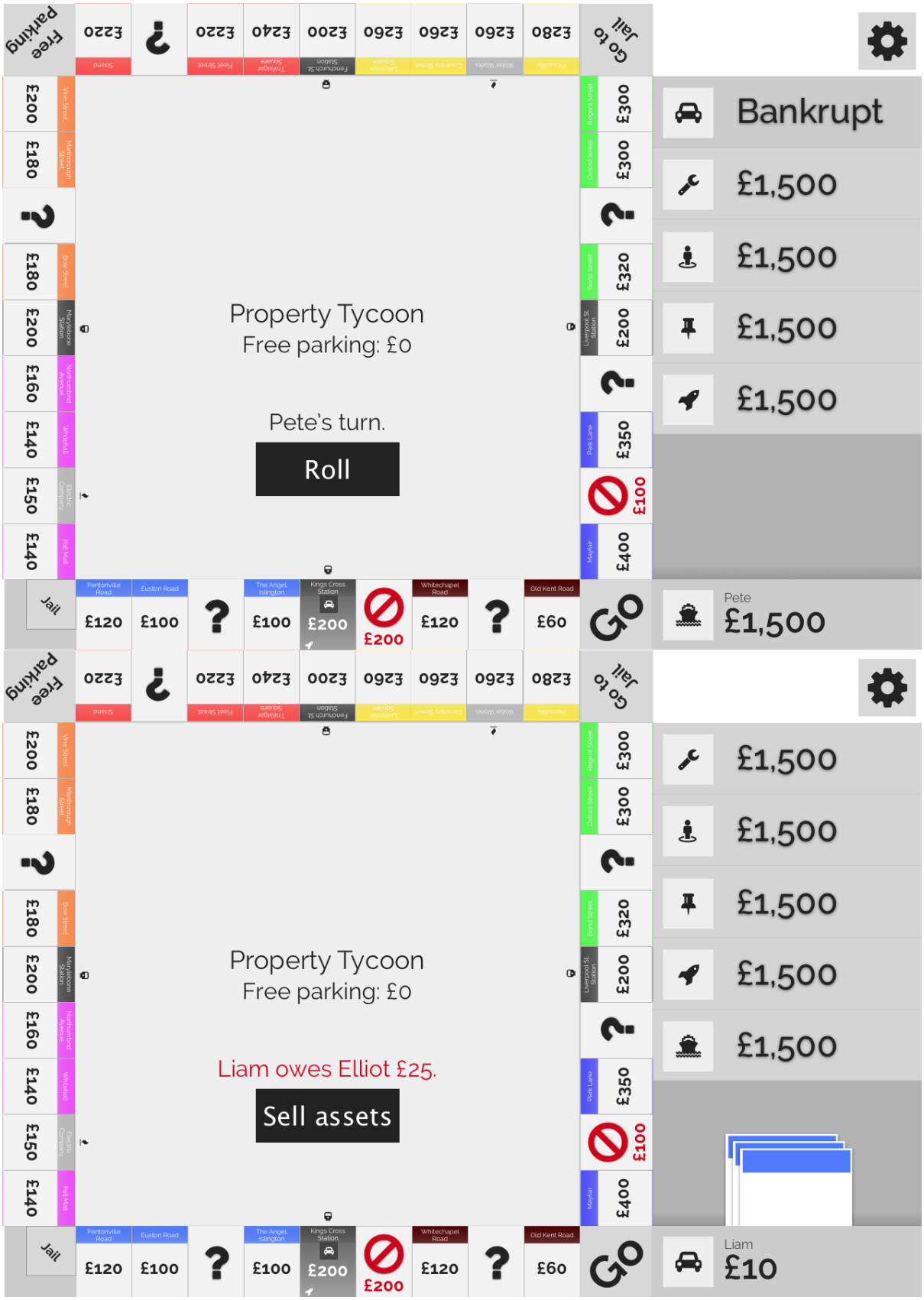

Property Tycoon
Return to main menu
Setup game
Start game
Players
Player 1
Name:
Type:
Person
AI
Piece:
Player 2
Name:
Type:
Person
AI
Piece:
Add player
Board
Classic Property Tycoon
Import new board
Game type
Trading on
Trading off
Full game
Standard property tycoon! Play until one player emerges victorious.
Abridged game
Same as the full game, except there's a time limit. Play until times up, the player with most assets wins!
Time limit:
45
mins

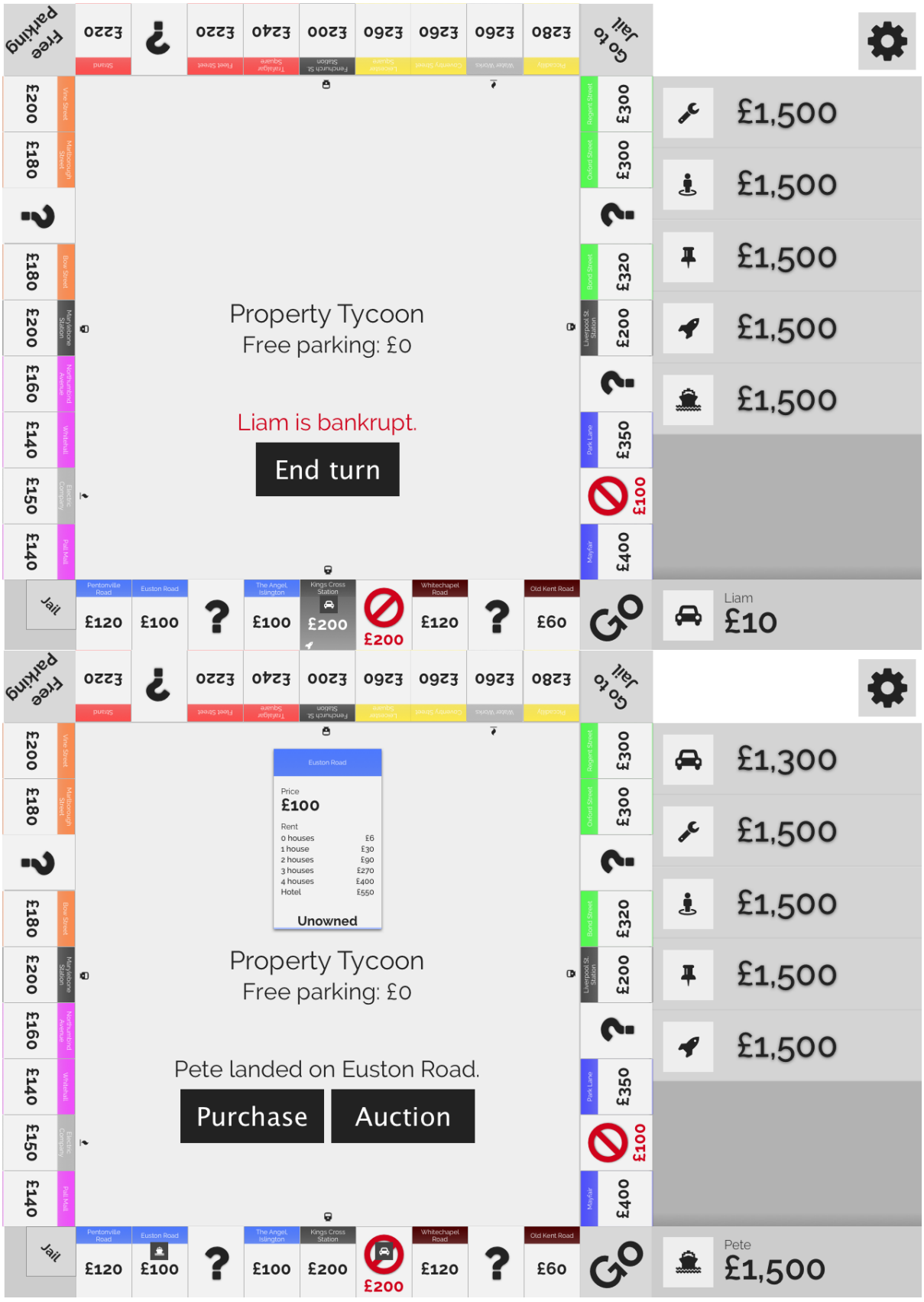

Property Tycoon
Return to main menu
Load game
Players: Elliot, Sam, Pete, Guy & Liam
Last played: 27th February 2018
Game created: 24th February 2018
Turn: 14
Load game
Players: Elliot, Sam, Pete, Guy, Liam & Kirsty
Last played: 23rd February 2018
Game created: 21st February 2018
Turn: 3
Load game
Players: Sam, Pete, Guy & Liam
Last played: 3rd February 2018
Game created: 3rd February 2018
Turn: 32
Load game
Players: Jeremy, John, Dianne & Emily
Last played: 1st January 2018
Game created: 30th December 2017
Turn: 22
Load game
Players: Elliot, Charlie, Maia, Lily & Noah
Last played: 25th December 2017
Game created: 24th December 2017
Load game
Highest roll plays first
Elliot's roll
You rolled: 5
Highest roll: 8
£1,500
£1,500
£1,500
£1,500
£1,500
Elliot
£1,500

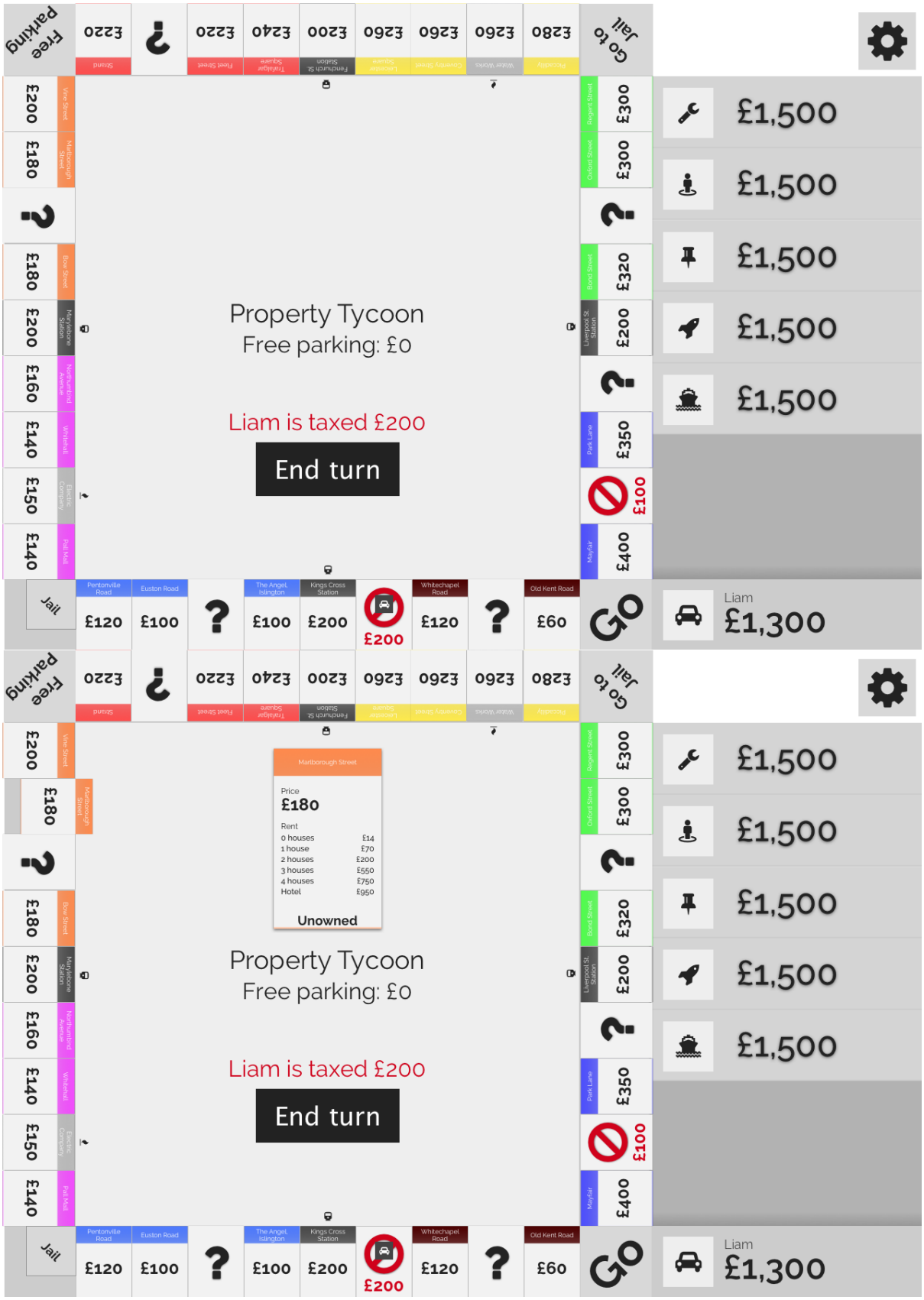

Euston Road
Price
£100
Rent
0 houses £6
1 house £30
2 houses £90
3 houses £270
4 houses £400
Hotel £550
Unowned
Pete auctioned Euston Road.
Sam's bid
+ £50
Bid
Withdraw
Pete
£1,500
£1,300
£1,500
£1,500
£1,500
£1,500
Property Tycoon
Free parking: £0
Liam pays Elliot £25.
End turn
Liam
£1,300
£1,500
£1,500
£1,500
£1,500
£1,500
Free Parking
Go to Jail
Jail
GO
£100
£200
£120
£60
£1,300
£1,500
£1,500
£1,500
£1,500
£1,500


Euston Road
Price
£100
Rent
0 houses £6
1 house £30
2 houses £90
3 houses £270
4 houses £400
Hotel £550
Unowned
Pete auctioned Euston Road.
Highest bid: £50 by Sam
Guy's bid
+ £50
Bid
Withdraw
£1,300
£1,500
£1,500
£1,500
£1,500
Pete
£1,500
Highest roll plays first
Elliot's roll
Roll dice
Highest roll: 8
Elliot
£1,500

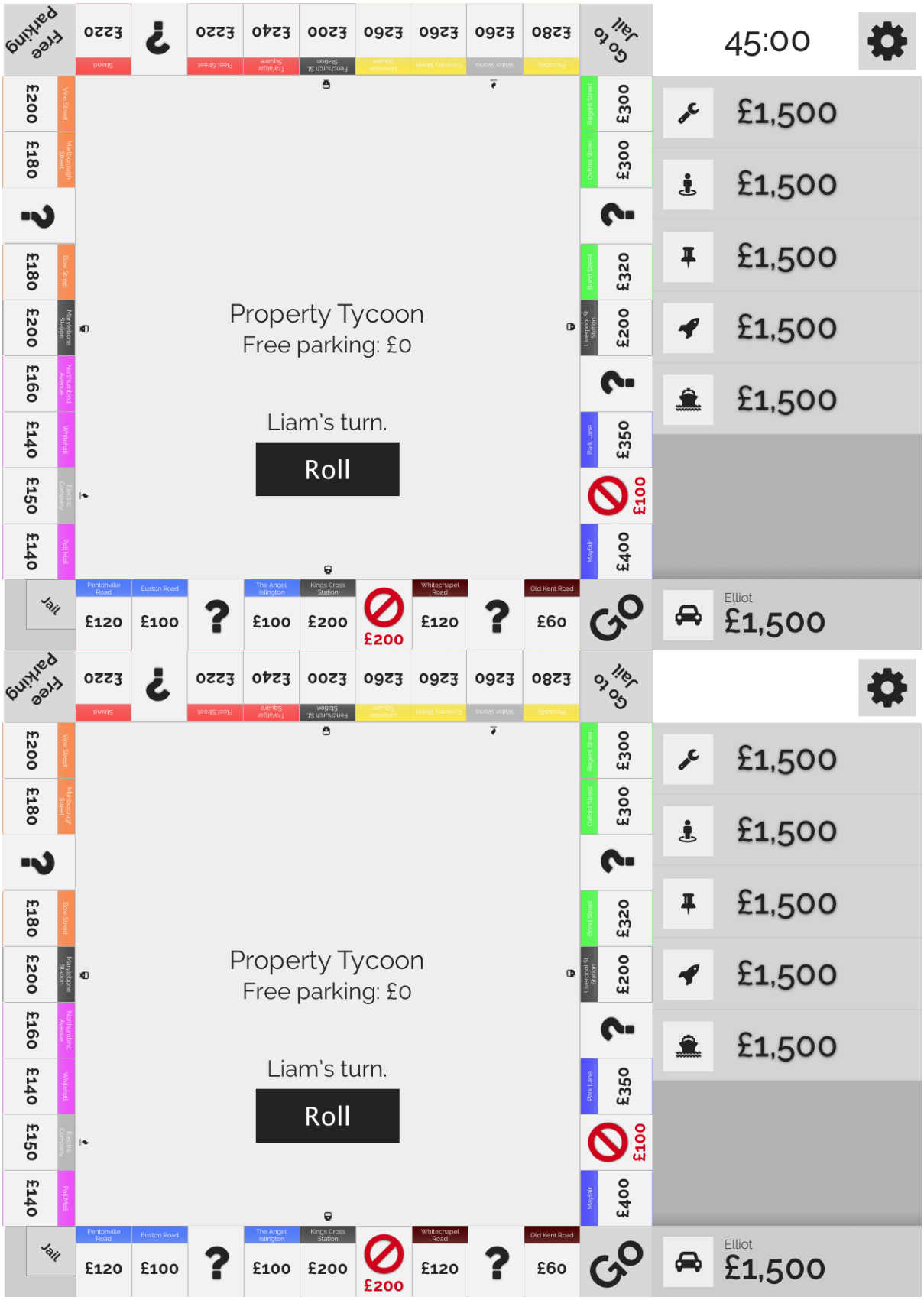

Menu
Return to game
Save game
Exit game
Free Parking
£220
£220
£240
£200
£260
£260
£260
£280
Go to Jail
£220
£180
£200
£160
£140
£150
£140
£300
£300
£320
£200
£350
£100
£400
The Angel, Islington
Price £100
Mortgage £50
Rent
0 houses £6
1 house £30
2 houses £90
3 houses £270
4 houses £400
Hotel £550
Euston Road
Price £100
Mortgage £50
Rent
0 houses £6
1 house £30
2 houses £90
3 houses £270
4 houses £400
Hotel £550
Pentonville Road
Price £120
Mortgage £60
Rent
0 houses £8
1 house £40
2 houses £100
3 houses £300
4 houses £450
Hotel £600
Assets sold
£50
Finish
Jail
Pentonville Road
£120
Euston Road
£100
The Angel, Islington
£100
Kings Cross Station
£200
£200
Whitechapel Road
£120
Old Kent Road
£60
GO
Liam
£10


Free Parking
£220
£220
£240
£200
£260
£260
£280
Go to Jail
£180
£200
£160
£140
£150
£140
£300
£300
£320
£200
£350
£100
£400
Property Tycoon
Free parking: £0
Pete's turn.
Roll
Bankrupt
£1,500
£1,500
£1,500
£1,500
Jail
£120
£100
£100
£200
£200
£120
£60
GO
Pete
£1,500
Property Tycoon
Free parking: £0
Liam owes Elliot £25.
Sell assets
£1,500
£1,500
£1,500
£1,500
£1,500
Liam
£10

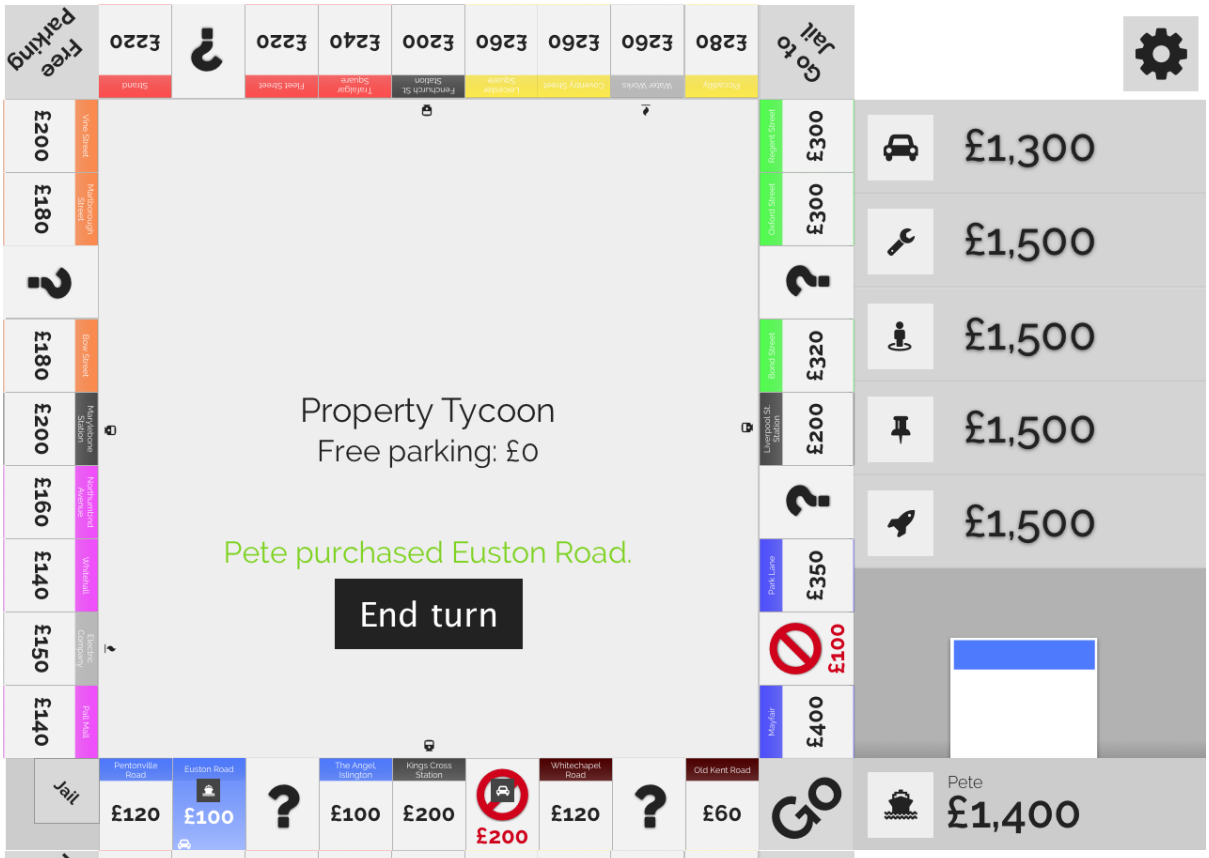

Free Parking
£220
£220
£240
£200
£260
£260
£280
Go to Jail
£180
£200
£160
£140
£150
£140
£300
£300
£320
£200
£350
£100
£400
Property Tycoon
Free parking: £0
Liam is bankrupt.
End turn
Jail
£120
£100
£100
£200
£200
£120
£60
GO
£1,500
£1,500
£1,500
£1,500
£1,500
Liam
£10
Euston Road
Price
£100
Rent
0 houses £6
1 house £30
2 houses £90
3 houses £270
4 houses £400
Hotel £550
Unowned
Pete landed on Euston Road.
Purchase
Auction
£1,300
£1,500
£1,500
£1,500
£1,500
Pete
£1,500

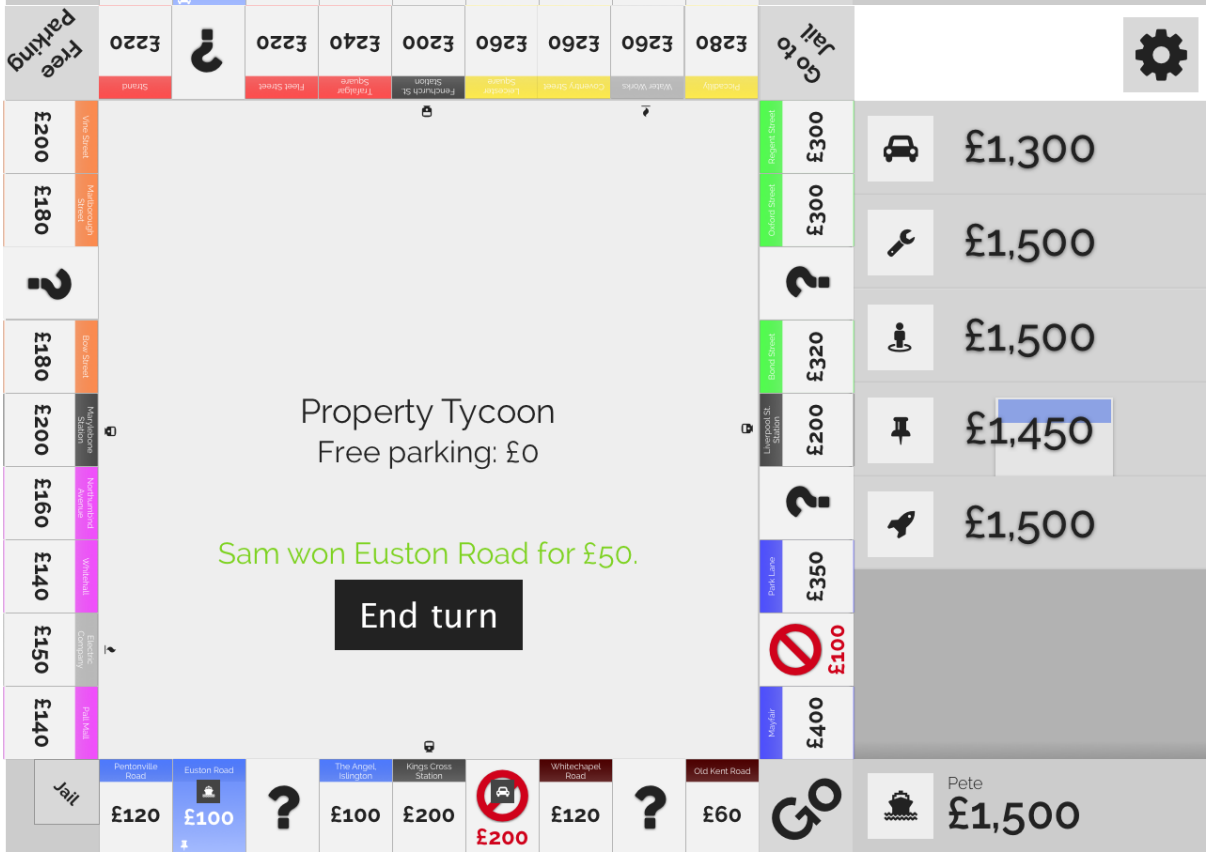

Free Parking
£220
£220
£240
£200
£260
£260
£280
Go to Jail
£180
£200
£180
£160
£140
£150
£140
£200
£300
£300
£320
£200
£350
£100
£400
Property Tycoon
Free parking: £0
Liam is taxed £200
End turn
£1,500
£1,500
£1,500
£1,500
£1,500
Jail
£120
£100
£100
£200
£200
£120
£60
GO
Liam
£1,300
Marlborough Street
Price
£180
Rent
0 houses £14
1 house £70
2 houses £200
3 houses £550
4 houses £750
Hotel £950
Unowned

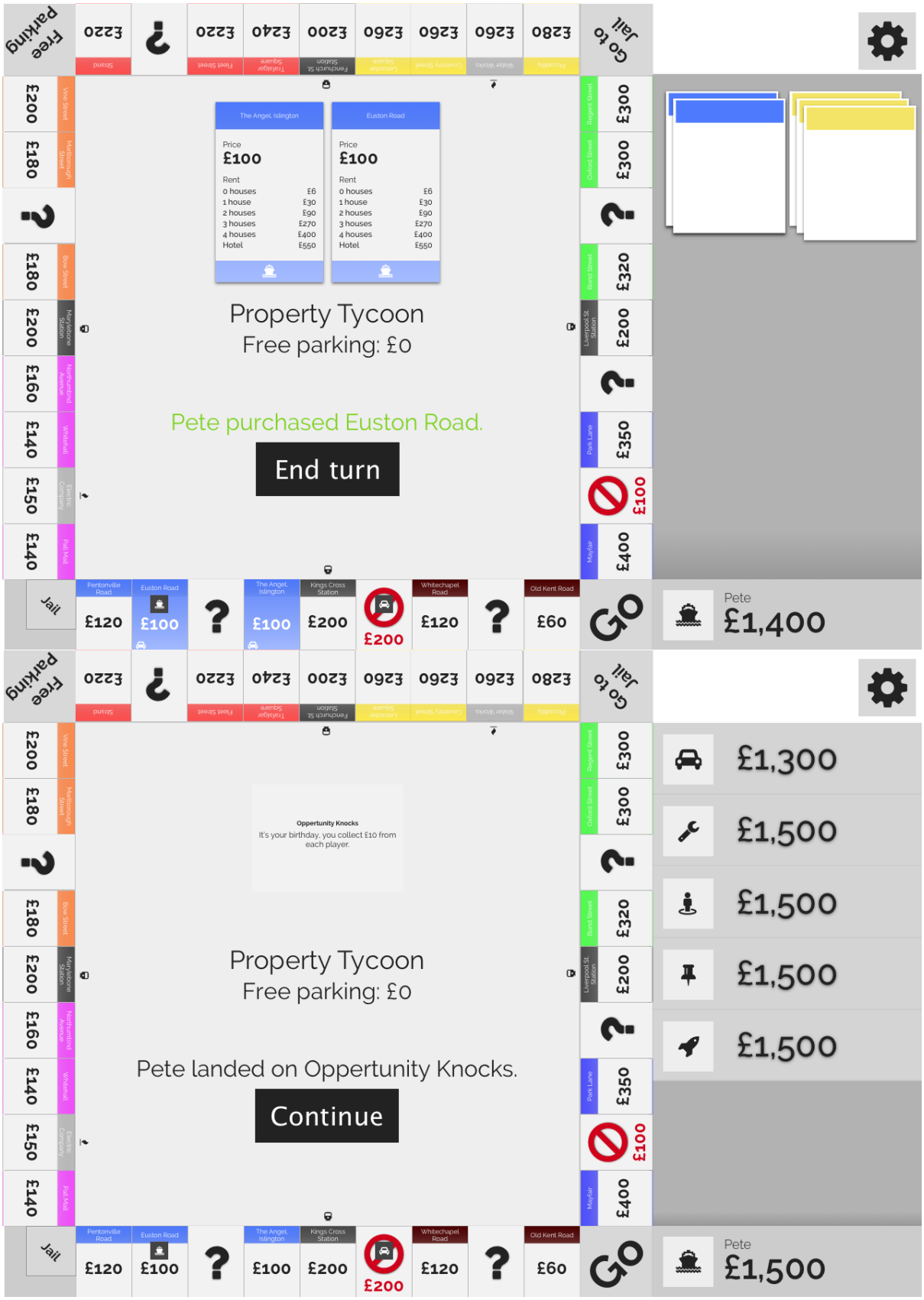

Property Tycoon
Free parking: £0
Liam rolled a 4!
£1,500
£1,500
£1,500
£1,500
£1,500
Liam
£1,500
The Angel, Islington
Price
£100
Rent
0 houses £6
1 house £30
2 houses £90
3 houses £270
4 houses £400
Hotel £550
Euston Road
Price
£100
Rent
0 houses £6
1 house £30
2 houses £90
3 houses £270
4 houses £400
Hotel £550
Pentonville Road
Price
£100
Rent
0 houses £6
1 house £30
2 houses £90
3 houses £270
4 houses £400
Hotel £550
Development costs
£150
Finish
Pete
£1,180



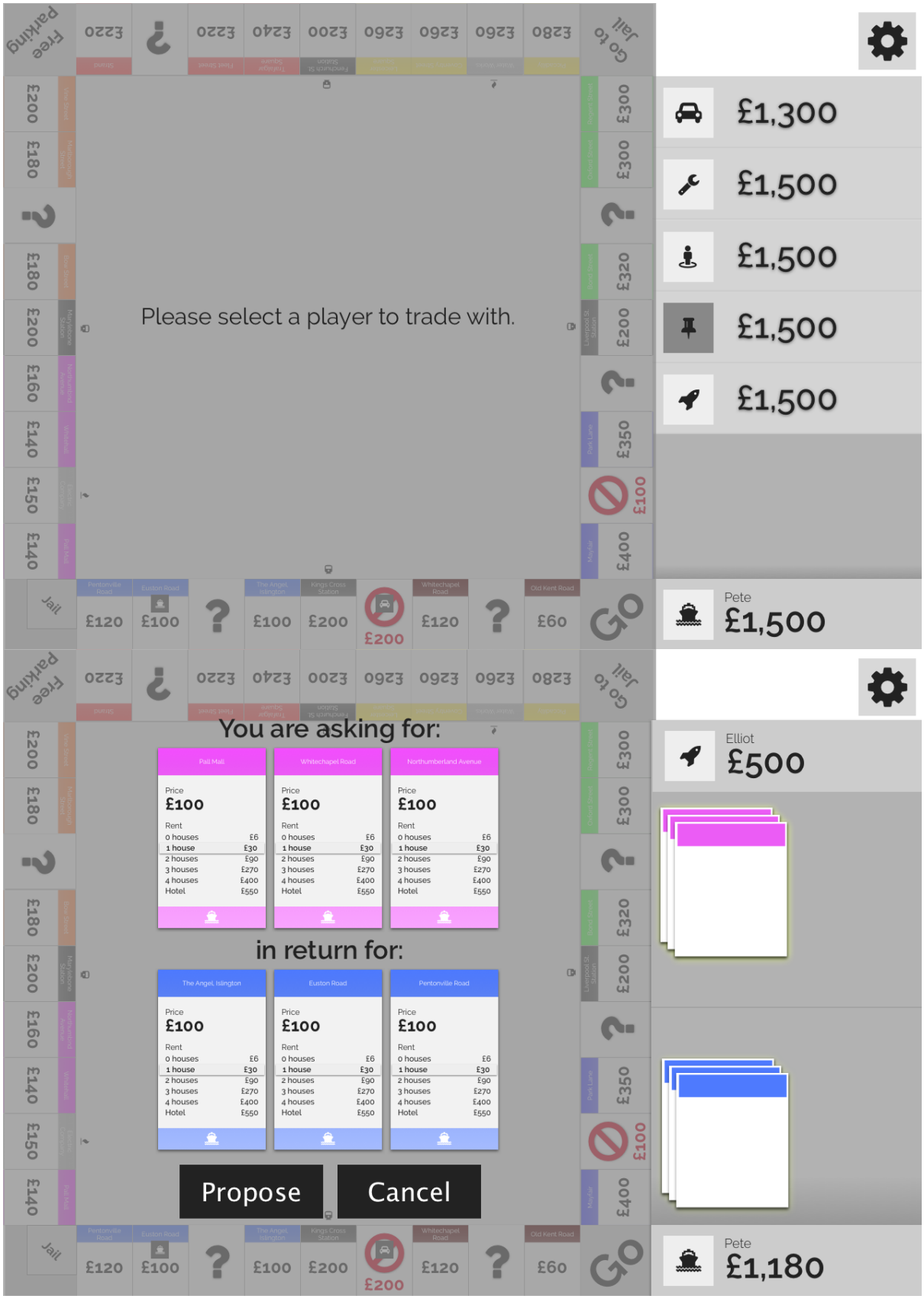
Free Parking
£220
£220
£240
£200
£260
£260
£280
Go to Jail
£180
£200
£160
£140
£150
£140
£300
£300
£320
£200
£350
£100
£400
Property Tycoon
Free parking: £0
Liam's turn.
Roll
£1,500
£1,500
£1,500
£1,500
£1,500
Jail
£120
£100
£100
£200
£200
£120
£60
GO
Elliot
£1,500

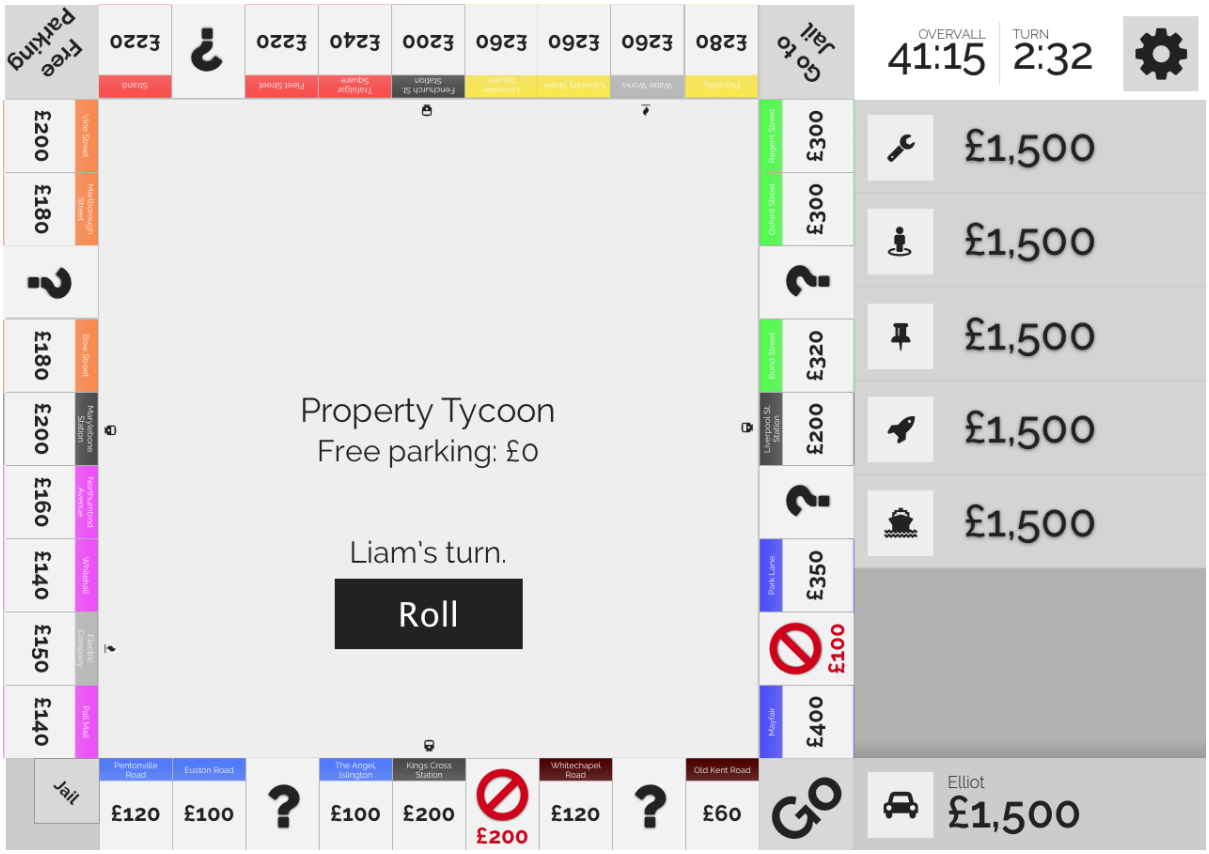

Free Parking
£220
£220
£240
£200
£260
£260
£280
Go to Jail
£180
£200
£160
£140
£150
£140
£180
£200
£320
£200
£350
£100
£400
Property Tycoon
Free parking: £0
Pete purchased Euston Road.
End turn
£1,300
£1,500
£1,500
£1,500
£1,500
Jail
£120
£100
£100
£200
£200
£120
£60
GO
Pete
£1,400


Free Parking
£220
£220
£240
£200
£260
£260
£280
Go to Jail
£180
£200
£160
£140
£150
£140
£180
£200
£320
£200
£350
£100
£400
Property Tycoon
Free parking: £0
Sam won Euston Road for £50.
End turn
£1,300
£1,500
£1,500
£1,450
£1,500
Jail
£120
£100
£100
£200
£200
£120
£60
GO
Pete
£1,500

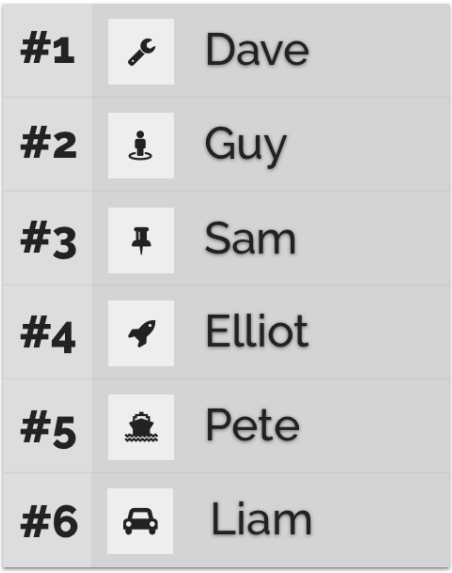

Free Parking
£220
£220
£240
£200
£260
£260
£280
Go to Jail
The Angel, Islington
Euston Road
Price
£100
Rent
0 houses £6
1 house £30
2 houses £90
3 houses £270
4 houses £400
Hotel £550
Property Tycoon
Free parking: £0
Pete purchased Euston Road.
End turn
£100
Pete
£1,400
Jail
Pentonville Road
£120
Euston Road
£100
The Angel, Islington
£100
Kings Cross Station
£200
Whitechapel Road
£120
Old Kent Road
£60
GO
£300
£300
£320
£200
£350
£400
Oppertunity Knocks
It's your birthday, you collect £10 from each player.
Property Tycoon
Free parking: £0
Pete landed on Oppertunity Knocks.
Continue
£1,300
£1,500
£1,500
£1,500
£1,500
Pete
£1,500


Property Tycoon
Free parking: £0
Pete purchased Euston Road.
Develop
Trade
End turn
£1,300
£1,500
£1,500
£1,500
£1,500
Pete
£1,180
Pete is asking for:
Price £100
Rent
0 houses £6
1 house £30
2 houses £90
3 houses £270
4 houses £400
Hotel £550
Pall Mall
Whitechapel Road
Northumberland Avenue
in return for:
The Angel, Islington
Euston Road
Pentonville Road
Accept
Cancel
Elliot
£500
Pete
£1,180


Please select a player to trade with.
£1,300
£1,500
£1,500
£1,500
£1,500
Pete
£1,500
You are asking for:
Pall Mall
Whitechapel Road
Northumberland Avenue
Price
£100
Rent
0 houses £6
1 house £30
2 houses £90
3 houses £270
4 houses £400
Hotel £550
in return for:
The Angel, Islington
Euston Road
Pentonville Road
Propose
Cancel
Elliot
£500
Pete
£1,180



Return to main menu



Number of turns: 23

Time taken: 1hour 23 minutes

Share

Main menu prototype

https://projects.invisionapp.com/share/JGG5FVZ5SR6#/screens/282793567

Game Board prototype

https://projects.invisionapp.com/share/FUG5G9RV8RM#/screens

Results screen prototype

https://projects.invisionapp.com/share/4JG5G9YUB3W#/screens

Wireframes explanation video

https://bit.ly/2rgAR7K

InVision was used to prototype our wireframes together. Three different prototypes were created 'Main Menu', 'Game Board' and 'Results screen'. These were used to make sure the wireframes included all elements possible to play the game. To access these wireframe prototypes the links are included above. An explanation video is also included above to show the thought process of how each element came to be.

Time was taken to ensure that the wireframes fully mapped out all game functionality and acted as a way to unify high-level designs. While not officially required in the specification, the wireframes acted as a more visual reference for the team and help realise any remaining kinks in the requirements.
Graphics

The graphics for the game were produced to closely resemble the graphics from the original game. This was to keep within the requirements and is one example of how we kept to our domain requirement "The game should be fun to play and have a colourful and intuitive interface that reflects the spirit and character of the original board game." There is also two of each graphic - the designs for the board required colour and silhouetted variants.

Implementation & Testing

Continuous Integration

In order to find the most effective service for hosting and testing our product, we looked into different providers of Continuous Integration (CI) tools. Below is the research we conducted, which showed us that the best functionality available in a free tier was Travis CI. The reason for this is the wide variety of support for a large-scale deployment, in terms of updating the product and incorporating new features.

Pros and Cons of different continuous integrations

	Travis CI	**Circle CI**	**Jenkins**
Price	Monthly fee (Free for Students)	Free	Free
Hosted	Cloud based	Cloud based	You need a dedicated server
Supported Languages	Android, C, C#, C++, Clojure, Crystal, D, Dart, Erlang, Elixir, F#, Go, Groovy, Haskell, Haxe, Java, JavaScript (with Node.js), Julia, Objective-C, Perl, Perl6, PHP, Python, R, Ruby, Rust, Scala, Smalltalk, Visual Basic	Go (Golang), Haskell, Java, PHP, Python, Ruby/Rails, Scala	Android, C, C#, C++, Clojure, Crystal, D, Dart, Erlang, Elixir, F#, Go, Groovy, Haskell, Haxe, Java, JavaScript
Customization	Median	Low	High
Processing Speed	Fast	Median	Depends on the dedicated server
Docker Support	Yes	Yes	Yes
Config Type	YML	YML	API

To set up our continuous integration, we wrote a YML config file. Within this the following states were set:

Option	**Set Value**
Language	Java
JDK	oraclejdk8
Environment Options	_JAVA_OPTIONS="-Djava.awt.headless=true -Dtestfx.robot=glass -Dtestfx.headless=true -Dprism.order=sw -Dprism.text=t2k -Dtestfx.setup.timeout=2500"
JDK package installer	oracle-java8-installer
Script	Gradlew check
Cache	Gradle wrapper and cache

This configuration file allows the Travis server to run the Java application (using JDK8) and gradle script. The UI tests are then ran using headless mode so that the server can reproduce them without fail. The reason this is so key to testing is that all future models can easily be tested in this way and it shows the full game working, removing a large part of the time-consuming testing phase.

Landing / documentation website

Incorporating Docusaurus (docusaurus.io), we were able to produce a website for the game for free with very little time. Docusaurus is a framework that uses React (Facebook's JavaScript framework) in order to produce a clear, well-built website. React is used to generate the web pages, including the full write-up of the project's documentation. This site is hosted using Netlify, which allows us to host the site in the game's private repo and produce the website from a specific branch.

Netlify allows us to install node.js and run the build file on the server side. This means that each time we push to the repo, a new version is built automatically. We use this command to achieve this:

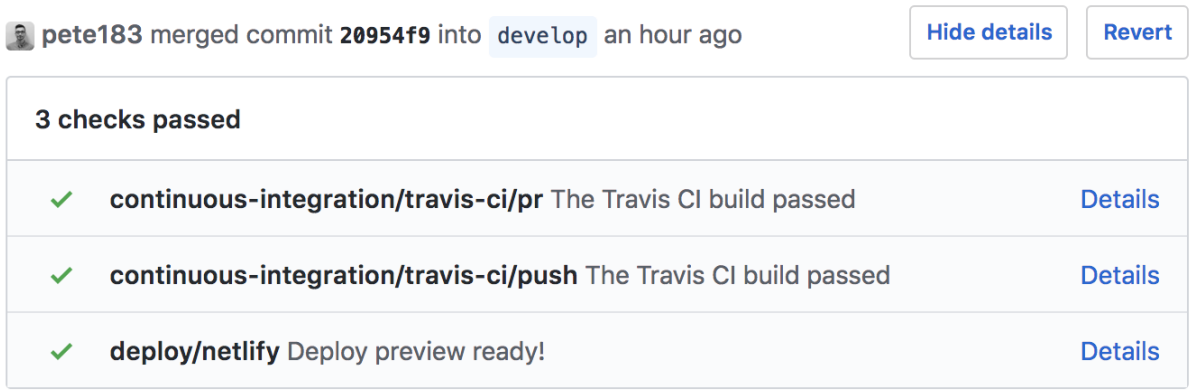
npm run deploy --prefix ./website/

A key benefit that comes alongside the ability to build from the repo is that during a pull request to the branch that Netlify is linked to, the website is built and can be viewed before the request is approved. Further to this, it enables us to use an SSL certificate issued by Let's Encrypt.

GitHub

We used GitHub to store the repo so that all team members had constant up-to-date access to the source. In order to ensure that no errors were pushed into the develop or master branch, we protected these branches. Protected branches disable force pushing, prevent branches from being deleted, and optionally require status checks before merging (https://help.github.com/articles/defining-the-mergeability-of-pull-requests/). This means that in our case, we could prohibit any merge into the develop branch without all tests passing and the Travis CI and the Netlify builds succeeding. Additionally, least one other member of the team must authorise the pull request.

Travis CI allows us to see if all tests on the push to the branch are working as expected, and also that the tests are all running as expected should the branches be merged. Netlify simultaneously deploys its own tests that check the build of the website, alongside making the build given the pull request a website that can be viewed before merging.



Game engine

The game engine was created to control all back-end functionality of the game. In principle, it controls the full data flow of the game in the sense that it controls the values of all variables related to any displayed data on the board and is responsible for the location of players and any other interaction with the user interface.

JSON PHP

Where converting the game data (supplied as an Excel file), to JSON manually would have been too time-consuming and inefficient, we used simple PHP scripts to reformat the data. Converting the Excel file as a CSV to JSON is relatively simple but manipulating the rows of the Cards JSON datatable to suit our requirements was more complex. The reason we chose PHP for this task is twofold: PHP has a robust built-in JSON parser, and as a language is extremely easy to deploy. For completeness, this PHP script is included in the resources directory.

UI

The user interface controls all graphical content displayed on the screen. All values and functionality are controlled by the game engine however any content shown on the screen is the responsibility of the UI. To do this, we used JavaFX to link the graphics and map them to the correct position on the screen.

Unit testing - JUnit testing

In order to unit test we utilised the JUnit package. By using this we could test every class produced for the game, by simply writing JUnit tests that build each class and test each method respectively. To do this we assert something has not yet happened, then call a method to apply the change, then assert the change has happened.

UI testing and System testing

When we began UI development, we knew that we wanted to extend the concept of test-driven development to the UI as well. We also saw this as an opportunity to help automate system testing, as the UI test suite would be asserting that, for example, click a specific button would yield a certain result.

We found a library for UI testing using JUnit and JavaFX called TestFX. We chose this library as it seems well maintained and has basic documentation, compared to other alternative libraries. TestFX also offers the ability to run UI tests 'headlessly' which means that the tests can run behind the scenes, allowing them to be run on Travis.

TestFX allowed us to define tests in a similar way to regular JUnit tests. The difference is that with TestFX tests you can use matchers to query the scene for UI elements. This allowed us to locate buttons, which TestFX then gives methods to move the mouse to and click on. We also had access to the GameEngine class from the UI tests so that we could ensure after button clicks that various properties of the game logic were properly affected.

Testing

In this project we have been concerned with 2 major types of testing: Unit Testing and System Testing.

In order to complete this, we implemented test driven development, for which we wrote JUnit tests for each method before the method was written, after which the test was expected to pass; and the method would be amended to work before moving on.

For system testing, we implemented TestFX, which will test the user interface of the game; by doing this the response the user interface has is monitored, hence the system is tested as the resulting movement in the UI will only be correct if the user interface and game engine are both working as expected. Thus, this is a very effective way of black box testing, as the full system is tested upon every build.

Below are the tests conducted on the game engine:

Artificial Intelligence Class Testing

Ref	Description	Input(s)	Expected output(s)	Actual output(s)	Pass/Fail	Action?
1	Can the AI buy a tile when they have enough money	A game board	The AI player bought the tile	The AI player bought the tile	Pass	None
2	Can the AI buy a tile when they do not have enough money	A game board	The AI player did not buy the tile	The AI player did not buy the tile	Pass	None
3	Can the AI buy Houses for their property if they have enough money	A game board	The AI player bought houses	The AI player bought houses	Pass	None
4	Can the AI buy Houses for their property if they do not have enough money	A game board	The AI player did not buy houses	The AI player did not buy houses	Pass	None

5	Will the AI trade if the deal is in their favour	A game board	The AI performs the trade	The AI performs the trade	Pass	None
6	Will the AI trade if the deal is not their favour	A game board	The AI does not perform the trade	The AI does not perform the trade	Pass	None
7	Does the AI bid on a property when they have enough money	A game board	The AI bids on the property	The AI bids on the property	Pass	None
8	Does the AI bid on a property if they do not have enough money	A game board	The AI does not bid on the property	The AI does not bid on the property	Pass	None
9	Does the AI pay a bill when it has enough money	A game board	The AI Pays the bill	The AI Pays the bill	Pass	None
10	Does the AI pay a bill when it does not have enough money	A game board	The AI sells property to affords the bill	The AI sells property to affords the bill	Pass	None

Board Class Testing

Ref	Description	Input(s)	Expected output(s)	Actual output(s)	Pass/Fail	Action?
1	Does the board read in JSON data correctly?	JSON data file	There will be 40 tiles read into the board	There are 40 tiles read into the board	Pass	None
2	Does is there an error if the JSON data is in the wrong format?	JSON data file	The data is not read in and an error is produced	The data is not read in and an error is produced	Pass	None
3	Are the tiles correctly read into the tile variable	JSON data file	The tile variable returns the tiles entered into it	The tile variable returns the tiles entered into it	Pass	None
4	Can tiles be added to the board correctly?	JSON data file	The tile has been entered into the board	The tile has been entered into the board	Pass	None

Card Action Class Testing

Ref	Description	Input(s)	Expected output(s)	Actual output(s)	Pass/Fail	Action?
1	Can the card action type be returned	A card, a card action, a board and a player	The card action is returned	The card action is returned	Pass	None
2	Can the origin be found	A card, a card action, a board and a player	The card action is set	The card action is set	Pass	None
3	Can the origin be set	A card, a card action, a board and a player	The origin is set	The origin is set	Pass	None
4	Can the intent be found	A card, a card action, a board and a player	The intent of the card is returned	The intent of the card is returned	Pass	None
5	Can the intent be set	A card, a card action, a board and a player	The intent of the card is set	The intent of the card is set	Pass	None
6	Can the action be performed	A card, a card action, a board and a player	The card action is performed correctly	The card action is performed correctly	Pass	None

Card Stack Test

Ref	Description	Input(s)	Expected output(s)	Actual output(s)	Pass/Fail	Action?
1	Can the card type be set	A card stack object	A new card type can be returned	The new card type has been returned	Pass	None
2	Can the card type be returned	A card stack object	A card type can be returned	A card type can be returned	Pass	None
3	Can the cards be shuffled	A card stack object	The cards will be in a different order to the original order	The cards were in a new order	Pass	None

Card Tile Class Test

Ref	Description	Input(s)	Expected output(s)	Actual output(s)	Pass/Fail	Action?
1	Can the card stack be set	A card tile	A new card tile has been set	The new card tile was set	Pass	None
2	Can the card stack be returned	A card tile	A new card tile is returned	The new card stack was returned	Pass	None

Dice Class Test

Ref	Description	Input(s)	Expected output(s)	Actual output(s)	Pass/Fail	Action?
1	Can the dice be rolled	A dice object	Two numbers between 1 and 6	Two numbers between 1 and 6	Pass	None
2	The double count be returned	A dice object	The amount of doubles rolled on the current turn	The amount of doubles rolled on the current turn I returned	Pass	None

Free Parking Class Test

Ref	Description	Input(s)	Expected output(s)	Actual output(s)	Pass/Fail	Action?
1	Can the money on free parking be collected	A free parking tile	The money stored on free parking is collected	The money on free parking was collected	Pass	None

Game Engine Class Test

Ref	Description	Input(s)	Expected output(s)	Actual output(s)	Pass/Fail	Action?
1	A game can be constructed from JSON data	A JSON file	The game is initialised from the JSON data	The game is initialised from the JSON data	Pass	None
2	Can the current players in the game be returned	A current game	All of the current players are returned	All of the current players are returned	Pass	None
3	Is the turn changed when each player finishes their turn	A current game	The turn is changed to the next player each time the turn is over	The turn was changed each time a player ended their turn	Pass	None
4	Can the number of turns played be returned	A current game	The amount of turns played	The amount of turns played	Pass	None
5	Can the current amount of time spent on the game be returned	A current game	The time the game has gone on for	The time that the game has gone on for	Pass	None
6	Can the game be saved	A current game	The game is saved into JSON data	The game is saved into JSON data	Pass	None

Go To Jail Class Test

Ref	Description	Input(s)	Expected output(s)	Actual output(s)	Pass/Fail	Action?
1	Can a player be sent to jail	A game board and a player	The player is in jail	The player is in jail	Pass	None

Group Class Test

Ref	Description	Input(s)	Expected output(s)	Actual output(s)	Pass/Fail	Action?
1	Can the owners of the groups be returned	A player and a group	The player owners of the tiles in a group	The player owners of the tiles in a group	Pass	None
2	Is it possible to check if an owner owns a full street	A player and a group	Return true if player owns the full group	Returns true if player owns group, false if otherwise.	Pass	None
3	Find how many properties in a group are owned by a player	A player and a group	The number of tiles owned in the same group by the player	The number of tiles owned in the same group by the player	Pass	None

Income Tile Class Test

Ref	Description	Input(s)	Expected output(s)	Actual output(s)	Pass/Fail	Action?
1	Can a new value be set?	The board, income tile and player along with the new value received for landing on the tile	The value will be set as the new rate.	The value was set at the new rate.	Pass	None
2	Can a value be returned?	The board, income tile and player	The value will be returned	The value was returned	Pass	None
3	Can a value be added to the income tile	The board, income tile and player and a value to add to the value of the tile	The new value of the tile	The new value of the tile	Pass	None
4	Can money be collected from the tile	The board, income tile and player	The value on the tile will be set to 0 and the player balance will go up	The value on the tile was set to 0 and the players balance went up	Pass	None

Jail Class Test

Ref	Description	Input(s)	Expected output(s)	Actual output(s)	Pass/Fail	Action?
1	Can the price to get out of jail be returned	A game board	The value to get out of jail	The value to get out of jail	Pass	None
2	Can the price to get out of jail be set	A game board, the value to assign as the fee	The fee will be set	The fee was set	Pass	None
3	Can a player be taken out of jail	A game board and a player	The player will be removed from jail	The player was removed from jail	Pass	None

Ownable Class Test

Ref	Description	Input(s)	Expected output(s)	Actual output(s)	Pass/Fail	Action?
1	Can an owner be set	A board, a player and an ownable	The owner is set	The owner has been set	Pass	None
2	Can the owner be returned	A board, a player and an ownable	The owner is returned	The owner was returned	Pass	None
3	Is the tile owned	A board, a player and an ownable	True if tile is owned	True when tile is owned, false when not owned	Pass	None
4	Can the price be set	A board, a player and an ownable	The price to buy the tile is set	The price to buy the tile was set	Pass	None

5	Can the price be returned	A board, a player and an ownable	The price to buy the tile is returned	The price to buy the tile was returned	Pass	None
6	Can the mortgage price be set	A board, a player and an ownable	The mortgaged value for the tile is set	The mortgaged value for the tile was set	Pass	None
7	Can the mortgaged price be returned	A board, a player and an ownable	The mortgaged value for the tile is returned	The mortgaged value for the tile was returned	Pass	None
8	Can the sell price be set	A board, a player and an ownable	The price to sell the tile is set	The price to sell the tile was set	Pass	None
9	Can the sell price be returned	A board, a player and an ownable	The price to sell the tile is returned	The price to sell the tile was returned	Pass	None
10	Is the tile mortgaged	A board, a player and an ownable	True if the tile is mortgaged	True if the tile is mortgaged, false if otherwise	Pass	None
11	Can the tile be set to mortgaged	A board, a player and an ownable	The tile is set to mortgaged	The tile was set to mortgaged	Pass	None

Player Class Test

Ref	Description	Input(s)	Expected output(s)	Actual output(s)	Pass/Fail	Action?
1	Can a player buy a tile	A game board and a player	The tile will be purchased	The tile was purchased	Pass	None
2	Can a player sell a tile	A game board and a player	The tile will be sold	The tile was sold	Pass	None
3	Is the player bankrupt	A game board and a player	True if the player is bankrupt, false otherwise	True if the player is bankrupt, false otherwise	Pass	None
4	Can the player mortgage a tile	A game board and a player	The tile will be mortgaged	The tile was mortgaged	Pass	None
5	Can the players name be set	A game board and a player	The players' names will be set	The players' names were set	Pass	None
6	Can the players name be returned	A game board and a player	The players' names will be returned	The players' names were returned	Pass	None
7	Is the player in jail	A game board and a player	True if the player is in jail, false otherwise	True if the player is in jail, false otherwise	Pass	None
8	Can the player be set as in jail	A game board and a player	The player will be set as in jail	The player is in jail	Pass	None
9	Can the players balance be set	A game board and a player	The players balance is set	The players balance was set	Pass	None

10	Can the players position be set	A game board and a player	The players position is set	The players position was set	Pass	None
11	Can the owned tiles of a player be set	A game board and a player	The players owned tiles are set	The players owned tiles were set	Pass	None
12	And the owned tiles of a player be returned	A game board and a player	The players owned tiles are returned	The players owned tiles were returned	Pass	None
13	Can the players piece be set	A game board and a player	The players piece will be set	The players piece was set	Pass	None
14	Can the player be charged a bill	A game board and a player	The player will be charged the bill, pay if possible otherwise a fault will be returned	The player pays the bill where possible, otherwise the fault is returned.	Pass	None

Property Class Test

Ref	Description	Input(s)	Expected output(s)	Actual output(s)	Pass/Fail	Action?
1	Can the cost of a house be set	A game board, a property and a player	The cost of the house will be set	The cost of the house was set	Pass	None
2	Can the cost of a house be returned	A game board, a property and a player	The cost of the house will be returned	The cost of the house was returned	Pass	None
3	Can a house be removed or added	A game board, a property and a player	A house will be removed or added where necessary	If a house is added it is added correctly or removed correctly as necessary	Pass	None
4	Can the amount of houses be returned	A game board, a property and a player	The amount of houses on the property will be returned	The amount of houses on a property were returned	Pass	None

Station Class Test

Ref	Description	Input(s)	Expected output(s)	Actual output(s)	Pass/Fail	Action?
1	Can the rent be returned	A game board, a station and a player	The rent cost for the property	The rent cost for the property	Pass	None
2	Can the rent be set	A game board, a station and a player	The price of rent will be set	The price of rent is set	Pass	None
3	Can the rent be charged	A game board, a station and a player	The price of rent will be charged where possible, else an error will be returned to the UI	The price of rent was charged where possible, otherwise an error was returned to the UI.	Pass	None

Tax Tile Class Test

Ref	Description	Input(s)	Expected output(s)	Actual output(s)	Pass/Fail	Action?
1	Can the value of tax be set	A game board, a tax tile and a player	The value of tax is set	The value of tax was set	Pass	None
2	Can the value of tax be returned	A game board, a tax tile and a player	The value of tax is returned	The value of tax was returned	Pass	None
3	Can tax be charged	A game board, a tax tile and a player	The price of tax will be charged where possible, else an error will be returned to the UI	The price of tax was charged where possible, otherwise an error was returned to the UI.	Pass	None

Tile Class Test

Ref	Description	Input(s)	Expected output(s)	Actual output(s)	Pass/Fail	Action?
1	Can the name of the tile be set	A game board, a tile and a player	The name of the tile is set	The name of the tile was set	Pass	None
2	Can the name of the tile be returned	A game board, a tile and a player	The name of the tile is returned	The name of the tile was returned	Pass	None
3	Can the position of the tile be set	A game board, a tile and a player	The position of the tile is set	The position of the tile was set	Pass	None

4	Can the position of the tile be returned	A game board, a tile and a player	The position of the tile is returned	The position of the tile was returned	Pass	None
5	Can a tile be compared by string name	A game board, a tile and a player	The tile name will be given, and a tile object returned	The tile name was given, and a tile object was returned	Pass	None

Trade Class Test

Ref	Description	Input(s)	Expected output(s)	Actual output(s)	Pass/Fail	Action?
1	Can an ownable be added to the trade	A game board, a trade object and two players	The ownable tile will be added	The ownable tile was added	Pass	None
2	Can the trade be accepted	A game board, a trade object and two players	Mimic an and gate which will only return true if both players agree	Only returns true if both players agree to the trade	Pass	None

Utility Class Test

Ref	Description	Input(s)	Expected output(s)	Actual output(s)	Pass/Fail	Action?
1	Can the rent for landing on the utility be calculated	A game board, both utility's, a group, the dice result and a player	The correct amount for rent given the dice value and the amount of utilities owned	The correct amount for rent given the dice value and the amount of utilities owned	Pass	None

UI Testing

New Game Screen Class Test

Ref	Description	Input(s)	Expected output(s)	Actual output(s)	Pass/Fail	Action?
1	Create a new game, test buttons and ensure that the game type can be changed.	A game engine and a stage.	The buttons will work correctly and the game type can be changed.	The buttons worked correctly and the game type was changed	Pass	None
2	Are all menu buttons working, and can the player navigate back to the main menu	A game engine and a stage.	All buttons within the scope will work correctly and the player can navigate back to the main menu screen	All the buttons within the scope are working and the player can navigate back to the main menu.	Pass	None

Main Menu Screens Class Test

Ref	Description	Input(s)	Expected output(s)	Actual output(s)	Pass/Fail	Action?
1	Can the buttons be clicked	A game engine and a stage.	All buttons within the scope can be clicked on	All buttons in the scope can be clicked on	Pass	None
2	Can the new game screen be reached from the main menu	A game engine and a stage.	The menu will change from one page to another	The menu changed from one page to the other	Pass	None
3	Can the load game screen be reached from the main menu	A game engine and a stage.	The menu will change from one page to another	The menu changed from one page to the other	Pass	None
4	Can the import screen be reached from the main menu	A game engine and a stage.	The menu will change from one page to another	The menu changed from one page to the other	Pass	None
5	Can the settings screen be reached from the main menu	A game engine and a stage.	The menu will change from one page to another	The menu changed from one page to the other	Pass	None
6	Can the game be exited	A game engine and a stage.	The game will close when the exit button is pressed	The game closes when the exit button is pressed	Pass	None

Game Board Class Test

Ref	Description	Input(s)	Expected output(s)	Actual output(s)	Pass/Fail	Action?
1	Can the property details be opened from each different property around the game board into the centre of the screen	A game engine and a stage.	The test should be able to press every tile on the board, and all ownable tiles should open their details in the centre of the screen	The ownable tiles were clicked and their information was pulled into the centre of the screen.	Pass	None
2	Can the player names be displayed on the right-hand side of the board	A game engine and a stage.	The test should be able to begin a new game, selecting different player pieces, entering names and selecting human. Then assert that the name is correct.	The test ensured that all player names appeared and the information that appeared was correct.	Pass	None

The above tests are also a good example of system testing, as such we have negated the need for alpha and beta testing. The reason for this is that the TestFX package can assert that the information on the screen is correct. This allows us to test every piece of information on the screen at any time to ensure the game is working as expected. The other benefit to this is that the tests can be run in headless mode, meaning that the mouse will move to check the buttons work, but not appear on the programmer's computer. This also allows us to run all of the tests using Travis CI, thus fully system test on every pull request.

User guide

Property Tycoon is the game of buying, renting or selling properties. The wealthiest player will win. Moving round the board by rolling a dice, the player lands on different properties. When you land on a property that is not already owned by anyone, you are able to purchase it. Players who own the property receives rent from the other players who land on it. If you run low on money during the game, the player is able to mortgage properties from the bank.

System Requirements

Device Minimum Requirements

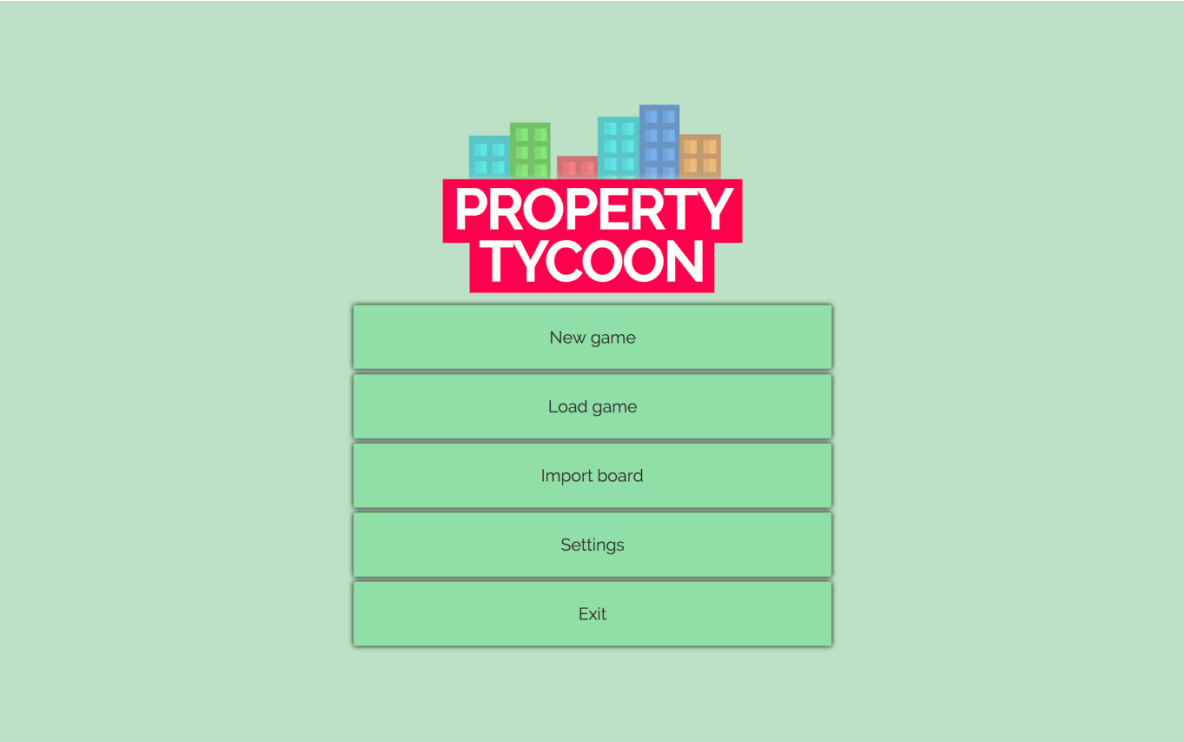
Operating System: Windows / MacOS
Processor: Dual-Core Intel
Memory: 100 MB
RAM: 512 MB
JDK: 8

Device Optimum Requirements

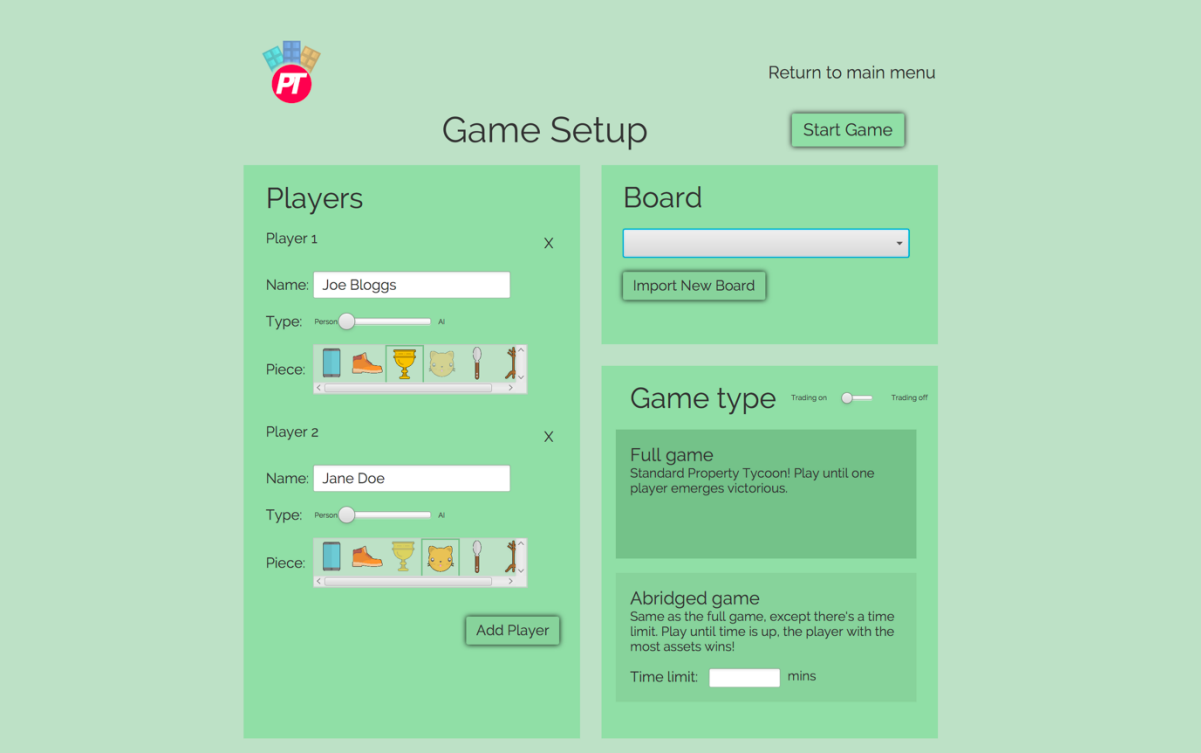
Operating System: Windows / MacOS
Processor: Dual-Core Intel
Memory: 120 MB
RAM: 2 GB
JDK: 8

Set up

To create a new game press the new game button in the main menu:



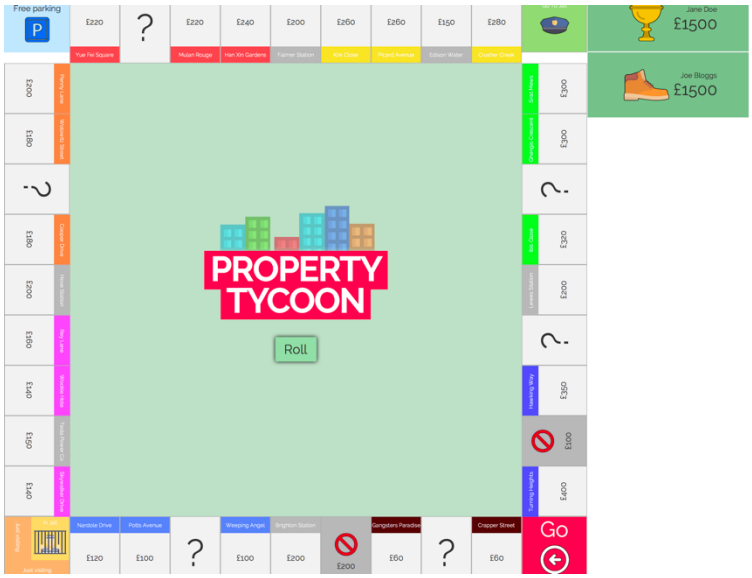
Enter 2-6 players and attach their name, token, and player type.



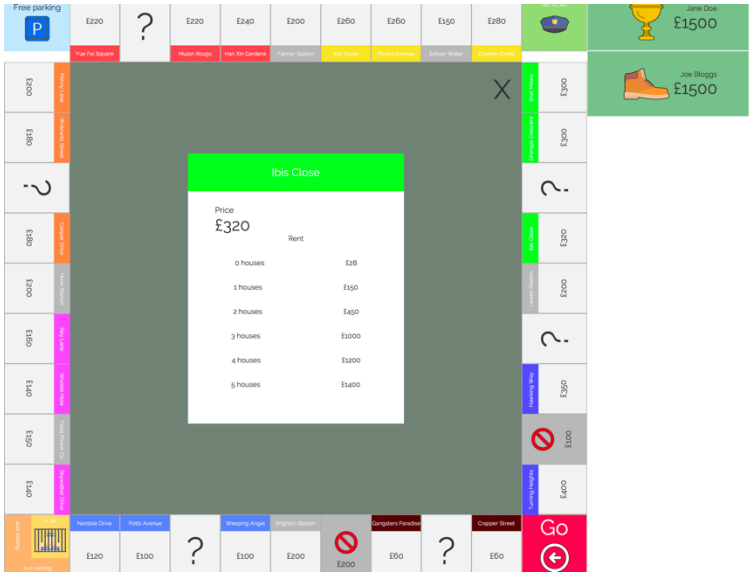
Now select a board from the drop down menu. If the board does not appear, please press the import board button and select board.json form the game directory. This is the default game; personalised game board will be coming soon! Close and open the application again and select the board from the drop-down box. The board will now be loaded, and following selecting your game mode you will be able to play!

General

If the application will not start up, you will need to force quit the application within your operating system's task manager. We then suggest you restart your computer; after you have restarted your computer you will then be able to run the application. Restarting the computer will clear all the cache the application has created which will normally resolve the error.

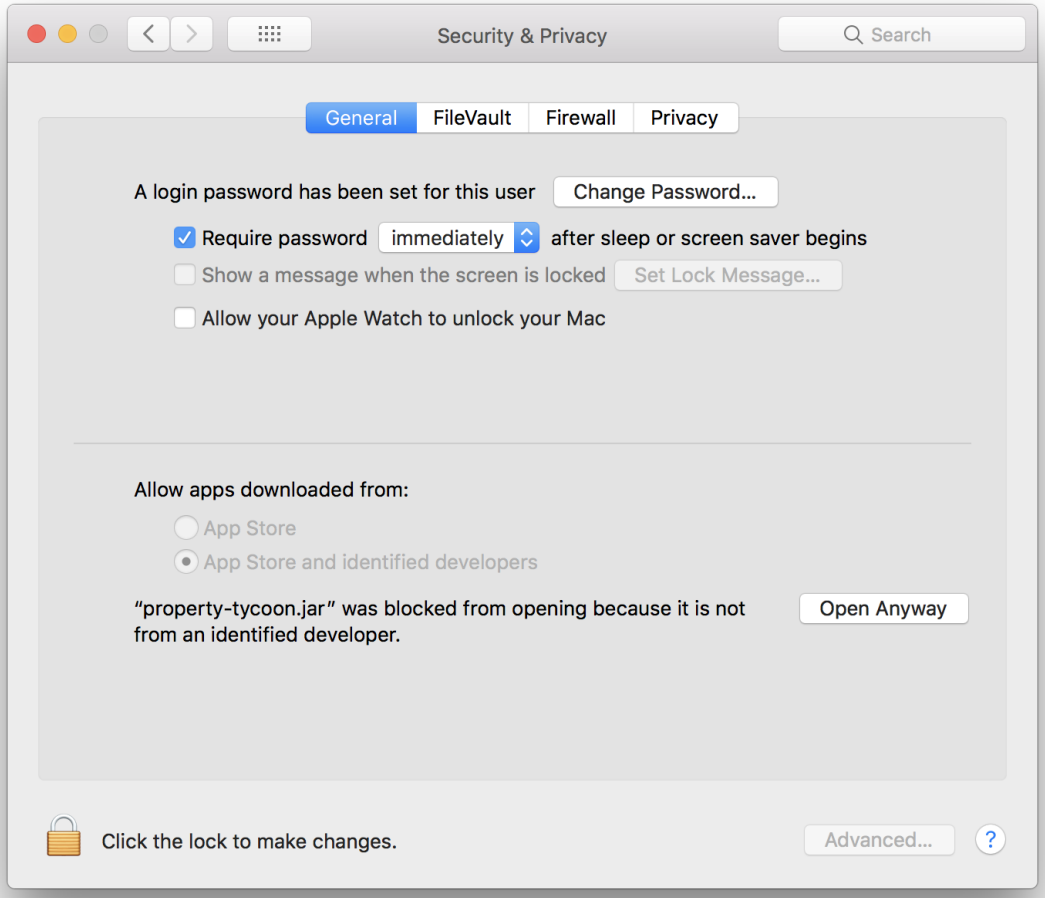


Now the game has loaded, select the roll button to start the game. The dice result will now appear on the screen. To load more information, select the property tiles to see more.



Download and installation

If the .jar file does not open on macOS, load ‘System Preferences’ and select ‘Security & Privacy’. Then select ‘open anyway’.



Contact information

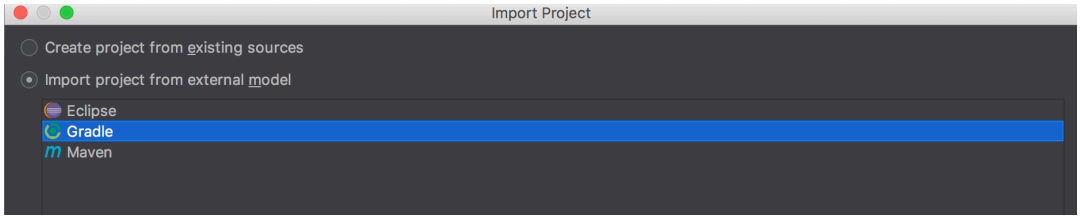
Company name: Team 💯
Email: hello@property-tycoon.com

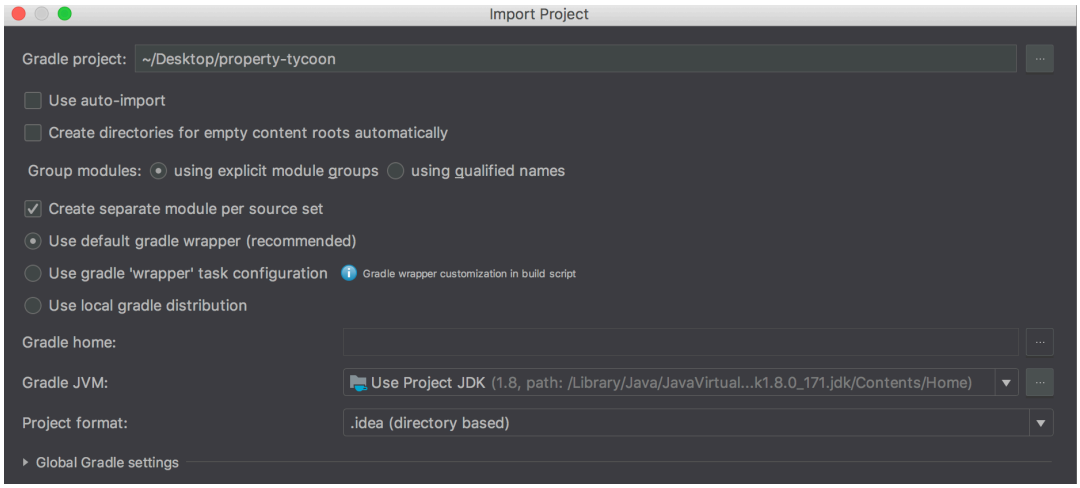
Technical Guide

Getting started

To get started using the project, you will first need to access the online public repository on GitHub (https://github.com/pete183/property-tycoon), or https://property-tycoon.com/ . From here, you will be able to download a clone and thus access all of the source code for the game.

To import the project, you must first open IntelliJ and press import project. After finding the property-tycoon directory, IntelliJ will ask if you would like to import from existing sources or from an external model. Click external model, and then Gradle, followed by the next button in the bottom right corner.





You will then be presented with a settings menu, which will be pre-setup correctly. However, in order to run this project, you must be using Java 8 as the Gradle JVM is Java 8. When you are sure you are using the correct version of java, simply press the next button, then finish. At this point, the import is complete.

Gradle JVM: Java 8

The project utilises the power of Gradle dependency management. The reason this is a key part of the project is to ensure the full development team are using the same version of each dependency and remove a large amount of errors. It will also automatically update the dependencies when new ones are added by other members of the development team.

Modelling Diagrams

In order to update the project, you will require the latest version of the modelling diagrams. These are available in the repository in the folder named “Modelling Diagrams”. Included in this folder are use case, flowchart, class and sequence diagram. The use case and flowchart show the data flow around the system such that as a future developer you can identify how to adapt the system. Further to this, the class diagram can be used to identify how each class connects in java showing how the data connects throughout the game engine and user interface. Finally, the sequence diagram can be used to see the flow of control over time, showing which part of the game currently is running the game.

JSON board file and saving new boards

Utilising JSON we created the standard property tycoon game board. In order to amend the standard board, you need to open the board.json file in any text editor and apply changes. When a player creates a new game, the location of the board is saved into a ‘.ted’ file. This means that the player can load in multiple different boards which could be customisable. Our implementation of Property Tycoon allows users to save games. These games are written into JSON from the variables that are stored in the game engine. These JSON files can then be imported on the main menu to allow players to come back to the game at a later time.

Travis CI

The continuous integration configuration is stored in the travis.yml file. Within this any of the follow configurations can be modified to allow for further updates. If there are any major changes within the project, they can be configured to allow for different languages environments and other implementation scripts.

Option	**Set Value**
Language	Java
JDK	oraclejdk8
Environment Options	_JAVA_OPTIONS="-Djava.awt.headless=true -Dtestfx.robot=glass -Dtestfx.headless=true -Dprism.order=sw -Dprism.text=t2k -Dtestfx.setup.timeout=2500"
JDK package installer	oracle-java8-installer
Script	Gradlew check
Cache	Gradle wrapper and cache

Website / JavaDoc

To edit the documentation on the website, navigate to the markdown files within the doc's directory. Any changes to these .md files will be shown when the website is built, however and new files must be added to sidebar.json. Which is located in the website directory. To edit this file, either add the file ID to one of the current headings, or create another heading following the format given by the previous entries.

To run a local version of the website run the command below. This will all refresh all the changes within the files.

npm start --prefix ./website/

To build the latest version of the website run the following command.

npm run deploy --prefix ./website/

When the website has been updated, a push to the GitHub repository on the develop branch will not update the current live version. However, when the branch is pulled into Master via a pull request, the hosting provider Netlify will build the website before making it public. This can be viewed in the pull request to check formatting and will also run React's unit testing. The other benefit from this is that the branch cannot be merged should the website build fail, ensuring that the current live version is always working. The game download link will always contain the most up to date version on the master branch automatically using continuous integration.

Potential Developments

ML for AI

In order to enhance the functionality of the game, we would like to implement a machine learning technique to generate an effective computer player. There are two methods of achieving this; utilising the power of a genetic algorithm or using a neural network.

We aimed to create a genetic algorithm to optimise the code for the computer player to achieve this. Our genetic algorithm would have been a population of slightly different versions of the computer player. We would then have created a way of assessing the fitness of the current solution in the game; to do this we would have given the GA access to the game, thus after each generation it could run the solutions against each other in order to find the most efficient.

The algorithm would then mutate the code that controls the computer player at random (given the rules of the language) to create a new solution based on two other solutions in the population. Given enough time to process, we would have found the most efficient player that always won. Further to this, we could have stopped the algorithm three different times, which would give us an easy, medium and hard difficulty respectively.

The current solution is accurate however the difficulty cannot be set, and it is based on rules such as for buying a property; the rules would be:

The amount of money it currently has
Does the player own another property in the group?
Does another player own a property in the group?

As such, a more refined player would be a good advancement of the game.

Website to create board

To further keep within the spirit and character of the game, we would have liked to attach extra functionality to our website. Currently in the menu screen is an option to select board; the reason for this is that in the future an advancement to create your own board with its own property names, colours, etc, on the property-tycoon.com website.

To do this we would have had a webpage that displayed the board version, allow an image to be uploaded, and allow property names, prices etc to be set. The website would then convert this information into JSON data which can be linked to the game in the new game screen.

Online multiplayer

We also would have liked to add an online multiplayer. This would have linked players on the windows or OSX platform over LAN/WAN such that they could play in the same game. This functionality would not have been free, which would go against our original feasibility however, it is a good advancement for Watson games, hence we aimed to keep this option available.

To do this, we would have built an API that can link the game data to an online server which would send and receive data from every player, providing the most up to date version of the game.

Evaluation

Following the completion of the project, in order to assess the success of the software and the development we must look back to the original task we set out to achieve.

Our client, Watson Games, came to us with a brief requiring the creation of a digital version of their game Property Tycoon. The client has 50 years of experience making board games, however has no experience with electronic games. The owner, Quentin Raffles, asked our company to digitise the game in the same format, stating that the game should be "fun to play," and have a "colourful and intuitive interface" that reflects the "spirit and character of the original board game".

In order to complete this we set out making a requirements document to give us a good idea of what the company was asking for. These were split into three categories: functional, non-functional and domain requirements. Our primary concern was meeting the customers expectation; thus, to evaluate we will look at the primary target requirements of the game.

Firstly, the client required the game to closely resemble their original board game. To do this we closely studied the intricacies of the game and mimicked them on screen. Prior to the implementation of this, we went through and created interactive wireframes which closely resembled the working game. After this, we then created all the graphics ready for implementation.

We then went through and created all the high-level and low-level diagrams including: Use case, flow charts of the game engine, class and sequence diagrams. These were used to keep everyone throughout the team on track implementing the same system. This was so the UI and the Game Engine could be connected without compatibility issues.

Talk about risks we encountered and how we came over them //TODO
Analysis of the team working together

Peer marking

Name	**Mark**
Peter Lloyd	20
Liam Scriven	20
Guy McClenahan	20
Elliot Massen	20
Sam Kennard	20

Appendices – Stored within the portfolio folder

i. Gantt / PERT Chart
ii. Graphics
iii. Wireframes
iv. PDF's of Modelling diagrams pre-trading and post trading
v. Minutes and agendas