I.T.I.S. A. Avogadro
Liceo scientifico tecnologico

Giobergia Flavio
5^A Liceo A.S. 2011/2012

Indice

1. *Monopoly*

1.1 Cos'è *Monopoly*

Monopoly è un gioco da tavolo nato quasi 80 anni fa e che dal 1934, anno della sua pubblicazione, detiene il record nel Guinness dei Primati come gioco (protetto da copyright) più giocato, con oltre un miliardo di giocatori.
Si basa sul concetto di monopolio, ossia di controllo del mercato da parte di una singola figura.

1.2 Regolamento

1.2.1 Scopo del gioco

Lo scopo del gioco è quello di immedesimarsi in un proprietario terriero senza scrupoli, che compra, costruisce, contratta e scambia proprietà per arricchirsi, facendo impoverire gli altri giocatori. L'ultimo giocatore a rimanere in campo è il vincitore.

1.2.2 Materiale necessario

- Pedine, una per giocatore
- Tabellone
- 2 dadi
- Elenco di tessere Probabilità e Imprevisti
- Elenco delle tessere delle proprietà
- Soldi (finti)
- Case e alberghi

Materiale necessario

1.2.3 Preparativi

Uno dei giocatori (o un esterno) svolge il ruolo di "banca": questi deve occuparsi dei preparativi e di gestire i soldi della banca secondo quanto richiesto dal gioco.
Il banchiere deve distribuire ad ogni giocatore la stessa quantità di denaro iniziale, che verrà concordata in precedenza e che dovrà essere proporzionale al numero di giocatori.
Oltre a distribuire i soldi, è possibile distribuire un numero predeterminato di proprietà, per rendere il gioco più interessante già dall'inizio e ridurne la durata. Anche il numero di proprietà distribuite deve essere proporzionale al numero di giocatori e alla durata desiderata del gioco.
Ogni giocatore, quindi, lancia i due dadi per determinare chi dovrà iniziare. Colui che realizza il punteggio più alto inizierà il gioco. L'ordine degli altri giocatori può essere orario o antiorario a partire dal primo, oppure può essere determinato a seconda dei punteggi ottenuti dal lancio dei dadi.

1.2.4 Proprietà

Le proprietà sono normalmente divise in 3 categorie: terreni, stazioni ferroviarie e compagnie.
I terreni sono le proprietà più comuni, ve ne sono 22 sul tabellone classico. Queste proprietà sono raggruppate in gruppi da 2 o 3 terreni, contraddistinti dallo stesso colore. Un giocatore che finisce su una proprietà che non appartiene a nessun altro giocatore, può acquistarla dalla banca, pagando il prezzo riportato sulla tessera del terreno stesso.
Se un giocatore, invece, atterra su una delle proprietà in possesso di un altro giocatore, il primo dovrà pagare una somma di denaro al proprietario. La somma che dovrà essere versata può variare in base a diversi parametri che verranno illustrati in seguito, ed è sempre riportata sulla tessera del terreno in questione.
Se un giocatore entra in possesso di tutti i terreni di un certo gruppo, può procedere alla costruzione di case e/o alberghi sui terreni stessi. Il costo di una casa o albergo è riportato sulla scheda del terreno. La costruzione di un albergo è consentita solo se sono già presenti 4 case sul terreno, ed è possibile costruire un solo albergo per terreno.
La costruzione delle case deve avvenire in modo proporzionato su tutti i terreni del gruppo; non è per esempio possibile costruire due case su un terreno se non ve ne è nessuna sugli altri, mentre è possibile costruirne due su un terreno se sugli altri ne è presente già una.
La costruzione di case aumenta il costo del pedaggio che gli altri giocatori dovranno versare al proprietario.
Il numero di case e alberghi totali presenti è limitato (normalmente, ci sono 32 case e 12 alberghi disponibili); qualora non fossero più disponibili case o alberghi, è necessario aspettare che un giocatore rimuova dai propri terreni delle case (per necessità di soldi o per costruire un albergo) o alberghi.
Se nessuna casa è presente su un gruppo di terreni in possesso dello stesso giocatore, gli altri giocatori che transitano su quei terreni devono pagare una somma pari al doppio del pedaggio normale.
Un altro tipo di proprietà è la stazione ferroviaria: ve ne sono 4 sul tabellone tradizione, in posizione centrale su ognuno dei lati. Il transito su una stazione in possesso di un altro giocatore porta al pagamento di una somma riportata sulla tessera della stazione stessa. Se il giocatore a cui va versata la somma è proprietario di più stazioni, la somma da versare aumenta di conseguenza, come riportato sulla tessera.
L'ultimo tipo di proprietà è la compagnia (o società): ve ne sono due nel gioco (società elettrica e dell'acqua potabile) e non comportano un pagamento fisso ad ogni passaggio. Se un giocatore si ferma su una di queste proprietà, dovrà pagare al proprietario una somma pari a 4 volte il numero comparso sui dadi durante l'ultimo lancio. Se il proprietario della società è in possesso anche

dell'altra, la somma da pagare sarà 10 volte il numero comparso sui dadi. In alcune versioni di *Monopoly* viene imposta una somma fissa da pagare. Ciò può essere verificato sulla tessera corrispondente alla proprietà.

1.2.5 Prigione

Nel gioco è presente una prigione, generalmente collocata 10 caselle dopo il Via. Se un giocatore finisce sulla casella "Prigione" in seguito ad un normale lancio dei dadi, si dice che è "in transito", ossia non viene considerato in prigione, ma solo di passaggio.
Per finire in prigione, vi sono 3 modi diversi:
il primo è quello di finire sulla casella "In Prigione" (che si trova solitamente 20 caselle dopo la Prigione). Chi finisce su quella casella viene mandato direttamente in Prigione.
Il secondo modo è quello di realizzare tre "doppi" con i dadi: quando un giocatore tira i dadi, se i due dadi mostrano lo stesso punteggio (ossia se realizza un "doppio") dovrà ritirare nuovamente e avanzerà ulteriormente. Se però il giocatore, durante lo stesso turno, realizza tre tiri doppi di seguito, verrà immediatamente mandato in Prigione.
L'ultimo modo per finire in Prigione è quello di pescare una tessera di Probabilità o Imprevisti che ordina di andarvi.
Anche per uscire dalla Prigione, vi sono 3 modi: il primo è quello di pagare la cauzione (soldi che finiranno alla Banca),in modo da uscire subito; il secondo è quello di tirare i dadi e realizzare un doppio, il terzo è quello di usare un buono "Esci gratis di Prigione", che si può ricevere dalle Probabilità o Imprevisti (o che può essere scambiato con un altro giocatore).
Se, dopo 3 turni, un giocatore è ancora in Prigione (ossia non è riuscito a realizzare un doppio tirando i dadi), dovrà obbligatoriamente pagare la cauzione o usare un buono "Esci gratis di Prigione".

1.2.6 Probabilità & Imprevisti

Sono presenti, tra le caselle del tabellone, anche alcune denominate "Probabilità" ed altre dette "Imprevisti". Quando un giocatore capita su una di queste caselle, deve pescare una tessera dal mazzo corrispondente (mazzo Probabilità o mazzo Imprevisti) ed eseguire le azioni riportate sulla tessera stessa.
Le istruzioni riportate possono portare a svantaggi o vantaggi per il giocatore: potrà ricevere o dovrà versare somme di denaro, venire spostato di posizione, ricevere buoni per uscire gratis di prigione o altro ancora.

1.2.7 Tasse

Sono solitamente presenti due caselle sul campo da gioco che richiedono, a chi finisce sopra, di pagare un certo quantitativo di tasse, scritto sulla casella stessa.

1.2.8 Altre caselle

Via: è la casella da cui i giocatori partono. Ogni volta che un giocatore finisce un giro completo e passa nuovamente sul Via, riceverà dalla Banca una certa quantità di denaro, scritta sulla casella stessa.
Parcheggio gratuito: il parcheggio gratuito è una casella che non ha alcun effetto sul corso del gioco: chi vi atterra sopra non dovrà riscuotere, pagare, spostarsi o altro.

1.2.9 Debito e bancarotta
Qualora un giocatore si ritrovi in debito di una somma di cui non dispone in contanti (ovvero ha bisogno di soldi per costruire o comprare altre proprietà), può ricorrere a vari metodi per entrare in possesso di denaro: può togliere delle case o alberghi dalle proprie proprietà, ricevendo dalla Banca la metà dei soldi spesi per la costruzione dell'edificio (togliendo case vale la stessa regola di proporzionalità spiegata in precedenza per la costruzione); può ipotecare una proprietà per ricevere dalla Banca la metà del prezzo della proprietà (un terreno non può essere ipotecato se presenta delle case su di esso; inoltre se un giocatore finisce su un terreno ipotecato, non dovrà pagare niente al suo proprietario); il terzo modo è quello di scambiare o vendere una o più proprietà ad altri giocatori: i terreni non possono essere venduti però se prima non sono state tolte le case e/o alberghi presenti sullo stesso.
Se un giocatore, nonostante queste possibilità, non riesce a raccogliere la cifra necessaria di cui è in debito, si dichiara in bancarotta: per questo giocatore il gioco finisce e tutte le sue proprietà verranno date al creditore.
Se il creditore non è nessuno dei giocatori (per esempio se non si hanno i soldi per pagare la cauzione della Prigione o si deve pagare una certa cifra a causa di una scheda Probabilità o Imprevisti), le proprietà vengono rese di nuovo disponibili per l'acquisto.

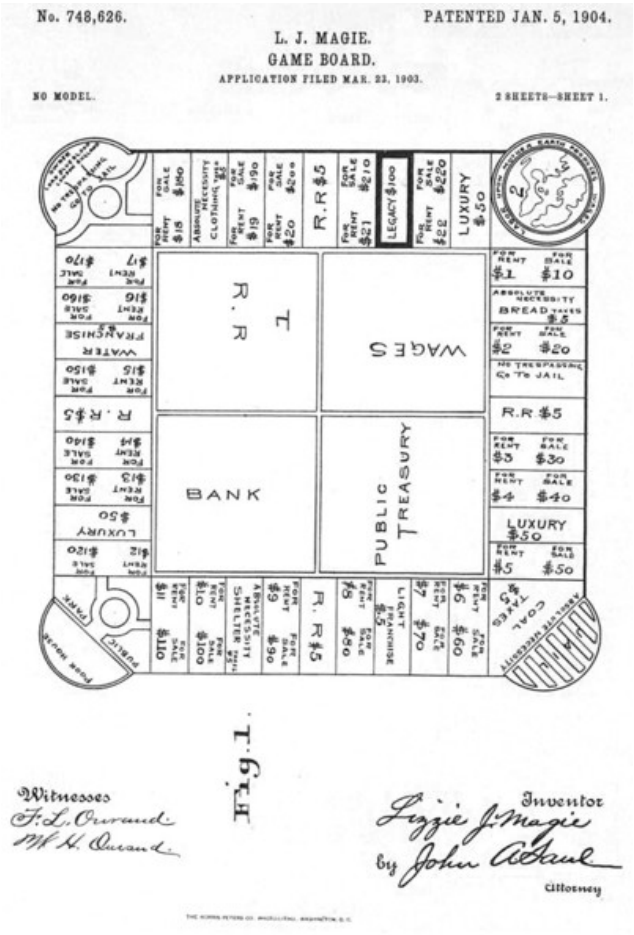
1.2.10 Vincitore
Il vincitore sarà l'ultimo giocatore rimasto, dopo che tutti gli altri avranno dichiarato bancarotta.

1.3 Varianti
Essendo *Monopoly* un gioco molto famoso, nel corso degli anni sono nate diverse varianti: alcune, “non ufficiali”, prevedono cambiamenti del regolamento, aggiungendo, togliendo o sostituendo regole per rendere il gioco più o meno difficile. Altre varianti più radicali sono state introdotte dai produttori stessi, con la vendita di estensioni per il gioco, edizioni speciali e altro ancora.
Di particolare interesse è, per l'Italia, la versione prodotta da *Hasbro* in occasione dei 150 anni di Unità: per tale versione, sono stati utilizzati come nomi dei terreni alcune città italiane. La scelta delle città non è stata fatta in base alla fama o alle dimensioni, ma con un sondaggio online. Per un certo periodo di tempo gli italiani hanno potuto votare su un sito la propria città, per farla comparire tra le 22 del gioco.
Decisamente strani sono stati i risultati: grandi città come Roma, Bologna, Firenze e Venezia non hanno ricevuto abbastanza voti, mentre Torino e Milano (quest'ultima in seguito all'invito a votare fatto da Letizia Moratti stessa ai suoi cittadini) ce l'hanno fatta.
La città che ha ricevuto più voti in assoluto è stata Chieti, seguita da Reggio Calabria e Catanzaro.
Diverse città del Sud hanno avuto successo, tra cui Barletta, Andria, Brindisi, Foggia, Cosenza.
E ancora, fra le più votate ci sono Messina, Ischia, Teramo, Ascoli Piceno, Terni, L’Aquila, Isola d’Elba, Sanremo, Caserta, Viareggio, Trani, e, ovviamente non poteva mancare, la città di Monopoli (BA).

2. Storia

2.1 *The Landlord's Game*

Alla sua nascita *Monopoly* non poté essere considerato esattamente una novità: già alcuni giochi da tavolo simili lo precedettero, tra i tanti, il più importante era certamente *The Landlord's Game*, che viene considerato l'antenato "ufficiale" di *Monopoly* (ufficiale perché si trattò del primo gioco del genere registrato all'ufficio brevetti).
The Landlord's Game (Landlord, in inglese, è il locatore) venne brevettato dall'americana Elizabeth Magie nel 1904, ma la produzione iniziò solo nel 1910.
Inizialmente la vendita avvenne solo in America, ma, dato il successo, nel 1913 iniziò la vendita anche in Gran Bretagna, dove venne presentato con un nome diverso (*Brer Fox an' Brer Rabbit*).
Magie, dopo essersi trasferita con il marito in Washington D.C., brevettò una nuova versione del gioco nel 1924. Questa nuova versione differiva da quella precedente per dettagli minori, come l'aggiunta di nomi alle varie strade e proprietà.
Anche la nuova versione incontrò il favore del grande pubblico; iniziarono così a nascere le prime varianti del gioco "fatte in casa".
Magie detenne il brevetto fino al 1935, quando la *Parker Brothers* glielo comprò per 500$.



Disegno del tabellone di *The Landlord's Game*

La *Parker Bros.* aveva appena iniziato a vendere *Monopoly*, ed era intenzionata a comprare tutti i brevetti di giochi simili, in modo da evitare eventuali problemi legali ed essere la sola in grado di vendere tale gioco.

2.2 La nascita di *Monopoly*

Quando il gioco *The Landlord's Game* cominciò a diffondersi, come già detto, vennero prodotte diverse copie "casalinghe".
Dato che il gioco originario venne brevettato, fu necessario utilizzare un altro nome per queste versioni; molti, quindi, cominciarono a chiamare queste versioni casalinghe *Auction Monopoly* (auction, in inglese, significa vendita all'asta, e si riferisce al fatto che le proprietà vengono messe all'asta durante il gioco) o, più semplicemente, *Monopoly*.
Queste nuove versioni introducevano nuove regole e modificavano le precedenti.
Molte persone, oltre a giocare semplicemente con queste regole, decisero di brevettarle.
Tra queste persone figura anche Dan Layman, che nel 1932 vendette la sua versione a una società di Indianapolis (la *Electronic Laboratories*), che la commercializzò con il nome di *Finance*.

Un certo Ruth Hoskins imparò tale gioco da Layman stesso, e, quando si trasfersì ad Atlantic City, cominciò a giocare a una versione leggermente modificata con diversi amici.
Con il passaparola, anche un certo Charles Todd venne a conoscenza del gioco: questi invitò una sua amica con il fidanzato (e futuro marito) Charles Darrow per una partita.
Darrow, un ingegnere che per colpa della crisi del '29 si trovava senza lavoro in quel periodo, si appassionò a tale gioco, tanto che cominciò a creare la propria versione, modificando le regole, il tabellone e apportando altre modifiche minori.
Così, nel 1934, Darrow provò a proporre alla *Parker Brothers* il suo gioco, ma si vide respingere l'idea.
L'ingegnere decise comunque di produrre il gioco per conto proprio, per metterlo poi in vendita in un grande magazzino di Filadelfia.
In breve tempo, più di 5000 copie vennero vendute e, a questo punto, la *Parker Brothers* decise di acquistare il gioco, che venne commercializzato con il nome di *Monopoly*.
Dal momento che esistevano ed erano sul mercato diverse versioni simili a *Monopoly*, la *Parker Bros.* comprò i brevetti di *The Landlord's Game* e *Finance* (in quest'ultimo caso, venne comprata tutta la *Electronic Laboratories*).

2.3 Get Out of Jail Free: *Monopoly*'s Hidden Maps

The following article explains how *Monopoly* boxes were used, during WWII, to sneak maps and other objects into nazi war camps, in order to help POWs break out of those places.

Source: **ABCNews**
Author: **Ki Mae Heussner**

It's a story that will forever change the way you think of the phrase, "Get Out of Jail Free."

During World War II, as the number of British airmen held hostage behind enemy lines escalated, the country's secret service enlisted an unlikely partner in the ongoing war effort: The board game *Monopoly*.

It was the perfect accomplice.

Included in the items the German army allowed humanitarian groups to distribute in care packages to imprisoned soldiers, the game was too innocent to raise suspicion. But it was the ideal size for a top-secret escape kit that could help spring British POWs from German war camps.

The British secret service conspired with the U.K. manufacturer to stuff a compass, small metal tools, such as files, and, most importantly, a map, into cut-out compartments in the *Monopoly* board itself.

"It was ingenious," said Philip Orbanes, author of several books on *Monopoly*, including "The World's Most Famous Game and How it Got That Way." "The *Monopoly* box was big enough to not only hold the game but hide everything else they needed to get to POWs."

British historians say it could have helped thousands of captured soldiers escape.

So how did a simple board game end up in a position to help out one of the most powerful military

forces on the planet? Silk and serendipity.

Silk Maps Were Key Escape Kit Elements

Of all the tools in a military-grade escape kit, the most critical item was the map. But paper maps proved too fragile and cumbersome, said Debbie Hall, a cataloguer in the map room at the Bodleian Library at the University of Oxford in Oxford, England.

For hundreds of years, even before World War II, silk was the material of choice for military maps, Hall said, because it wouldn't tear or dissolve in water as easily as paper and was light enough to stuff into a boot or cigarette packet. Unlike maps printed on paper, silk maps also wouldn't rustle and attract the attention of enemy guards, she said.

"Initially, they had some problems printing on silk," Hall said. "It's quite technically challenging."

But then MI9, the British secret service unit responsible for escape and evasion, found the one British company that had mastered printing on silk: John Waddington Ltd., a printer and board game manufacturer that also happened to be the U.K. licensee for the *Parker Bros.* game *Monopoly*.

"Waddingtons in the pre-war era was printing on silk for theater programs. For celebration events for royalty and that kind of thing," said Victor Watson, 80, who retired as chairman of the company in 1993. "It made a name for itself for being able to print on silk."

He was just a child during the war but said his father Norman Watson, president of the company at the time, worked with British secret service to embed the maps in *Monopoly* games.

He said a secret service officer named E.D. Alston (known around Waddington as "Mr. A.") used to come by to place the orders in person.

"Because he was in the secret service, I never knew who he was," Watson said.

Before leaving for missions, British airmen were told that if they were captured, they should look for escape maps and kits in *Monopoly* boards and other games delivered by charity groups. They were told that "special edition" *Monopoly* sets would be marked with a red dot on the free parking space.

A Monopoly board and a silk map

Watson said that in addition to the concealed compass, tools and maps, real bank notes were hidden under the fake money.

During the war, the Official Secrets Act prevented anyone involved from disclosing the plan, and Watson said his father was concerned that the company could be targeted by the Germans if they were tipped off

"It was very special and very secretive," Watson said, adding that he didn't learn about the company's role helping the military until years later.

Different Maps for Different Regions

Waddington printed six different maps that corresponded with regions surrounding six different German camps, Orbanes said. *Monopoly* kits bound for a camp in Italy, for example, would include a map of Italy and Italian currency (lira).

To make sure each set reached its destination, the secret service devised another code.

"Each game was pinpointed as to the camp it would go to," Orbanes said. To innocuously tag each board game, a period was added after different locations on the board.

A period after "Mayfair," for example, meant that the game was intended for Norway, Sweden and Germany. And a period after Marylebone Station meant it was a game destined for Italy. (It being a British version of game, London streets replaced the Atlantic City streets used in the original American version.)

Hundreds of Thousands of Silk Maps Helped POWs Escape During WWII

While "Mr. A." may have been responsible for bringing the war to Waddington's door, map experts credit another MI9 officer, Christopher Clayton Hutton, with hatching the master plan.

"He put two and two together," Hall said, adding that Hutton was likely not alone in implementing it. "He was the first who had this idea to get maps into camps concealed in board games. It looks innocent, they wouldn't arouse any suspicion... it just looked like someone was being charitable."

Hall and others familiar with the *Monopoly* maps say not wanting to compromise the integrity of the Red Cross, the secret service created fake charity groups to smuggle the games into the German camps.

Barbara Bond, Pro-Chancellor at the U.K.'s University of Plymouth who is writing a book on silk maps, said *Monopoly* games weren't the only vehicles used to conceal escape maps. Decks of cards, the board game Snakes and Ladders and pencils also concealed maps for prisoners.

"There was a whole industry going on," she said.

During the war, hundreds of thousands of silk maps were used to help prisoners escape. And she said it marked a change in the way the military viewed POWs.

During World War I, she said, "If you were captured in battle that was it."

But after Winston Churchill and others shared their experiences as POWs, she said, the perception of them changed.

"The POWs could still do a job," Bond said. "Not only was it their duty to fight if they were captured, it was their duty to escape."

The silk (and rayon) maps and the clever ways they were distributed, she said, reflected that philosophy.

All 'Special Edition' *Monopoly* Sets Destroyed

Though silk maps from that era exist in libraries, homes and museums around the world, none of the original rigged *Monopoly* sets still remain.

After the war, everything was destroyed, Watson said.

But though the games themselves are gone, their legacy is a source of pride for the makers of *Monopoly*, past and present.

"Since Charles Darrow created *Monopoly* in the 1930s, the game has had a rich and interesting story. The use of *Monopoly* by the British government to sneak maps, money and supplies to prisoners of war during World War II is a little-known, but important part of our history," said a spokeswoman for toymaker Hasbro, Inc. "We are always honored when this iconic game becomes an important part of the fabric of a family's, or a country's, history and memories."

In the 1970s, Watson had the chance to meet a few former POWs who actually used Wadddington's maps to escape from a prisoner camp at Colditz Castle, near Leipzig, Germany.

"It was really exciting," he said. Although it's impossible to know precisely how many prisoners escaped with the help of the hidden maps, experts estimate that about 35,000 members of the British, Commonwealth and U.S. forces who were taken prisoner during the war returned to Allied lines before the end of the war.

"We reckon that 10,000 used the *Monopoly* map," Watson said.

2.4 *Monopoly* ai giorni nostri

Monopoly, ai giorni nostri, continua ad essere uno dei giochi più diffusi, riuscendo a tener testa all'avvento delle nuove tecnologie: molti giochi da tavolo, infatti, con l'arrivo delle console e dei computer sono scomparsi, mentre altri giochi, come *Monopoly*, si sono adeguati, riuscendo non solo a resistere, ma anche a trarre vantaggio da questa situazione.
Dal 2008, la *Electronic Arts* ha sviluppato questo gioco per le piattaforme più recenti: è quindi possibile trovare versioni per *PlayStation*, *WII*, *Xbox*, ma anche smartphone, tablet e computer.
Nonostante le varie versioni, non è stata pubblicata alcuna versione ufficiale per il sistema operativo Linux: diversi programmatori indipendenti, quindi, hanno provveduto a crearsi le proprie versioni.
Le più diffuse sono *Kapitalist, Atlantik* (per *KDE*, uno dei Desktop Enviroment più diffusi su Linux) e *GtkAtlantik* (per *Gnome*, un altro dei principale DE).
Queste versioni sono open source, ossia i codici sorgenti sono resi pubblici per essere modificati a piacimento, ma il livello di personalizzazione consentito all'utente medio senza conoscenze di programmazione è molto basso: il tabellone utilizzato non può essere modificato, le proprietà sono fisse e molti altri parametri non sono modificabili.
Proprio per queste limitazioni ho deciso di sviluppare una nuova versione di *Monopoly* per Linux, in modo da dare la possibilità a chiunque, con pochissimo sforzo e conoscenze, di personalizzare a proprio piacimento quanti più aspetti possibili del gioco.

3. Critica al gioco

3.1 Carattere diseducativo

Nonostante il successo, non sono poche le critiche mosse contro questo gioco. Molti criticano il modo in cui è stato progettato: nel gioco, infatti, la maggior parte del tempo è spesa ad aspettare che gli avversari completino il proprio turno, il che può diventare noioso dopo poco tempo.
Inoltre il gioco, a detta di altri, pecca di strategia, in quanto è semplicemente necessario costruire più case e hotel possibili, e affidare il resto alla pseudo-casualità che regola il lancio dei dadi.
Un'altra critica, molto più pesante, viene spesso mossa nei confronti di *Monopoly*: questo gioco infatti è ritenuto diseducativo proprio per l'idea di base, ossia quella di ridurre in polvere i propri avversari per arricchirsi sempre più. Per avere un vincitore, infatti, gli altri giocatori devono essere mandati in bancarotta. Quest'ideologia non è, per molti, adeguata al pubblico infantile.

3.2 *Monopoly* bandito in URSS e Cuba

In URSS, il gioco venne bandito per la sua ideologia di carattere capitalistico. Solo con la dissoluzione dell'Unione Sovietica il gioco venne reso legale, e venne commerciata una versione in cirillico.
A Cuba, invece, il gioco è stato bandito fin dall'inizio del governo di Fidel Castro, sempre perché considerato troppo vicino alle idee capitalistiche, ed è tuttora illegale.

4. Modelli matematici

4.1 Catena di Markov

Monopoly, come la maggior parte dei giochi, può essere rappresentato utilizzando un modello matematico.
Dal momento che la posizione verso cui ci si muoverà ad un determinato turno non è legata a quanto è accaduto in precedenza, il modello sarà a "mancanza di memoria", termine che sta a significare che, appunto, gli spostamenti che avvengono non sono condizionati da eventi precedenti, ma solo dalla probabilità legata al lancio dei dadi.

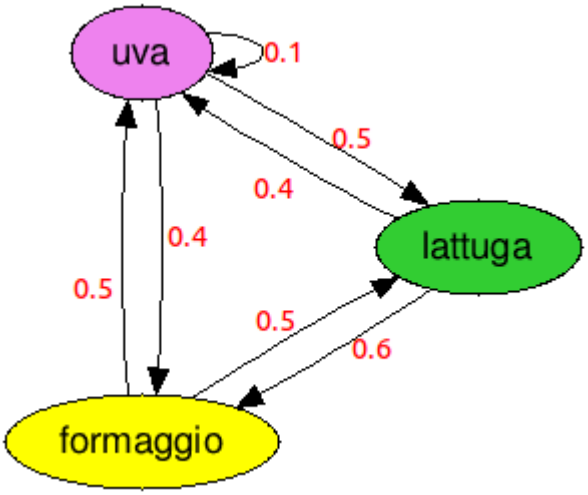
Un modello con queste caratteristiche è la catena di Markov, che prende il nome dal matematico russo Andrey Markov.
Una catena di Markov rappresenta dei processi pseudo-casuali non legati tra loro: lo stato successivo dipende solo dallo stato corrente, e non da quelli precedenti.
Un esempio banale utilizzato per descrivere un impiego della catena di Markov è il seguente:

C'è una creatura che si nutre solo di formaggio, uva o lattughe, servendosi di regole ben precise:

- mangia una volta al giorno
- se oggi ha mangiato formaggio, domani mangerà uva o lattuga con la stessa probabilità
- se oggi ha mangiato uva, domani mangerà uva con una probabilità del 10%, formaggio con una probabilità del 40% o lattuga con una probabilità del 50%
- se oggi ha mangiato lattuga, domani mangerà uva con una probabilità del 40%, o formaggio con una probabilità del 60%

È evidente che la situazione descritta si basa solo su quanto accade un determinato giorno, senza includere quanto accaduto i giorni precedenti: questa situazione, quindi, può essere descritta con una catena di Markov.

Un modo efficiente per descrivere una catena di Markov come quella appena illustrata, è quella di usare dei grafi orientati (grafi formati da un numero finito di nodi e archi orientati).



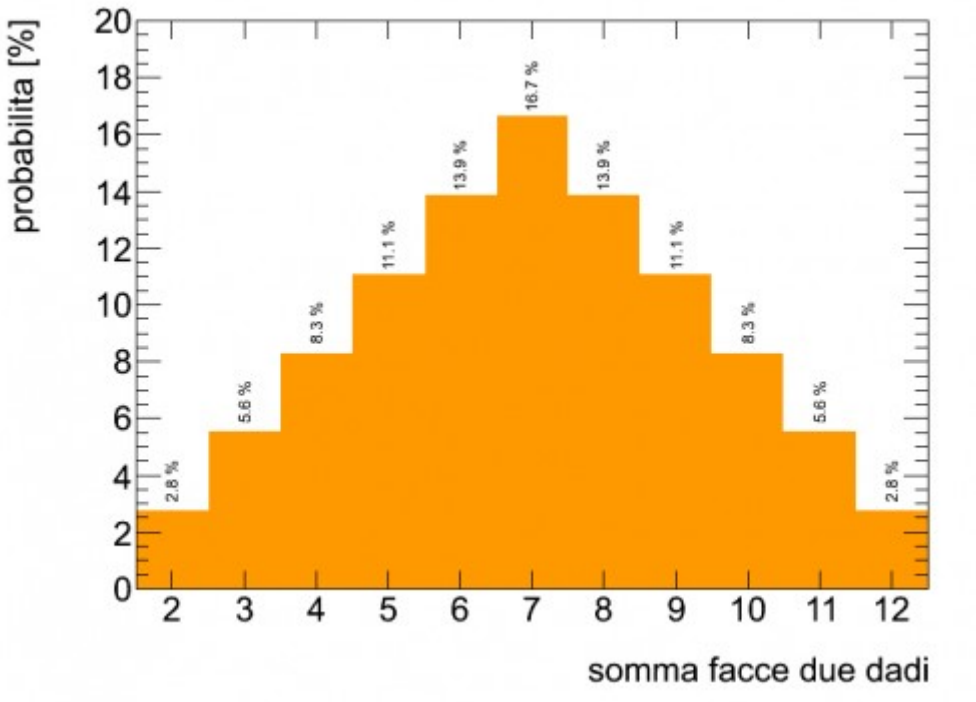
Nota: Il grafo è stato fatto utilizzando *graphviz*, un insieme di librerie e software per la

visualizzazione di grafi. Il linguaggio usato è molto intuitivo, e consente la creazione di grafi (come quello appena riportato) con poche righe, come è possibile vedere nel sorgente riportato qui sotto. Le probabilità scritte vicino agli archi sono invece state aggiunte utilizzando *gpaint*.

```
digraph g{
	a -> b
	a -> c
	b -> c
	c -> a
	b -> a
	c -> b
	b -> b
	a [label="formaggio",fillcolor="yellow",style="filled"];
	b [label="uva",fillcolor="violet",style="filled"];
	c [label="lattuga",fillcolor="limegreen",style="filled"];
}
```

Ovviamente, questo è un esempio banale di come può essere utilizzata una catena di Markov, ma quest'ultima può essere utilizzata anche in casi più complessi, come accade per *Monopoly*.

Nel *Monopoly*, infatti, ad ogni turno vengono lanciati due dadi, e dalla casella in cui ci si trova ci si sposta del numero realizzato con i dadi.
Se dovessimo pensare alle caselle come ai nodi di un grafo, da ogni casella partirebbero 11 archi (uno per ogni diverso numero formato dai due dadi) collegati a 11 nodi diversi e consecutivi.
Le probabilità sarebbero, ovviamente, quelle del lancio di due dadi:



(Immagine presa da www.borborigmi.org)

Questa, però, sarebbe una semplificazione eccessiva del gioco, in quanto ci sono alcuni eventi particolari che consentono il passaggio a caselle diverse da quelle normalmente accessibili con un lancio dei dadi.

Uno di questi eventi è il finire su una casella “Probabilità” o “Imprevisti”: finendo su una di queste

caselle, infatti, il giocatore potrebbe pescare una carta che lo farà procedere fino a un'altra casella. Da questi nodi non è possibile far partire lo stesso numero di archi usati in precedenza, in quanto sarà anche necessario usarne alcuni per collegare le caselle indicate dalle carte Probabilità o Imprevisti: questi nuovi archi avranno come probabilità associata quella di pescare la relativa carta da uno dei due mazzi. Gli 11 archi relativi ai vari tiri dei dadi, inoltre, avranno anch'essi la probabilità cambiata, in quanto le percentuali riportate sopra dovranno essere calcolate rispetto alla probabilità di non pescare una carta che farà cambiare posizione.
Un'altra casella speciale è “In prigione”, che manderà chi vi transita sopra direttamente in Prigione: da questo nodo partirà un solo arco, che punterà alla Prigione e avrà come probabilità associata 1 (100%).

Una volta in prigione, poi, si verifica un'altra distinzione, ossia se il giocatore intende tirare i dadi per provare a fare un doppio ed uscire o decide di pagare subito: ciò va, ovviamente, ad influenzare sulle probabilità.

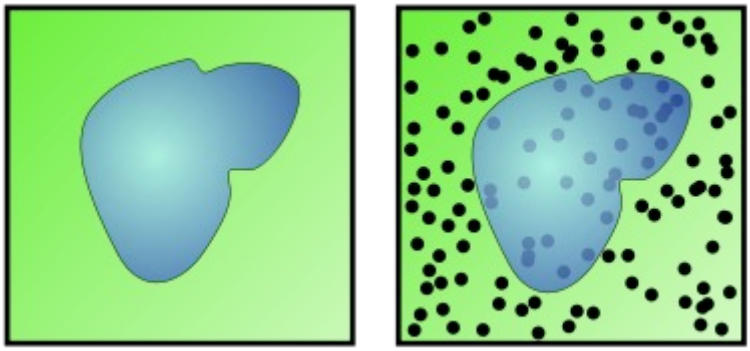
Le variabili che condizionano le probabilità di visitare più una casella rispetto ad un'altra sono molte, risulta quindi complicato effettuare i calcoli; per far ciò verrà introdotto quindi un nuovo metodo.

4.2 Metodo Monte Carlo

Il metodo Monte Carlo consiste nel simulare più volte la situazione desiderata, in modo da arrivare a un valore sperimentale vicino a quello teorico.

Per ottenere un numero sufficiente di simulazioni non è pensabile eseguire le operazioni manualmente, per questo si ricorre a simulazioni computerizzate.

Un semplice esempio del metodo Monte Carlo è quello di calcolare l'area di un lago di dimensioni sconosciute: conoscendo però l'area del rettangolo o quadrato all'interno del quale il lago è compreso, viene chiesto di sparare colpi di cannone in modo da coprire tutta l'area del rettangolo/quadrato, nel modo più uniforme possibile.
Contando il numero di colpi sparati e quello di colpi finiti nel lago, è possibile ottenere una stima delle dimensioni del lago, in quanto il rapporto tra superficie del quadrato o rettangolo e quella del lago è uguale al rapporto tra palle di cannone tirate e quelle finite nel lago.



(Immagine presa da wikipedia)

Altro esempio interessante è quello del calcolo di π sperimentalmente, utilizzando un metodo simile a quello appena illustrato:

Si prende, su un piano cartesiano, una circonferenza di raggio 1 e centrata nell'origine: l'equazione di tale circonferenza è $y^2 + x^2 = 1$.
Si prendono ora N punti a caso compresi nel quadrato in cui è inscritta la circonferenza e si conta quanti di questi punti ricadono all'interno della circonferenza (che quindi verificano la disequazione $y^2 + x^2 \leq 1$). Assumiamo che X punti ricadono all'interno della circonferenza.

Facendo il rapporto:

$$\frac{\text{Superficie}_{\text{quadrato}}}{\text{Superficie}_{\text{cerchio}}} = \frac{N}{X}$$

È possibile trovare un'approssimazione dell'area del cerchio, che, essendo definita come

$$A = \pi r^2$$

ci consente di trovare un'approssimazione di π.

Per evitare di eseguire tutto il lavoro manualmente, ho scritto un piccolo sorgente in C che svolge il lavoro, e restituisce in output l'approssimazione di π ottenuta:

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main (int argc, char *argv[]) {
	int i,in;
	float x,y;
	srand (time(NULL));
	if (argc<2 || !atoi(argv[1])) {
		printf ("Utilizzo: %s <numero di punti>\n",argv[0]);
		return 0;
	}

	for (i=0,in=0;i<atoi(argv[1]);i++) {
		x=(float)rand()/(float)RAND_MAX*((rand()%2)?1:-1);
		y=(float)rand()/(float)RAND_MAX*((rand()%2)?1:-1);
		if (pow(x,2)+pow(y,2)<=1) {
			in++;
		}
	}
	printf ("%f\n",(in*4.0)/atoi(argv[1]));
}
```

Effettuando il calcolo con 1 milione di punti (un numero abbastanza basso che consente tempi di esecuzione molto brevi) si ottiene circa 3.142496 come valore di π: ovviamente non è molto approssimato, ma comunque già si avvicina, considerando il basso numero di ripetizioni effettuate.

È possibile, ovviamente, utilizzare il metodo Monte Carlo anche per *Monopoly*: il seguente codice sorgente è una simulazione del gioco utilizzando tabellone e carte Imprevisti/Probabilità classici. Per quanto questa simulazione possa assomigliare a una partita vera, bisogna tenere in considerazione il fatto che, per certi versi, sarà un po' diversa: nella simulazione, per esempio, viene assunto che l'utente pagherà sempre subito la cauzione per uscire di Prigione, strategia che all'inizio del gioco viene spesso utilizzata dai giocatori ma che, con l'avanzare dello stesso, comincia ad essere accantonata a favore della tecnica del “restare in Prigione il più tempo possibile per evitare di

passare sui terreni degli altri giocatori". Inoltre, il modo in cui vengono pescate le carte dai mazzi Imprevisti e Probabilità è casuale, mentre in alcuni casi, per velocizzare le operazioni, nel gioco reale la carta appena pescata da uno dei due mazzi viene messa al fondo del mazzo stesso, cambiando quindi significativamente le probabilità di pescare una certa carta.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main () {
	srand(time(NULL));
	int i,pos,r[2],n,db;
	int passaggi[40];
	/*
	 * 0: casella "normale"
	 * 1: casella "Probabilita'"
	 * 2: casella "Imprevisti"
	 * 3: caselle "In Prigione"
	 * 4: casella "Compagnia"
	 * 5: casella "Stazione"
	 */
	int caselle[40]={
			 0,0,1,0,0,5,0,2,0,0,
			 0,0,4,0,0,5,0,1,0,0,
			 0,0,2,0,0,5,0,0,4,0,
			 3,0,0,1,0,5,2,0,0,0
			};

	// numero di lanci dei dadi
	n=100000;
	memset (passaggi,0,40*sizeof(int));
	for (i=0,pos=0,db=0;i<n;i++) {
		r[0]=(rand()%6)+1;
		r[1]=(rand()%6)+1;
		if (r[0]==r[1]) {
			db++;
			if (db==3) {
				// L'utente ha tirato
				// 3 doppi, viene
				// mandato in Prigione
				pos=10;
				db=0;
				goto b;
			}
		}
		else {
			db=0;
		}
		pos+=r[0]+r[1];
		if (pos>39) {
			pos%=40;
		}
		lab:
		switch (caselle[pos]) {
			case 1:
				// Probabilità, l'utente
				// potrebbe essere mandato
				// su un'altra casella
				switch (rand()%17) {
					case 0:
```

```
                        // Va al via
                        pos=0;
                        break;
                  case 1:
                        // Va in prigione
                        pos=10;
                        break;
                  default:
                        // Nulla cambia
                        // la sua posizione
                        break;
            }
            break;
      case 2:
            // Imprevisti, l'utente ha 10 possibilità
            // su 17 di cambiare posizione
            switch (rand()%17) {
                  case 0:
                        // Va al Via
                        pos=0;
                        break;
                  case 1:
                        // Trafalgar Square
                        pos=24;
                        break;
                  case 2:
                        // Prossima compagnia
                        for (;caselle[pos]!=4;pos++) {
                              if (pos==40) {
                                    pos=0;
                              }
                        }
                        break;
                  case 3:
                  case 4:
                        // Ci son 2 tessere per
                        // avanzare alla prossima
                        // stazione
                        for (;caselle[pos]!=5;pos++) {
                              if (pos==40) {
                                    pos=0;
                              }
                        }
                        break;
                  case 5:
                        // Pall Mall
                        pos=11;
                        break;
                  case 6:
                        // Indietro di tre
                        pos-=3;
                        // Viene rimandato indietro
                        // di 3, quindi viene rifatto
                        // il calcolo nel caso in cui
                        // l'utente finisce su un altra
                        // probabilità o imprevisti
                        goto lab;
                        break;
                  case 7:
                        // Prigione
                        pos=10;
```

```
                                    break;
                              case 8:
                                    // Kings Kross
                                    pos=5;
                                    break;
                              case 9:
                                    // Mayfair
                                    pos=39;
                                    break;
                              default:
                                    // Niente
                                    break;
                        }
                        break;
                  case 3:
                        // In Prigione
                        pos=10;
                        break;
                  default:
                        break;
            }
            b:
            passaggi[pos]++;
      }
      for (i=0;i<40;i++) {
            printf ("%d\t->\t%.2f %%\n",i,passaggi[i]*100.0/n);
      }
      return 0;
}
```

Facendo eseguire 100,000 tiri di dadi, si ottengono le seguenti percentuali:

```
0     ->    2.99 % Via
1     ->    2.12 % Terreno
2     ->    1.93 % Probabilità
3     ->    2.17 % Terreno
4     ->    2.28 % Tasse
5     ->    2.87 % Stazione
6     ->    2.27 % Terreno
7     ->    0.94 % Imprevisti
8     ->    2.31 % Terreno
9     ->    2.31 % Terreno
10    ->    6.31 % Prigione
11    ->    2.69 % Terreno
12    ->    2.63 % Compagnia
13    ->    2.44 % Terreno
14    ->    2.42 % Terreno
15    ->    2.89 % Stazione
16    ->    2.88 % Terreno
17    ->    2.52 % Probabilità
18    ->    3.00 % Terreno
19    ->    2.99 % Terreno
20    ->    2.89 % Parcheggio gratuito
21    ->    2.81 % Terreno
22    ->    1.13 % Imprevisti
23    ->    2.83 % Terreno
24    ->    3.06 % Terreno
25    ->    3.06 % Stazione
26    ->    2.69 % Terreno
27    ->    2.75 % Terreno
28    ->    2.84 % Compagnia
```

```
29    ->    2.57 % Terreno
30    ->    0.00 % In Prigione
31    ->    2.67 % Terreno
32    ->    2.57 % Terreno
33    ->    2.32 % Probabilità
34    ->    2.58 % Terreno
35    ->    2.40 % Stazione
36    ->    0.97 % Imprevisti
37    ->    2.20 % Terreno
38    ->    2.11 % Tassa
39    ->    2.59 % Terreno
```

Queste sono le percentuali con cui ogni casella viene visitata (al fianco di ogni percentuale è stata messa un'indicazione del tipo di casella): è possibile osservare alcune cose interessanti:

- La casella con più passaggi è la Prigione: il che è abbastanza ovvio, in quanto vi sono diversi modi di finire in Prigione (pescando una carta Probabilità o Imprevisti, facendo 3 “doppi” consecutivi o finendo sulla casella apposita).
- La casella “In Prigione” non ha alcun passaggio: questo perché chi finisce su quella casella viene mandato direttamente in Prigione.
- Gli Arancioni e i Rossi sono i gruppi di terreni che vengono visitati più frequentemente, proprio per il fatto che si trovano poco dopo la Prigione.
- Gli Imprevisti ricevono poche visite dato che, sulle 17 tessere, 9 portano a qualche altra casella.

Alla luce di questa e di altre osservazioni che si possono fare partendo dai risultati ottenuti, è possibile disegnare una strategia ottimale che non consiste nel appropriarsi dei terreni più costosi, ma di quelli più visitati.

5. *MonopoLinux*

5.1 Cos'è?

MonopoLinux è una versione open source di *Monopoly*.
Il motivo principale per cui ho deciso di sviluppare tale gioco è sicuramente il mio interesse nei confronti di tale gioco da tavolo, che ha caratterizzato diverso tempo della mia infanzia e degli ultimi anni.
Nonostante l'esistenza di diverse altre versioni di *Monopoly* per Linux, ho deciso di crearne comunque una nuova: per essa avevo un solo punto da seguire, ossia consentire all'utente un'elevata possibilità di personalizzazione del gioco e, finora, credo di aver raggiunto questo scopo.
MonopoLinux, infatti, non si caratterizza di certo per la sua grafica avanzata, ma più per quanto un utente medio può personalizzarlo senza troppi sforzi.
Il gioco è stato scritto in lingua inglese, sia per renderlo accessibile ad un'utenza maggiore, sia per mantenere i nomi del gioco in lingua originale, sia perché ultimamente mi è capitato di giocare spesso al gioco in inglese, imparando così i diversi termini del gioco originale.

5.2 Strumenti scelti

Per lo sviluppo di un software è necessario scegliere alcuni strumenti da utilizzare per la programmazione. Come sistema operativo ho scelto Linux, come linguaggio di programmazione il C e come librerie grafiche le SDL.

5.2.1 Linux

Ho scelto Linux in quanto si tratta del sistema operativo che uso quotidianamente da più di 6 anni e che fornisce al programmatore tutti gli strumenti necessari, come editor di testo, compilatori, debugger e quant'altro; cose che, sugli altri sistemi operativi, vanno ricercati e scaricati, spesso anche a pagamento.
MonopoLinux, inoltre, condivide la stessa filosofia open source di GNU/Linux, che quindi si presta perfettamente come sistema operativo per lo sviluppo anche da un punto di vista “etico”.

5.2.2 C

Ho scelto di programmare il gioco in C per lo stesso motivo per cui ho scelto Linux, ossia perché son più di 6 anni che lo utilizzo come linguaggio principale; il che mi ha portato ad avere una conoscenza molto buona dello stesso, più che sufficiente a consentirmi di sviluppare questo gioco senza incontrare difficoltà dettate da una carenza di conoscenze; cosa che sarebbe successa se avessi sviluppato il gioco in Java, che pur offrendo la possibilità di lavorare con la grafica ad un livello più astratto, non è un linguaggio che ho avuto, finora, l'occasione di approfondire.

5.2.3 SDL

Le librerie SDL (Simple Directmedia Layer) sono librerie grafiche multipiattaforma, e già questo è un ottimo motivo per utilizzarle, ma il motivo per cui le ho scelte non è solo questo: da quando mi sono state mostrate da un amico, sono rimasto stupito della semplicità con cui possono essere usate, e, anche se mi son sempre limitato a piccoli lavori, le ho sempre trovate molto funzionali.

Consentono inoltre di lavorare a basso livello e di gestire direttamente i pixel sullo schermo, cosa che la maggior parte delle librerie grafiche ad alto livello non consentono, almeno in modo "diretto".
Quindi questo è il motivo principale per cui ho deciso di utilizzare le SDL, per poter gestire la grafica a basso livello: quasi sempre (tranne per la gestione delle scritte), sviluppando *MonopoLinux*, ho gestito direttamente i pixel della finestra, creando alcune basilari funzioni per velocizzare il lavoro ma mai usando funzioni "preconfezionate".

5.3 Features desiderate

Per questo gioco, come già detto, ho deciso di puntare sulla personalizzazione; tutte le features caratteristiche, quindi, sono basate sulla possibilità, da parte dell'utente, di modificare:

- Tabellone di gioco
- Immagine e ordine delle caselle
- Disposizione e caratteristiche delle proprietà
- Pedine utilizzate
- Colore assegnato ad ogni giocatore (range #000000 → #FFFFFF)
- Carte di Imprevisti & Probabilità
- Impostazioni e regole del gioco

Potendo modificare queste caratteristiche, è possibile creare proprie versioni di *Monopoly* molto diverse dall'originale, cosa impossibile con le altre versioni del gioco, originali o meno.
Nonostante l'alto livello di personalizzazione, alcune limitazioni esistono comunque: è possibile giocare "solo" fino a 8 giocatori (il videogioco ufficiale ne consente 4), vi sono limiti di case/alberghi costruibili (fino a 8 case e 4 alberghi) e altro: ciò è dovuto ai limiti "grafici"; avere, per esempio, 20 giocatori sarebbe, per com'è scritto il gioco, possibile, ma le pedine non potrebbero essere visualizzate a causa del ridotto spazio disponibile.
La grafica del gioco non è uno dei punti forti: come anticipato, sono state utilizzate le librerie SDL e una gestione a basso livello, quindi quanto ottenuto non è nemmeno lontanamente paragonabile alle sfarzose versioni originali: queste ultime hanno una grafica 3D discretamente avanzata, mentre quella offerta da *MonopoLinux* (ma anche da tutte le altre versioni non ufficiali per Linux) è una semplice grafica 2D con vista dall'alto.

5.4 Situazione attuale

Il gioco, attualmente si trova in una versione beta già giocabile, ma carente di una parte che consente di giocare solo "a metà": non ho ancora iniziato a sviluppare l'intelligenza artificiale, il che rende il gioco utilizzabile solo tra umani.
Il gioco, per ora, non presenta bug evidenti (quelli scoperti sono stati corretti), ma essendo io l'unico beta tester è probabile che, una volta utilizzato anche da altri, i bug comincino ad essere scoperti.

5.5 Sviluppo

Di seguito sono spiegate alcune delle parti fondamentali del gioco, illustrando come sono state create e come funzionano, più alcune altre informazioni.

5.5.1 Struttura player

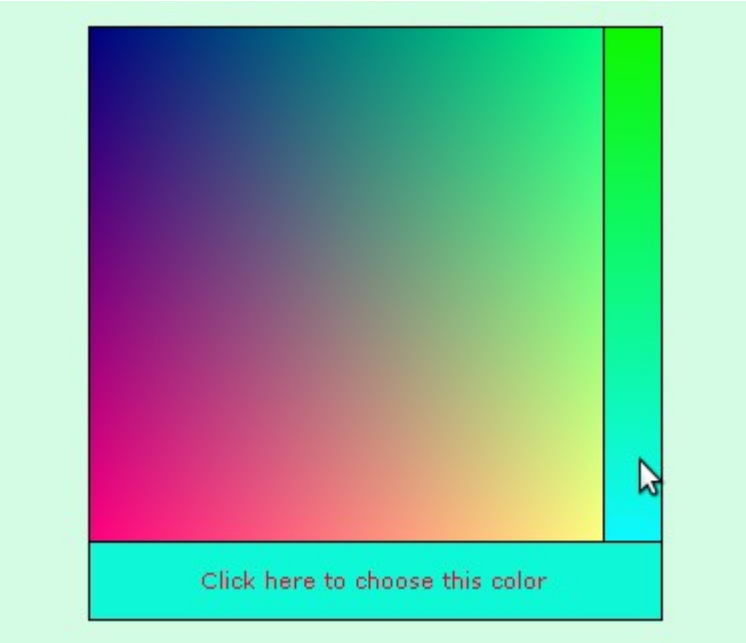
All'inizio del gioco, viene chiesto inserire le impostazioni di ogni utente tramite la seguente schermata.

Ogni giocatore abilitato sarà quindi descritto dalla struttura player, così definita:

```
struct player {
      int id;
      int token;
      int ai;
      int pos;
      int money;
      int prop[28][2];
      int outpr;
      int inprison;
      int bankrupt;
      int oweto;
};
```

- "id" contiene l'id del giocatore, che sarà semplicemente un numero identificativo compreso tra 0 e NUMERO_GIOCATORI-1.
- "token" contiene il colore (in HEX 24 bit) della pedina del giocatore. Il colore può essere deciso clickando sul riquadro colorato al fianco di ogni giocatore nella schermata mostrata in precedenza. L'utente potrà scegliere quindi il colore desiderato.
  Dovevo quindi illustrare tutti i colori a 24 bit in modo semplice sia per il giocatore (chiedere di inserire il codice esadecimale direttamente sarebbe stato troppo complicato per l'utente medio), sia per me, in quanto utilizzare i metodi usati, per esempio, nei programmi di grafica, sarebbe risultato complicato sia da realizzare, sia poi da leggere il valore

selezionato. Così, ho creato un quadrato di lato 255 pixel, variando i valori R e G all'interno dello stesso, tenendo B costante (a 0x7F) e colorando il pixel nell'intersezione con il colore relativo agli R e G. Una volta che l'utente ha clickato nell'area del colore interessato, a lato vengono mostrate tutte le sfumature del colore selezionato, variando il parametro B che prima era stato tenuto fisso.
Così il tutto è risultato molto semplice da realizzare, e risulta anche facile per l'utente selezionare il colore desiderato:



- “ai” se settato a 1, indica che il giocatore è un'Intelligenza Artificiale, altrimenti è un umano. Quest'opzione può essere scelta dalla schermata mostrata in precedenza anche se, come già detto, la parte relativa all'AI non è ancora stata ultimata, pertanto risulta inutile utilizzarla.
- “pos” indica la posizione del giocatore sul tabellone. Può assumere un valore da 0 (Via) a 39.
- “money” indica i soldi posseduti dal giocatore.
- “prop” è una matrice che serve a rappresentare le proprietà del giocatore. La matrice è di 28 righe per 2 colonne; ogni riga rappresenta una proprietà.
  Se il valore prop[n][0] è uguale a 0, ciò significa che il giocatore non possiede l'n-esima proprietà; se è a 1 significa invece che la possiede; mentre il valore -1 indica il fatto che il giocatore possiede la proprietà, ma questa è ipotecata.
  Il valore della seconda colonna, invece, rappresenta la quantità di case e alberghi costruiti su un certo terreno, quindi, nel caso di stazioni e compagnie, questo campo verrà ignorato.
  Essendo il valore un int, esso occupa 32 bit: i 16 più significativi rappresentano il numero di hotel, mentre i 16 meno significativi, il numero di case. Se quindi un terreno ha, per esempio, questo valore impostato a 0x00030000, significa che avrà 3 hotel, mentre il valore 0x00000007 rappresenterà 7 case.
  Come già detto, per una questione grafica, è stato necessario imporre un limite di case e alberghi costruibili: possono essere messi un massimo di 8 case o 4 hotel per proprietà, anche se dalle impostazioni è possibile diminuire questi valori.
- “outpr” rappresenta il numero di carte “Esci di Prigione Gratis” l'utente possiede. Sia per motivi grafici che per somiglianza con il gioco, ogni giocatore può avere un massimo di 2 di queste carte.
- “inprison” se impostato a 0, indica che il giocatore non è in prigione, altrimenti il suo valore

rappresenta il numero di turni che il giocatore deve ancora passare in prigione.
- "bankrupt" se impostato a 1, indica che il giocatore è andato in bancarotta e quindi non partecipa più al gioco.
- "oweto" contiene l'id dell'ultimo giocatore a cui l'utente ha dato una somma di denaro. Ciò serve per quando un giocatore finisce in debito, in modo da poter ridare i soldi che vengono poi trovati a chi di diritto. In caso di bancarotta, tutti gli averi del giocatore che fallisce vengono dati al giocatore indicato da oweto.
  Se questo valore è impostato a -1, significa che l'utente deve dei soldi alla Banca.

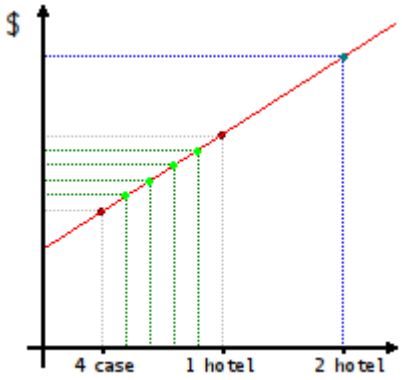
5.5.2 Struttura property

Ogni proprietà viene messa in una struttura così composta:

```
struct property {
      int id;
      char type;
      char *name;
      int *values;
      int col;
};
```

- "id" contiene l'id della proprietà (varia da 0 a 27, in quanto ci devono essere 28 proprietà all'interno del gioco (non è consentito averne di più o di meno, in quanto risulterebbe problematico)
- "type" indica il diverso tipo di proprietà e può assumere 3 diversi valori:
  'P', se la proprietà è un terreno
  'S', se la proprietà è una stazione
  'U', se la proprietà è una compagnia.
  Ogni tipo di proprietà ha caratteristiche diverse già note ai giocatori di *Monopoly*.
- "name" è una stringa che contiene il nome della proprietà.
- "values" è un puntatore ad int che viene allocato dinamicamente, e a seconda del valore di type (e quindi a seconda del tipo di proprietà), conterrà un numero di elementi diverso.
  Se la proprietà è un terreno, values conterrà 7 valori:
  0 → costo del pedaggio
  1 → costo con 1 casa
  2 → costo con 2 case
  3 → costo con 3 case
  4 → costo con 4 case
  5 → costo con 1 albergo
  6 → prezzo dell'ipoteca
  7 → costo per aggiungere 1 casa
  È evidente che il prezzo del terreno non è rappresentato; infatti questo viene ricavato dal prezzo dell'ipoteca, che è sempre la metà del prezzo del terreno.
  Un'altra cosa che può saltare all'occhio, è che viene definito il prezzo che dev'essere pagato da 1 a 4 case e con 1 albergo, ma, com'è stato detto in precedenza, è possibile aggiungere fino a 8 case o 4 alberghi.
  In questi casi, il prezzo da pagare al proprietario viene calcolato in modo molto semplice. Se c'è più di 1 albergo, vengono messi su un piano cartesiano (avente sull'asse x il numero di case e sull'asse y il prezzo da pagare) i valori da pagare per 4 case e per 1 albergo (considerato come 5 case sull'asse delle x). Viene quindi calcolata l'equazione della retta passante per questi due punti; quest'equazione viene quindi usata per calcolare il prezzo di 2

alberghi (contati come 6 case), 3 (7 case) o 4 (8 case).
Per le case, invece, il discorso si complica, in quanto bisogna fare in modo che il prezzo delle case sia sempre minore a quello dell'albergo; viene quindi presa l'equazione trovata in precedenza e viene cambiata la scala utilizzata sull'asse x: invece che considerare un albergo come 5 case, viene considerato come il numero massimo di case costruibili+1: in questo modo, la parte compresa tra 4 case e 1 albergo può essere utilizzata per dare un valore.
Per rendere il concetto più chiaro, è possibile prendere in riferimento questo grafico:



I valori di 4 case e 1 albergo sono presi per tracciare la retta in rosso. Da questa retta, poi vengono ricavati i prezzi degli altri hotel (blu) e di 5,6,7 o 8 case (in verde).
Se la proprietà è una stazione, invece, values contiene 5 valori:
0 → costo se il proprietario possiede 1 stazione
1 → costo se il proprietario possiede 2 stazioni
2 → costo se il proprietario possiede 3 stazioni
3 → costo se il proprietario possiede 4 stazioni
4 → prezzo dell'ipoteca
Anche qua, vale lo stesso discorso di prima: se il gioco prevede più di 4 stazioni, il prezzo da pagare se il proprietario possiede anche le altre viene calcolato nello stesso modo in cui vengono calcolati i prezzi degli hotel.
Se invece la proprietà è una compagnia (o società), values 3 valori:
0 → costo con una società
1 → costo con due società
2 → prezzo dell'ipoteca
In questo caso, il "costo" consiste nel numero per cui moltiplicare il valore ottenuto dai dadi, visto che le compagnie hanno questa caratteristica.
Anche qui, il numero per cui moltiplicare, se sono presenti più di 2 società, viene calcolato nello stesso modo utilizzato precedentemente.

- "col" contiene, se la proprietà si tratta di un terreno, del colore che distingue il gruppo di terreni a cui appartiene: ciò serve sia per una questione grafica che per riuscire a identificare i vari gruppi di terreni.
  Se invece la proprietà è una società, a seconda che il valore di col sia 0 o 1, significa che la società è rispettivamente elettrica o dell'acqua.

5.5.3 Struttura board

La struttura board contiene invece le informazioni riguardanti ogni casella del tabellone.

```
struct board {
      int id;
      char type;
      int v;
};
```

- “id” contiene l'id (da 0 a 39) della casella descritta.
- “type” può contenere uno dei seguenti caratteri, a seconda del tipo di casella in questione:
  'S' indica il Via (Start)
  'P' indica una proprietà
  'C' indica una casella “Probabilità” (Community Chest)
  'I' indica una casella “Tassa Patrimoniale” (Income Tax)
  'H' indica una casella “Imprevisti” (Chance)
  'J' indica la Prigione (Jail)
  'F' indica il Parcheggio Gratuito (Free Parking)
  'G' indica la casella In Prigione (Goto Jail)
  'L' indica una casella “Tassa di Lusso” (Luxury Tax o Super Tax)
- “v” viene utilizzato solo quando la casella rappresenta una proprietà, ed è utilizzata per indicare l'id (da 0 a 27) della proprietà presente in quella casella.

5.5.4 Configurazione & impostazioni

Invece che chiedere tutte le informazioni per configurare e impostare il gioco, all'inizio viene semplicemente chiesto all'utente di immettere il percorso di una cartella contenente già tutte le informazioni necessarie.
Queste informazioni sono contenute nei file:

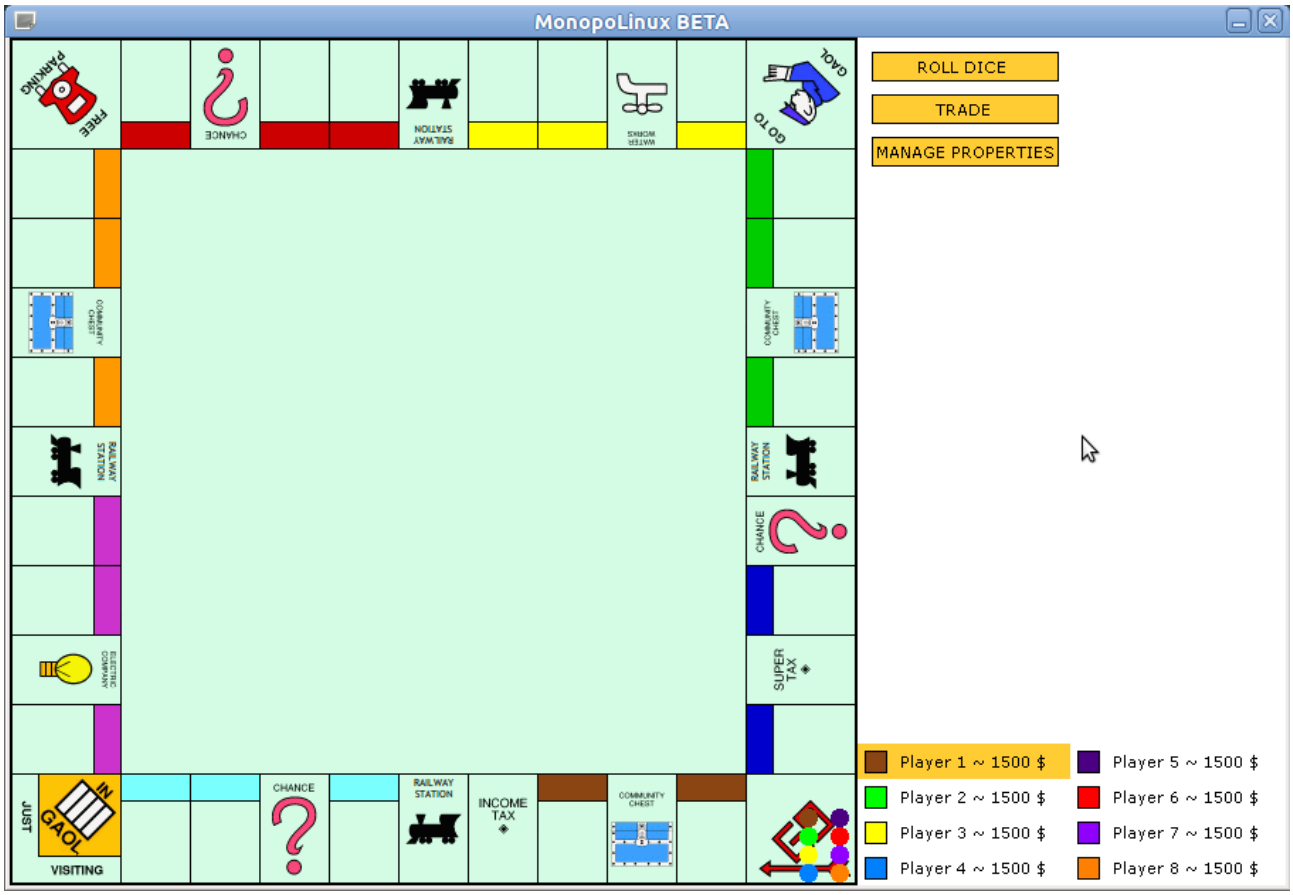
- board.txt: contiene le informazioni da inserire all'interno della struttura board, quindi come sono disposte le caselle sul tabellone.
- config.txt: contiene informazioni sulle impostazioni e regole del gioco utilizzate.
- properties.txt: contiene le informazioni sulle varie proprietà, da inserire nella struttura property.

Nella cartella d'esempio fornita con l'archivio del gioco, sono presenti questi 3 file e, basandosi sui contenuti e i commenti inseriti, è possibile creare i propri file senza difficoltà.
Oltre a questi tre file, ve ne devono essere anche altri all'interno della cartella: per esempio le immagini del tabellone, delle caselle e delle pedine:

- board.ppm: contiene il tabellone di gioco. Dev'essere di dimensioni 598x598 pixel e, ai quattro vertici del quadrato, devono essere presenti le 4 caselle angolari (Via,Prigione,Parcheggio Gratuito e In Prigione), ognuna di dimensioni 76x76 pixel.
- chance.ppm: contiene la casella degli Imprevisti. Ogni casella dovrà essere di dimensioni 48x76 pixel.
- chest.ppm: contiene la casella Probabilità.
- electric.ppm: contiene la casella della Compagnia Elettrica.
- income.ppm: contiene la casella Tassa Patrimoniale.
- prop.ppm: contiene la casella per i terreni. I pixel bianchi (#FFFFFF) di questa casella, quando rappresentati, saranno colorati del colore del terreno.
- station.ppm: contiene la casella delle stazioni.
- super.ppm: contiene la casella Tassa di Lusso.
- water.ppm: contiene la casella della Società dell'Acqua Potabile.

- token.ppm: contiene la forma della pedina di ogni giocatore. La pedina dovrà essere contenuta in un'immagine di dimensioni 22x13 pixel. I pixel bianchi (#FFFFFF) dell'immagine saranno rappresentati del colore scelto dal giocatore, mentre i pixel fucsia (#FF00FF) saranno trasparenti.
- chance/ : la directory chance contiene, al suo interno, diversi file di testo numerati da 1 in poi. Ogni file rappresenta una tessera Imprevisti. Verrà spiegato in seguito cosa dovranno contenere questi file.
- chest/ : questa directory contiene, analogamente a chance/, un insieme di file, ognuno a rappresentare una diversa carta Probabilità.

La seguente immagine mostra come il tabellone, le proprietà e le pedine vengono caricate:

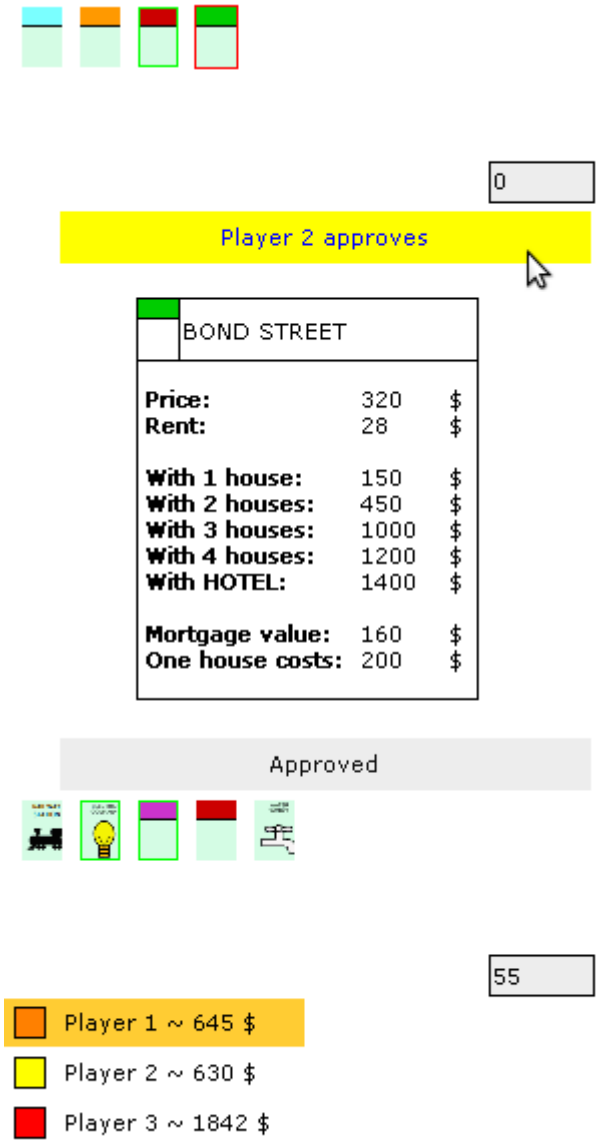


5.5.5 Menù

Il menù principale è formato da 5 pulsanti, di cui 3 sempre presenti, più 2 che compaiono solo in determinate condizioni.
Il primo pulsante può assumere 3 diversi valori:

BANKRUPT: questo pulsante compare quando il giocatore ha una quantità negativa di soldi. Se premuto, il giocatore viene dichiarato in bancarotta e viene quindi escluso dal gioco. Se invece il giocatore è in grado di trovare abbastanza soldi, non avrà bisogno di premerlo.
ROLL DICE: se l'utente non ha ancora tirato i dadi, o ha realizzato un tiro doppio in precedenza, viene mostrato questo pulsante. Quando viene premuto, i due dadi vengono lanciati.
END TURN: una volta lanciati i dadi, l'utente può effettuare le sue operazioni. Finite queste, deve premere questo pulsante per finire il turno e far iniziare il giocatore successivo.

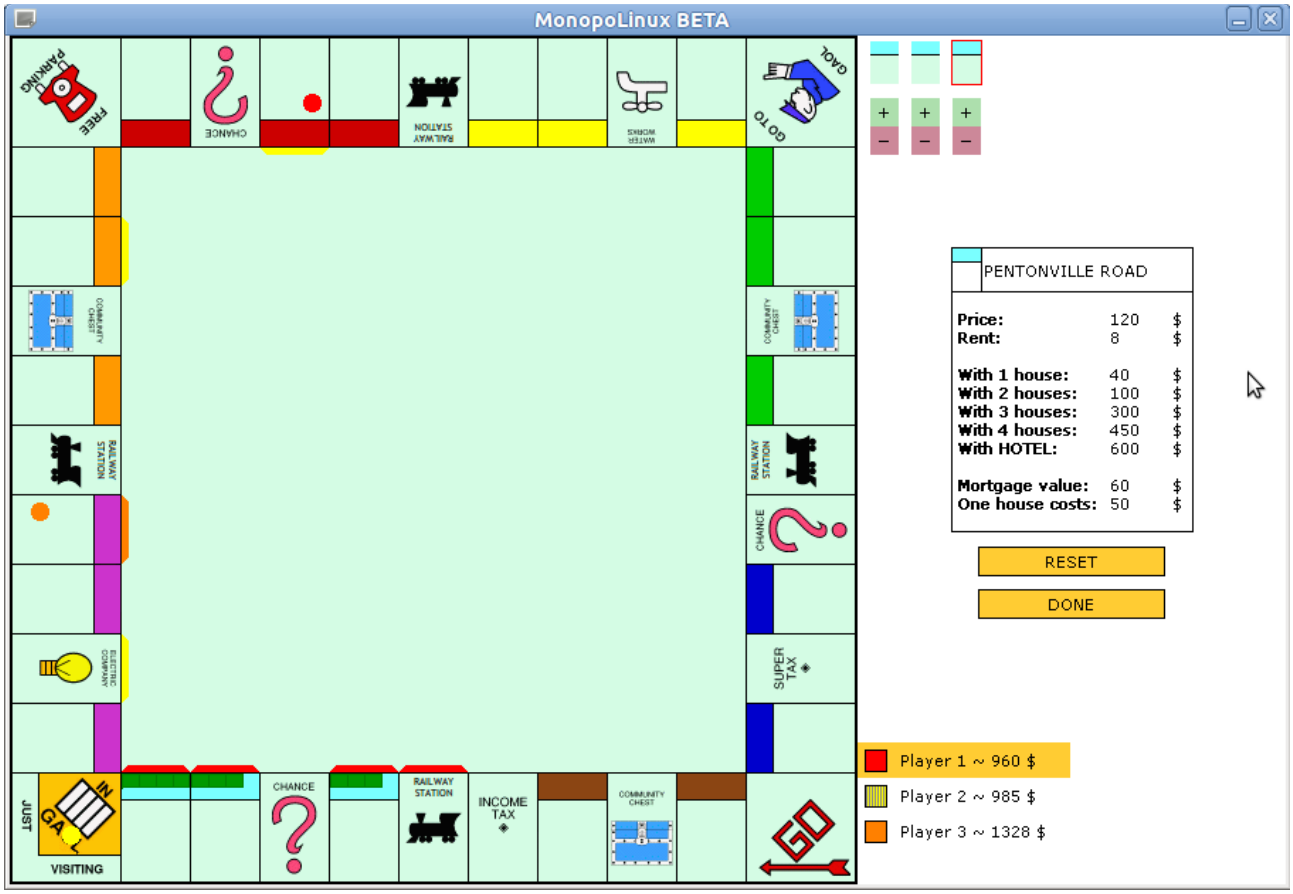
Il secondo pulsante avrà come valore TRADE, ossia viene consentito all'utente di scambiare proprietà con gli altri giocatori.
Se premuto, aprirà un ulteriore menù, che consentirà al giocatore di selezionare un altro giocatore con cui effettuare lo scambio.
Una volta selezionato, viene mostrata la vera e propria schermata di scambio, che mostrerà le proprietà di entrambi i giocatori, come mostrato nella schermata sottostante.



Nella parte superiore della schermata sono mostrate le proprietà dell'altro giocatore, mentre nella parte inferiore sono mostrate le proprie.
È possibile clickare su una delle proprietà per avere maggiori informazioni sulla stessa (in questo caso viene mostrata la tessera di Bond Street). Clickando di nuovo sulla proprietà, essa verrà selezionata per lo scambio (bordo verde).
Le due input a sfondo grigio servono invece per inserire l'importo di denaro da dare all'altro utente.
Se uno (o entrambi) i giocatori posseggono dei buoni “Esci Gratis di Prigione” (fino a un massimo di 2), essi vengono rappresentati come dei piccoli quadrati e possono essere selezionati per lo scambio.

Quando entrambi i giocatori si sono accordati, ognuno procede a clickare sulla scritta “Player X approves” per dare il proprio benestare ad effettuare lo scambio. Se un utente ha già clickato sul pulsante e delle modifiche vengono effettuate nel frattempo, l'utente dovrà clickare nuovamente sul pulsante.

Il terzo pulsante, MANAGE PROPERTIES, consente di gestire le proprietà, ipotecandole o costruendo su di esse.
Costruendo, le proprietà vengono mostrate in tempo reale sul tabellone:



Possono inoltre essere presenti, in alcune occasioni, anche altri 2 pulsanti: uno per comprare (BUY) una proprietà quando si capita su di essa. Questo pulsante può anche diventare POST BAIL (Paga cauzione) quando il giocatore si trova in Prigione.
Il quinti pulsante, USE CARD, campare quando l'utente in Prigione ha a disposizione delle carte Esci Gratis di Prigione e consente di usarne una per uscire senza pagare la cauzione.

5.5.6 Probabilità & Imprevisti

Come anticipato, le carte Probabilità e Imprevisti vengono descritte con dei file, messi nelle directory chest/ e chance/.
Ogni file contiene al suo interno le istruzioni da eseguire quando quella determinata carta viene pescata.
Per far ciò, ho scritto un linguaggio di programmazione molto basilare, in grado di consentire ad un utente di programmare letteralmente le tessere, in modo da poter realizzare le azioni desiderate.
Di seguito verrà quindi spiegato come utilizzare questo linguaggio di programmazione.

5.5.7 CPL

CPL (Card Programming Language) è un linguaggio di programmazione elementare scritto per l'occasione.
Ovviamente è molto basilare, con poche istruzioni e funzioni presenti, un utilizzo non articolato delle variabili (solo di tipo integer) che consente solo operazioni elementari, e l'utilizzo del solo if … else … , senza includere quindi parti più complesse come cicli o altro ancora.
Per convertire le istruzioni in azioni, ho scritto un piccolo interprete che, preso il file da leggere, lo processa ed effettua direttamente le azioni ordinate.
Data la presenza di poche istruzioni elementari, CPL è un linguaggio utilizzabile da chiunque.
La prima riga del file deve sempre essere la frase "scritta" sulla carta, ossia una spiegazione del contenuto delle azioni che verranno effettuate. Questa spiegazione verrà mostrata all'utente quando la carta viene pescata.
Le righe successive contengono le istruzioni che dovranno essere eseguite.

Per commentare, bisogna iniziare una riga con un #.
Le variabili sono solo di tipo intero, e non hanno bisogno di inizializzazione; esse vengono create non appena ricevono un valore:

```
a = 10
```

Una riga viene terminata dal carattere a capo (\n), nessun altro indicatore di fine riga è richiesto.
Le operazioni consentite sono quelle elementari, quindi somma, differenza, prodotto e divisione, e possono essere effettuate tra variabili, o numeri:

```
a = 10
b = a*5
c = a+b
```

Condizioni:

è possibile usare l'if … else … tipico del C:

```
if (condizione) {
… istruzioni
}
else {
… altre istruzioni
}
```

La forma utilizzata dev'essere quella appena illustrata; in quanto già

```
} else {
```

non verrà riconosciuto dall'interprete, in quanto quest'ultimo è molto basilare.
La condizione dovrà essere una comparazione tra una variabile (a sinistra) e un numero (a destra), utilizzando i seguenti operatori:

```
<< minore di
>> maggiore di
<= minore o uguale a
>= maggiore o uguale a
== uguale a
!= diverso da
```

Le funzioni presenti possono essere divise in due categorie; quelle che ritornano un valore e quelle che non ne ritornano nessuno.
Quelle che ritornano un valore devono avere il valore ritornato messo all'interno di una variabile, e sono:

```
int dice (void)
int getmoney (void)
int gethouses (void)
int gethotels (void)
int getpos (void)
```

dice () ritorna un valore casuale compreso tra 2 e 12 (risultato della somma di due numeri casuali da 1 a 6).
getmoney () ritorna la quantità di soldi posseduti dal giocatore che ha pescato la tessera.
gethouses () ritorna il numero totale di case in possesso del giocatore.
gethotels () ritorna il numero totale di hotel in possesso del giocatore.
getpos () ritorna la posizione del giocatore.

Altre funzioni, invece, eseguono solo azioni, prendendo, a volte, degli argomenti ma senza ritornare nulla.

```
void moveto (int p)
void advance (int p)
void injail (void)
void money (int n,int p)
void station (int p)
void utility (int p)
void outofjail (void)
```

moveto (int p) sposta il giocatore alla posizione p.
advance (int p) sposta il giocatore in avanti (o indietro, se p<0) di p caselle.
injail () manda il giocatore direttamente in prigione, senza passare dal Via.
money (int n, int p) aumenta i soldi del giocatore di n (o li diminuisce, se n<0). Se p vale 0, i soldi sono presi dalla (o dati alla) Banca, se invece p vale 1, i soldi verranno presi da (o dati a) ogni giocatore. Quindi, per esempio, money (10,0) darà al giocatore 10$, mentre money (-10,1) farà perdere al giocatore 10$ per ogni altro giocatore, che quindi riceverà tale cifra.
station (int p) fa avanzare il giocatore alla stazione successiva, e, se la stazione è in possesso di un altro giocatore, viene fatta pagare una somma pari a p volte quella dovuta normalmente.
utility (int p) fa avanzare il giocatore alla società successiva. Se questa è in possesso di un altro giocatore, vengono tirati 2 dadi: la somma dei due moltiplicata per p sarà la quantità di denaro da pagare al proprietario della compagnia.
outofjail () il giocatore riceve un buono “Esci di Prigione Gratis”, se ne ha meno di 2.

Queste sono le poche funzioni disponibili per CPL. Di seguito, è riportato un esempio di una possibile carta Probabilità o Imprevisti:

```
Lancia due dadi. Se la somma delle facce è minore di 6, ritira 100 volte il
 risultato realizzato, altrimenti, paga alla Banca 30 volte il valore ottenuto.
a = dice()
if (a>=6) {
      a=a*30
      a=0-a
      money(a,0)
```

```
}
else {
      a=a*100
      money(a,0)
}
```

5.5.8 Intelligenza Artificiale

Pur non avendo ancora implementato l'AI, ho già più o meno pensato ad alcuni “pilastri” su cui essa dovrebbe appoggiarsi.
Prima di tutto, l'AI cerca di studiare il tabellone di gioco, identificando (come spiegato precedentemente) le caselle su cui avviene più passaggio e prendendole in considerazione come buoni acquisti.
Quindi, ad ogni turno, viene effettuata l'analisi di quanto un terreno può risultare utile o meno all'AI e agli altri giocatori, assegnando così un “prezzo” dipendente solo in parte dal prezzo effettivo, ma anche influenzato da quanto questo terreno serve ai vari giocatori (se, per esempio, a un giocatore manca un solo terreno per completare un gruppo, il prezzo calcolato dall'AI per tale terreno sale considerevolmente), dal passaggio più o meno frequente e altri parametri minori, come i soldi in possesso, la probabilità di finirvi sopra per gli altri giocatori nel corso di 1 o 2 tiri di dadi, e altro ancora.
Viene anche assegnato un nuovo valore alla moneta, che prenderà o perderà di valore a seconda che l'AI sia, in quel turno, in possesso di più o meno denaro (se, per esempio, l'AI è in debito con un altro giocatore, sarà molto più propensa a ricevere contanti che quando invece di soldi ne ha a volontà).
L'AI usa quindi questi prezzi per determinare se fare o meno scambi con gli altri giocatori, proponendo scambi abbastanza equilibrati (in termini di prezzi calcolati).
Qualora uno scambio venisse proposto all'AI, questa la prenderà in considerazione o meno basandosi, ancora una volta, sui prezzi calcolati.

5.6 Cenni per il giocatore

Riporto ancora un paio di informazioni utili per utilizzare correttamente il gioco:
Nella prima schermata che compare viene chiesto di inserire la directory contenente le informazioni. Nell'archivio del gioco è presente la cartella “default”. Inserendo direttamente questo nome, è possibile utilizzare le regole e impostazioni del gioco classico. È possibile anche inserire il path completo della directory ma, a meno che non venga spostata per altri motivi, non vi è alcun motivo per farlo.
Quando viene chiesto di inserire i giocatori, bisogna assicurarsi che tutti siano impostati su Human: dato che l'AI non è ancora pronta, un possibile giocatore AI salterebbe sempre il suo turno, e, provando a scambiare con esso, non si riceverebbe alcuna risposta.
Durante il gioco, è possibile in ogni momento tornare alla schermata principale premendo il tasto ESC.

5.7 Conclusioni

Questo, quindi, è *MonopoLinux*, un progetto ancora in versione beta e quindi in sviluppo ma, a mio parere, con buone potenzialità.
Completata la parte relativa all'AI, vorrei rendere *MonopoLinux* giocabile in rete, creando un server apposito a cui i client si possono collegare.
Lo sviluppo dell'attuale versione del gioco ha richiesto alcuni mesi, e, nonostante i vari problemi insorti e le parti più complicate da affrontare, sono riuscito ad arrivare a rilasciare una beta abbastanza soddisfacente.

Una versione beta del gioco è scaricabile all'indirizzo:

http://darkjoker.sytes.net/MonopoLinux_BETA.rar

I sorgenti di questa versione sono abbastanza disordinati, commentati in corso d'opera, quindi delle volte di getto e non in modo abbastanza efficace.
L'idea iniziale era quella di dividere il lavoro in diversi file, ottenendone uno principale contenente la funzione main () e gli altri contenenti le varie funzioni utilizzate, raggruppate. Dato che però buona parte del codice è relativo all'interfaccia grafica, il file GUI.h contiene la maggior parte del codice. In seguito, verrà fatta una divisione più “equa” del codice.
Il gioco è stato testato solo sotto Linux, ma non dovrebbero esserci problemi ad eseguirlo sotto Windows in quanto sono state usate librerie cross-platform.
Per compilare il codice, è richiesto un compilatore e le librerie SDL:

```
$ gcc -o main main.c -lSDL -lSDL_ttf
```

Per l'esecuzione, invece, si dovrà lanciare il comando:

```
$ ./main
```

Verrà quindi avviata l'interfaccia grafica.

6. Sitografia

Storia, modelli matematici:	http://en.wikipedia.org
Get Out of Jail Free:	http://abcnews.go.com
Per la creazione del grafo, è stato utilizzato graphviz:	htttp://www.graphviz.org
Librerie grafiche utilizzate per *MonopoLinux* (SDL):	http://www.libsdl.org
Informazioni più dettagliate sulla storia di Monopoly:	http://www.adena.com/adena/mo/
Il logo e il gioco Monopoly son di proprietà della Hasbro:	http://www.hasbro.com/monopoly/