ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA
SCUOLA DI SCIENZE

CORSO DI STUDI IN
INGEGNERIA E SCIENZE INFORMATICHE

Tangram ed edutainment per l'apprendimento della geometria

Relazione Finale in

Sistemi Multimediali

Relatore

Dott.ssa Silvia Mirri

Presentata da

Raffaele Gori

Sessione II°
Anno Accademico 2014/2015

Alla mia famiglia e agli amici, ma soprattutto a mio fratello.

Indice

INDICE ..... **I**

INTRODUZIONE **3**

1 ICT ED ELEMENTI DI EDUTAINMENT NELL'AMBITO DELL'APPRENDIMENTO **7**

1.1 RELAZIONE TRA GIOCO E APPRENDIMENTO 7
1.1.1 Per la matematica / Risoluzione di problemi 8
1.2 VIDEOGIOCO COME NUOVO TIPO DI MEDIUM PER L'APPRENDIMENTO 10
1.2.1 Gamification e Edutainment 12
1.3 STATO DELL'ARTE 14
1.3.1 Videogiochi matematici notevoli per web e piattoforme mobili. 14
1.3.2 Differenze nella programmazione di app native, web app e app ibride 18
1.3.3 ICT e scuola in Italia 21

2 SPECIFICHE DEL PROGETTO E STRUMENTI DI SVILUPPO UTILIZZATI **27**

2.1 PRESENTAZIONE E CONTESTUALIZZAZIONE DEL PROGETTO DI TESI 27
2.1.1 Contesto in cui si inserisce il progetto di tesi 27
2.1.2 Il ruolo del Tangram 28
2.2 GESTIONE DEL CONTESTO DI GIOCO 30
2.3 TECNOLOGIE E METODI DI PROGETTAZIONE E SVILUPPO UTILIZZATI PER LA REALIZZAZIONE DEL PROGETTO 31
2.3.1 HTML5. 32
2.3.2 CSS3 35
2.3.3 JavaScript 39
2.3.3 KineticJS e Canvas 44
2.3.3 Apache Cordova 46
2.4 GESTIONE DI TEMI E AMBIENTAZIONI DI GIOCO MULTIPLI 48

3 IMPLEMENTAZIONE E FASE DI TEST **51**

3.1 ARCHITETTURA DELL'APPLICAZIONE 51
3.1.1 Analisi nel dettaglio dei singoli blocchi 53
3.2 FASI DI GIOCO E CASI D'USO 74
3.3 TEST EFFETTUATI, BUG RISCONTRATI, SOLUZIONI APPORTATE E MIGLIORAMENTI POSSIBILI 78

CONCLUSIONI.....85
BIBLIOGRAFIA I

Introduzione

Da lungo tempo è noto che esiste un legame forte tra *apprendimento* e *divertimento*, ambiti che nascono come tutt'altro che privi di interazioni, nonostante l'idea generale che si ha della scuola come luogo grigio e tedioso sia ben lontana dall'essere piacevole e divertente: basti pensare al fatto che quello del gioco è il principale mezzo a disposizione dei bambini in età prescolare per imparare a conoscere le dinamiche di base dell'ambiente che li circonda, e tuttavia in questo periodo essi imparano moltissimo in un arco di tempo piuttosto breve, e senza mai mostrare fatica o noia.

E' opinione di molti studiosi che anche superata la fase iniziale della vita il gioco possa continuare a costituire un validissimo sostegno all'apprendimento in ogni campo e disciplina, poiché il divertimento che si trae da esso porta con sé elevati livelli di *motivazione.*

Inoltre la presentazione di obiettivi chiari e di continui feedback mantiene alta l'attenzione del giocatore/discente-inconsapevole e lo sprona senza gravarlo di imposizioni, facendo leva su meccanismi psicologici quali il *bisogno di gratificazione* o la *competitività*, ad applicarsi per superare gli ostacoli che incontra, riducendo la percezione della fatica.

Il concetto di *gamification* (che deriva dall'inglese *gamify* ed indica l'operazione di *rendere un gioco* qualcosa che non nasce necessariamente per esserlo) nasce, in epoca recente, dall'applicazione di dinamiche di gioco a contesti non prettamente ludici, al fine di veicolare determinati messaggi o perseguire determinati scopi.

Oggi le sue applicazioni coinvolgono moltissimi ambienti e contesti, che spaziano dall'incentivazione della produzione e della motivazione tra i dipendenti di un'azienda,

alla fidelizzazione dei clienti di un negozio o di uno store virtuale, fino anche alla spinta al raggiungimento di determinati obiettivi in termini di forma fisica.

All'applicazione dei concetti di gamification all'ambiente scolastico ci si riferisce con il termine più specifico di *edutainment*, termine che nasce dalla fusione delle due parole inglesi *eduactional* (educativo) ed *entertainment* (intrattenimento), che indica, appunto, l'inserimento di dinamiche di gioco all'interno di un contesto di insegnamento, al fine di veicolare concetti e sviluppare abilità e conoscenze in maniera piu' piacevole e produttiva di quanto avvenisse con il metodo classico, basato su cicli di *spiegazione*, *studio individuale* e *verifica dell'apprendimento*, che mostrano i propri limiti nel momento in cui gli studenti non siano in grado di mettere in atto efficaci strategie di memorizzazione ed interiorizzazione dei concetti, o non si trovino ad essere sufficientemente motivati nella loro azione di discenti.

I *videogiochi*, nell'era digitale, stanno assumendo un ruolo importantissimo nel contesto dell'insegnamento attraverso il gioco: essi si dimostrano un ottimo mezzo di veicolazione del sapere e costituiscono il primo medium veramente nato in ambito multimediale, quindi il più adatto per l'uso di pacchetti di questo tipo che coinvolgano più canali di assimilazione.

In questo contesto si inserisce il progetto sviluppato in questa tesi, che consiste in una applicazione per dispositivi mobili destinata ad essere innestata, come modulo di gestione di un piccolo videogioco, all'interno di un sistema più ampio che applica i principi di gamification all'apprendimento della matematica.

In particolare in questa sede ci si è occupati del modulo relativo alla familiarizzazione con alcuni concetti di geometria tramite la digitalizzazione dell'antico gioco cinese del Tangram.

Gli obiettivi del progetto sono quelli di fornire una applicazione di facile utilizzo che non risulti noiosa dopo poche partite e garantire, nel contempo, tutte le caratteristiche necessarie per una sua facile integrazione come componente di un sistema più complesso.

A tal fine elementi cui si è prestata particolare attenzione nella progettazione sono la possibilità di gestione di temi multipli tramite la parametrizzazione di ogni aspetto legato alla presentazione, e la strutturazione del programma in maniera tale da risultare un modulo facilmente isolabile, fornendo una interfaccia di comunicazione con il sistema di livello superiore limitata allo scambio di pochi e specifici dati di interesse.

Questo volume di tesi è strutturato come segue:

- Nel primo capitolo ci si è occupati dell'analisi delle relazioni esistenti tra il gioco e l'apprendimento intesi dapprima in senso lato e calati, poi, nell'ambito della materia di interesse: la matematica. Si è affrontato, inoltre, il tema della validità del videogioco come un ottimo mezzo multimediale per la veicolazione della conoscenza in età digitale, e si sono introdotti contestualmente i concetti di gamification ed edutainment. La sezione successiva affronta un excursus sullo stato dell'arte, in cui si presentano alcune tra le principali applicazioni di tipologia simile a quella del software prodotto, si illustrano le diverse tipologie di approccio all'implementazione di applicazioni destinate all'esecuzione su dispositivi mobili e si ripercorre brevemente la storia dell'inserimento e dell'integrazione, mai completamente avvenuta, delle *ICT* (*Information and Communications Technologies*, Tecnologie dell'Informazione e della Comunicazione) nella scuola italiana dagli anni '80 fino ai giorni nostri.
- Il secondo capitolo è dedicato alla presentazione delle specifiche e ad una breve contestualizzazione del progetto sviluppato in questa tesi, di cui vengono mostrate anche le dinamiche di gestione del contesto di gioco e del supporto a temi multipli. In questo capitolo vengono inoltre illustratele tecnologie e gli strumenti utilizzati per la realizzazione dell'applicazione, cercando di fornire un livello di dettaglio soddisfacente, senza però eccedere in particolari che non risulterebbero utili ai fini della presentazione.
- Il terzo ed ultimo capitolo del volume di tesi consiste in una analisi nel dettaglio dell'architettura e dei singoli moduli del sistema implementato, cui si aggiunge la presentazione delle fasi di gioco che caratterizzano una partita. Di tali fasi si sono forniti, oltre ai diagrammi UML di stato e casi d'uso, anche snapshot relativi ai vari momenti pregnanti. A completare il capitolo c'è, infine, una sezione dedicata al testing e alla contestuale presentazione degli elementi di criticità evidenziati in questa fase, corredata da quella delle soluzioni proposte o suggerite come spunto per eventuali miglioramenti ed implementazioni futuri.

1 ICT ed elementi di Edutainment nell'ambito dell'apprendimento

In questo capitolo si vuole introdurre il lavoro svolto presentando il contesto in cui il progetto affonda le sue radici e l'idea che sta alla base dell'attenzione rivolta al software videoludico in relazione all'istruzione.

1.1 Relazione tra gioco e apprendimento

Quella che il gioco sia indissolubilmente legato all'apprendimento non è certamente un'idea nuova (tant'è che il termine *scholè* in greco originariamente indicava il tempo libero da dedicare a sé stessi, e conseguentemente il divertimento, come in latino il termine *otium*), ma per avere un esempio lampante del fatto che non può essere lontana dalla verità basta pensare che nei primi anni di vita i bambini non fanno altro che ciò che viene definito giocare, ma quello è il modo in cui imparano come funziona l'ambiente che li circonda e come esso risponde alle loro azioni, in un contesto in cui il gioco e l'apprendimento non sono distinguibili, ma certamente viene imparato moltissimo in poco tempo.

Si parla, per questo tipo di apprendimento, di apprendimento *percettivo-motorio* [CAN03], caratterizzato dell'essere in larga misura inconscio e naturale, ma fonte di una conoscenza interiorizzata, concreta e duratura.

Come meccanismo di apprendimento, tuttavia, il gioco non esaurisce la sua efficacia nei primi anni di vita, e rimane, per le sue caratteristiche, sempre un validissimo strumento.

Nella sezione in cui si espongono le motivazioni pedagogiche dell'introduzione delle nuove tecnologie (e di videogiochi educativi) nella scuola dell'obbligo, in un documento redatto nell'ambito del progetto DANT nella regione Trentino nel 2004 si legge: *"Il gioco è la mascheratura che la natura ha dato all'apprendere"* [NES04] e "*È la natura del gioco o del premio che si avrà alla fine a determinare il fatto che si farà più attenzione a come si impara.*" [NES04]

In queste due frasi si vuole esprimere il concetto che l'attività del gioco, per come si è venuta a configurare dal punto di vista evolutivo, costituisce una forma di apprendimento naturale ed inconsapevole, la cui efficacia deriva dal fatto che il divertimento che il giocatore prova mentre gioca porta in lui un alto livello di motivazione, e lo induce a dare fondo a tutte le proprie conoscenze e risorse per comprendere al meglio la realtà di gioco.

Poiché è mosso da un meccanismo di premiazione, però, ciò avviene in maniera assolutamente naturale e senza fatica.

In questo contesto sono nati progetti di insegnamento in cui alle metodiche tradizionali vengono affiancati nuovi tipi di medium, come pacchetti didattici arricchiti di testo, suoni, immagini ed animazioni, che mettono il controllo nelle mani degli studenti, e momenti destinati a giochi che richiedono agli alunni di applicare ciò che hanno appreso, riscontrando che i primi producono un maggior livello di motivazione e partecipazione, e minori livelli di ansia [YIL01] ed i secondi, fornendo uno scopo immediato allo studio delle materie hanno risultati migliorativi in termini di propensione allo studio e sviluppo di alcune abilità personali.

Conseguenza di questo è un apprendimento più solido e duraturo negli studenti.

1.1.1 Per la matematica / Risoluzione di problemi

Per quanto riguarda la matematica esiste un legame profondo tra la *risoluzione di problemi*, che sta alla base del pensiero matematico, ed i *giochi matematici*, che nella maggior parte dei casi hanno forma di problema.

Requisito fondamentale che distingue un esercizio da un problema propriamente detto è il rapporto che esiste tra le modalità di risoluzione del quesito e le conoscenze possedute dal soggetto cui il quesito è sottoposto: gli strumenti di risoluzione di un problema non sono definiti a priori, quindi per approdare alla soluzione è necessaria la costruzione in itinere di una strategia adeguata, e questo, nel caso di uno studente,

comporta la necessità di attingere con proprietà a tutte le conoscenze possedute ed individuare quali di queste possano risultare utili alla risoluzione del problema posto[BOL07].

Che vengano usati per l'introduzione di nuovi concetti a partire da quelli conosciuti, o per la verifica dell'apprendimento di concetti già introdotti, l'uso di giochi nell'ambito dell'insegnamento/apprendimento della matematica porta con sé una serie di possibilità non offerte dai metodi tradizionali.

Nell'ambito dell'introduzione di nuovi concetti, per esempio, porre agli alunni problemi non risolvibili con gli strumenti forniti loro sino a quel momento (ma comunque comprensibili avendo a disposizione tali strumenti) ed aiutarli nella risoluzione inducendone il ragionamento e guidandolo secondo un approccio maieutico porta a risultati migliori in termini di memorizzazione dei concetti e delle nozioni, poiché risulta più semplice memorizzare qualcosa che si è scoperto da sé piuttosto che qualcosa recepito in maniera passiva dall'esterno, inoltre secondo quanto sostenuto dallo psicologo e filosofo Lev Vygotskij nella sua teoria dell'apprendimento sociale e più recentemente dalla psicologia cognitivista, una situazione di interazione con gli altri può indurre uno sviluppo nell'alunno non raggiungibile altrimenti, tanto che egli definisce una Zona di Sviluppo Prossimale (ZSP) come *"la distanza tra il livello effettivo di sviluppo, così come è determinato da problem-solving autonomo, e il livello di sviluppo potenziale, così com'è determinato attraverso il problem-solving sotto la guida di un adulto o in collaborazione con i propri pari più capaci"(Vygotskij, 1934)* [MIA04] Per quanto riguarda, invece, la verifica delle competenze acquisite, una situazione di risoluzione di problemi sotto forma di gioco permette una verifica più profonda del livello di competenze raggiunto dagli studenti, perché non si limita a sondare quale livello di memorizzazione di informazioni e procedure di risoluzione per esercizi standardizzati essi hanno accumulato, ma, non ponendo la domanda sotto forma di esercizio tipico, consente di analizzare anche quanto questi concetti siano stati interiorizzati e quanto ogni studente riesca a padroneggiarli ed utilizzarli in maniera corretta.

1.2 Videogioco come nuovo tipo di medium per l'apprendimento

Parlando di giochi come utile mezzo per l'insegnamento, e calando il discorso in un ambito più prettamente collegato al progetto di tesi, si parlerà ora di videogiochi, e di come essi presentino caratteristiche particolarmente valide per l'insegnamento, soprattutto in uno scenario sempre più evoluto dal punto di vista tecnologico.

In relazione ai videogiochi esiste sempre la tendenza a relegarli al campo dell'intrattenimento ricreativo, ma la loro struttura e l'interazione che essi hanno con l'utente sono molto più complesse e potenti di quanto non si creda.

Il videogioco è potenzialmente lo strumento di apprendimento multimediale più potente oggi a nostra disposizione, poiché è l'unico che nasce direttamente in questo contesto.

In un ambiente in cui la maggior parte degli strumenti di e-learning nascono come trasposizione digitale di materiale concepito per lo stampato, e presentano una minima parte di multimedialità e di interattività, "il videogioco è il primo *medium* della simulazione a tre dimensioni: coinvolge l'occhio con le immagini, l'orecchio con la musica e soprattutto con gli effetti acustici simulati, e coinvolge […] il tatto, prolungando i nostri piedi e le nostre mani nello spazio oltre il video" [CAR96].

Molti studi hanno mostrato che l'uso di pacchetti multimediali nell'insegnamento ha effetti migliorativi in termini di motivazione generale, di coinvolgimento dei discenti, e di minore produzione di ansia. [YIL01]

Conseguenza di questo è un apprendimento più solido e duraturo da parte degli studenti, tanto che, ponendo a confronto lezioni svolte in modo tradizionale con lezioni svolte in maniera multimediale, nel 2001 Yildirim riscontrò che a fronte di nessuna differenza significativa nell'immediato, a distanza di un mese i soggetti che avevano partecipato a lezioni multimediali conservavano meglio le informazioni rispetto a quelli sottoposti alle lezioni tradizionali[CAN03].

Paivio (1986) identifica due principali canali attraverso cui l'informazione viene elaborata: uno deputato alle fonti di informazione di tipo testuale e vocale, ed uno deputato alle fonti di informazione non verbale, quali possono essere immagini e suoni.

L'apprendimento di una informazione veicolata da entrambi i canali in maniera interattiva risulta considerevolmente migliore rispetto a quello che si avvale di un canale singolo, aiutando il soggetto a costruire diversi percorsi cognitivi per il reperimento di

tale informazione. Questa condizione viene detta "*doppia elaborazione*" o "*dual processing*".

Oltre a questo il videogioco presenta anche altre caratteristiche che, se ben sfruttate, possono essere molto utili alla didattica.

La principale caratteristica di questo tipo è proprio quella più avversata da chi vede nei videogiochi più un ostacolo all'apprendimento che un potenziale strumento: l'*attrattività*, che in caso di abuso può diventare generazione di *dipendenza*.

Cangià nota che nel termine stesso *videogioco* sono già insiti gli aspetti attrattivi di audio e video tipici della televisione ed il piacere suscitato dall'aspetto ludico, con tutta la spinta motivazionale e di godimento che questo comporta.

In particolare già nel 1981, riporta Cangià, Malone, osservando l'interazione di bambini e ragazzi con i videogiochi, individua sei caratteristiche principali che conferiscono ai videogiochi potere di attrazione su chi ne usufruisce.

La prima di queste caratteristiche è la *sfida*, determinata dalla presentazione di più livelli di difficoltà, di informazioni nascoste o incomplete, benché necessarie, o da elementi di casualità.

La sfida soddisfa il bisogno di autorealizzazione e spinge al superamento di sé stessi, tanto che si è riscontrato che le situazioni di gioco che producono maggiore stimolo sono quelle più impegnative, purché ancora non insormontabili per il giocatore.

Altri elementi di attrazione sono la *curiosità*, sia in termini di ricerca di stimolazioni sensoriali piacevoli, sia come ricerca del completamento dell'informazione posseduta dal soggetto, il *controllo*, nel momento in cui il giocatore percepisce come le sue azioni e decisioni abbiano effetto sul gioco, la *presenza di elementi di fantasia*, la *competizione*/*cooperazione*, e, infine, il *riconoscimento*, offerto al giocatore tramite il punteggio raggiunto, particolari benefit o l'elogio, che soddisfano il bisogno di approvazione ed esibizione.

Un ruolo importate in termini di potere attrattivo dei videogiochi è ricoperto anche dalla dinamica interattiva, che coinvolge totalmente il giocatore in un dialogo con la macchina, dalla presentazione di obiettivi specifici e compiti da svolgere, e dalla sensazione di essere protagonisti.

Fondamentalmente il dialogo tra videogioco e giocatore avviene secondo uno schema di ripetizione, schema che ben si sposa con il tipo di apprendimento percettivo-

motorio di cui si è parlato al paragrafo 1.1 in relazione al modo di conoscere il mondo nei bambini piccoli.

La prima fase del ciclo di una tipica sessione di gioco consiste nella raccolta di informazioni da parte dell'utente, informazioni che in un secondo momento vengono analizzate al fine di prendere, da ultimo, delle decisioni, e compiere azioni che modificheranno lo stato del sistema.

Terminato questo ciclo si presenta una nuova configurazione dell'ambiente di gioco, e quindi si prosegue con l'inizio del ciclo successivo iniziandone l'analisi.

In questi cicli l'utente ha l'opportunità di affinare alcune abilità che, quand'anche il gioco non fosse strettamente finalizzato all'istruzione, sono generalmente utili anche nell'ambiente scolastico in senso lato, come lo sviluppo delle proprie capacità analitiche e la comprensione dei meccanismi di causa-effetto, allenate nella seconda fase del ciclo, ed il pensiero strategico, accompagnato alla capacità di *decision-making*, sviluppati nella fase finale del ciclo iterativo.

In definitiva quello del videogioco non si limita ad essere un mezzo di comunicazione qualsiasi, o uno strumento di divertimento fine a sé stesso, e per tanto da relegarsi all'ambito ricreativo o come premio per uno sforzo fatto, ma per le sue caratteristiche intrinseche di motivazione, feedback continuo, attrattività e stimolazione del ragionamento potrebbe ricoprire, se ben utilizzato, un ruolo importante della didattica, se non altro come ausilio complementare alle modalità canoniche di insegnamento.

"Se le migliori esperienze di apprendimento si basano sulla motivazione, sulla declinazione di obiettivi chiari, sull'interpretazione dei risultati e su feedback immediati e continui, allora i videogiochi sono eccezionali strumenti di apprendimento, perché funzionano esattamente con queste caratteristiche" [CEC12].

1.2.1 Gamification e Edutainment

Con il termine *Gamification* (dall'inglese *to gamify = rendere un gioco*) ci si riferisce all'applicazione di meccaniche ludiche a contesti che non hanno direttamente a che fare con il gioco al fine di veicolare messaggi o indurre comportamenti di vario tipo.

Si tratta di un potente mezzo per agire sui bisogni psicologici dell'individuo all'interno di un sistema modificato in maniera tale da presentare aspetti ludici ed indurlo

a modificare le proprie abitudini o a focalizzare la propria attenzione su determinati messaggi stimolandone l'interesse, il coinvolgimento e la partecipazione.

L'introduzione di concetti come punti, livelli e sfide incoraggia gli utenti alla partecipazione e alla costruzione di relazioni all'interno del gioco, motivandoli al raggiungimento di determinati obiettivi.

In relazione ai meccanismi usati dalla gamification per stimolare e guidare il comportamento dell'utente, occorre individuare due concetti fondamentali: le *meccaniche di gioco*, e le *dinamiche di gioco*.

Per quanto riguarda le meccaniche di gioco, esse costituiscono i principali strumenti effettivamente utilizzati per creare l'infrastruttura ludica da utilizzare, mentre le dinamiche di gioco esprimono i bisogni ed i comportamenti inconsci che tale infrastruttura mira a stimolare in maniera finalizzata.

Concettualmente ogni meccanica fa riferimento ad una particolare dinamica, ed in particolare le dinamiche generalmente coinvolte nella gamification sono la ricompensa, lo stato, la conquista di un risultato, l'espressione di sé e la competizione.

In questo contesto le meccaniche principalmente usate sono l'introduzione di un *punteggio* che soddisfi il bisogno di *ricompensa* dell'utente, la presenza di *livelli* che diano costanti *traguardi* da raggiungere, di *sfide*, con il compito di *motivare* l'utente e spingerlo a continuare a partecipare, la possibilità di guadagnare *beni virtuali* , per dare all'utente l'idea di poter *esprime la propria personalità*, e la redazione di *classifiche* finalizzate alla stimolazione della *competizione* tra gli utenti.

L'applicazione di processi di gamification può coinvolgere moltissimi ambienti e contesti, andando dalla incentivazione della produzione e della motivazione tra i dipendenti di una azienda, alla fidelizzazione dei clienti di un negozio o di uno store virtuale, fino anche alla spinta al raggiungimento di determinati obiettivi in termini di forma fisica.

Il campo di applicazione della gamifiation di maggiore interesse in relazione a questo progetto è senza dubbio quello dell'*Edutainment*, che mira all'inserimento di momenti di apprendimento all'interno di attività ricreative, quali giochi, programmi televisivi o film.

Il termine Edutainment nasce dalla crasi tra le parole inglesi *educational* ed *entertainment*, a voler indicare un *divertimento educativo*, e unisce le due anime di un contenuto volto tanto ad istruire quanto a divertire.

Concetto chiave è la volontà di trasmettere non solo dei contenuti, ma anche il piacere per l'imparare, così che l'individuo acquisisca una maggiore facilità di apprendimento: in ambito lavorativo questo si può tradurre in una migliore capacità di *problem solving* e di affrontare situazioni per cui non si è stati preparati: un'abilità che si differenzia dalla semplice formazione.

Esiste un grosso filone dell'E-learning che si occupa dello sviluppo di applicazioni e giochi di questo tipo, che tramite contenuti interattivi insegnano ai bambini a padroneggiare nozioni e abilità, senza uscire dalla dimensione ludica.

L’utente ha modo di muoversi in una rappresentazione di un certo tipo di problema e di esplorare le varie possibili soluzioni e alternative, costruendo un'esperienza personale di cui è soggetto in un ambiente volto a facilitare l'apprendimento.

Negli ultimi anni sono sempre più diffusi i software di Edutainment (soprattutto nelle scuole primarie), e prevalentemente si tratta di software di tipo open source, a riflettere il principio di libertà dell’istruzione.

1.3 Stato dell’arte

1.3.1 Videogiochi matematici notevoli per web e piattaforme mobili

Sulla rete si trovano molti siti che forniscono sterminati archivi di giochi online gratuiti, ed un numero paragonabile è reperibile sugli store virtuali sotto forma di applicazioni per smartphone e tablet.

In questa sezione ci concentreremo su una particolare tipologia di giochi: quelli di tipo matematico.

Si possono trovare fondamentalmente due tipologie di applicazioni e giochi di tipo matematico: quelli espressamente matematici, in cui si opera direttamente su numeri e si devono compiere calcoli, e quelli matematici in senso lato, in cui si devono combinare forme, prevedere comportamenti fisici cercando di identificare i rapporti di causa-effetto, o completare rompicapo di varia natura cercando di costruire o applicare un algoritmo risolutivo.

Alcuni titoli degni di nota appartenenti alla prima tipologia selezionati in un articolo per il sito *Android Police* sono: *Countdown Maths Game Pro*, *Math Effect*, *Quento* e *2Vars*.

Per quanto riguarda Countdown Maths Game Pro la dinamica di gioco prevede che per ogni round siano forniti come dato al giocatore sette numeri, sei dei quali sono risorse da combinare, ed il settimo è l'obiettivo da raggiungere.

Lo scopo del gioco è quello di costruire una equazione che combini i sei numeri risorsa in modo tale da ottenere il settimo usando i quattro operatori di somma, sottrazione, moltiplicazione e divisione intera (per rendere il gioco più semplice e fruibile in una fascia di età più larga, i programmatori hanno deciso di non supportare la numerazione in virgola mobile).

Un esempio banale di caso d'uso potrebbe essere: 1, 2, 3, 4, 5 : 15 → 1 + 2 + 3 + 4 + 5 = 15.

L'applicazione supporta diverse modalità di gioco, tra cui quella di *gioco libero*, che non presenta limiti di tempo o punteggio, quella *velocità*, che ha come obiettivo quello di completare un certo numero di quiz nel minore tempo possibile, quella *a tempo*, in cui si deve completare il maggior numero di quiz possibile in un determinato lasso di tempo, ed una modalità *multi-giocatore*, in cui si gioca la stessa partita su due dispositivi differenti e vince chi termina per primo o conquista il punteggio maggiore.

Passando al secondo gioco proposto, si introduce, ora, Math Effect, che su Play Store si presenta come un gioco per lo sviluppo del proprio QI ispirato ad una metodologia di allenamento della mente di origine giapponese [KID15].

La dinamica di gioco consiste nella presentazione al giocatore di una serie di equazioni risolte dal gioco: il giocatore ha il compito di stabilire se le soluzioni fornite per le equazioni proposte sono corrette o meno.

Anche per questo gioco sono previste le modalità *a tempo*, e di *velocità*.

Si passa, ora, a Quento: un puzzle molto divertente che presenta una scacchiera 3x3 popolata da 5 cifre intervallate da operatori di somma e sottrazione.

Scopo del gioco è quello di navigare la scacchiera partendo da una cifra ed intervallando le cifre agli operatori per ottenere una equazione la cui soluzione corrisponda ad un numero obiettivo mostrato sopra la tabella.

Ogni livello presenta tre quiz, ed il giocatore ottiene stelle/punti per ogni quiz risolto.

2Vars consiste in equazioni a due operandi di cui il giocatore ha solo l'operatore e la soluzione, e due set di valori multipli tra cui scegliere per compilare i campi vuoti.

Risolto un quiz si progredisce di livello e la difficoltà aumenta.

Ad inizio partita un contatore dei tentativi rimasti è settato a 3, e quando il giocatore commette errori esso viene decrementato: il gioco finisce quando il contatore raggiunge lo zero.

Passando alla seconda categoria di giochi matematici non espressamente concepiti per l'insegnamento, titoli di interesse sono *2048*, *Sudoku*, *Balls on my screen*, *Tangram*, e *Tetris*, il gioco più famoso al mondo.

All'inizio di una partita a 2048 al giocatore viene presentata una tabella 4x4 popolata da alcune tessere contenenti numeri multipli di 2.

Quando vengono a contatto, se contengono lo stesso numero, queste tessere si uniscono a due a due in singole tessere contenenti numero doppio.

L'unico mezzo in mano al giocatore per unire le tessere è quello di far collassare tutto il contenuto della griglia verso l'alto, verso il basso, verso sinistra o verso destra, allo stesso modo in cui si comporterebbe spostando la direzione della gravità nel gioco.

Ogni volta che delle tessere si uniscono ne compaiono di nuove, e lo scopo del gioco è quello di riuscire ad ottenere il numero 2048 prima dell'inevitabile saturazione della griglia, raggiunta la quale la partita ha termine.

Si passa, ora, alla presentazione del Sudoku, gioco derivato dal *quadrato latino* di Eulero, a sua volta caso particolare di *quadrato magico*.

Quello del Sudoku è uno dei rompicapo più famosi degli ultimi anni e spopola in una infinità di forme e varianti, perciò non si è ritenuto opportuno fare riferimento ad una versione particolare quanto, piuttosto di parlare dei principi di funzionamento generali.

Al giocatore viene fornita una matrice 9x9 parzialmente popolata (possono essere precompilate dalle 20 alle 35 celle sulle 81 totali) suddivisa in 9 sottomatrici 3x3 dette *regioni*, ed il suo obiettivo è quello di completarla coerentemente con quanto contenuto inizialmente, fornendo una soluzione tale che ogni riga, ogni colonna ed ogni riquadro contengano tutte le 9 cifre, condizione che implica l'impossibilità che si presentino ripetizioni della stessa cifra all'interno degli stessi colonna, riga o riquadro.

Esistono, come detto, diverse varianti di questo gioco, che ne alterano la forma, il numero delle celle (e quindi delle cifre inseribili) ed in generale la difficoltà, ma verranno qui considerate varianti sul tema, e per tanto non affrontate.

Parlando, ora, di Balls on my screen, se ne presenta il semplice principio di funzionamento: la partita ha inizio con una matrice 5x5 di punti colorati contenenti un numero, e scopo del giocatore è quello di individuare ed evidenziare percorsi che colleghino un quantitativo di punti contenenti lo stesso numero pari al numero da loro contenuto.

Le modalità di gioco sono quella *classica*, in cui si ha a disposizione un minuto di tempo per collegare quanti più punti contenenti lo stesso numero possibili (se si verifica una situazione di mancanza di mosse disponibili si può trascinare i punti con numero 1 per aggiungere o sottrarre una unità al numero contenuto in un'altra cella), la modalità *arcade*, in cui si ha un contatore per il tempo residuo che decrementa sempre più rapidamente e per ogni combinazione si guadagna tempo, e la modalità *zen*, che non prevede limiti temporali per la partita e permette di salvare le modifiche.

Per concludere la sezione, si passa, ora, al Tetris, rimandando al paragrafo 2.1.2 per una trattazione del gioco del Tangram più dettagliata di quanto non si sarebbe potuto fare in questa sede.

Il nome Tetris deriva da *tetramino*, termine con cui si identifica una figura piana composta da quattro quadrati interconnessi, che in questo caso costituiscono le tessere del gioco.

I tetramini possono essere di sette tipi, corrispondenti a tutte le combinazioni di 4 quadrati con almeno un lato in comune, ed hanno nomi I, S, Z, T, O, L, J, come le lettere dell'alfabeto che più assomigliano alla loro forma.

Per tutto il tempo di una partita, dall'alto della schermata scendono tetramini di tipo casuale a cadenza regolare, e compito del giocatore è quello di disporli in modo che si formino righe complete di blocchi nel riquadro di gioco, allorché esse scompaiono ed i blocchi sovrastanti scendono ad occupare lo spazio liberato.

Molti matematici hanno studiato il gioco del Tetris per indagare se fosse possibile definire un algoritmo di risoluzione e creare un software in grado di giocare a questo gioco, ma si è giunti a definire che si tratta di un problema di tipo NP-completo [DEM08], che impegnerebbe il calcolatore per molto tempo ad ogni mossa, poiché dovrebbe provare ogni rotazione del pezzo per vederne gli incastri, e quindi non è facilmente implementabile un algoritmo che riesca a giocare.

Si è dimostrato, inoltre, che non è possibile giocare all'infinito, poiché esiste una sequenza di tetramini di tipo S e Z che comporta la sconfitta del giocatore, e dilatando il

tempo di gioco fino a farlo tendere all'infinito che questa sequenza si presenti è un evento certo.

1.3.2 Differenze nella programmazione di app native, web app e app ibride

Quando si parla della realizzazione di applicazioni per dispositivi mobili, come quella realizzata in questo progetto, i principali approcci di programmazione sono 3: lo sviluppo di applicazioni native, quello di applicazioni web, e quello di applicazioni ibride.

L'approccio più prestazionale in termini di esperienza utente, ottimizzazione dell'uso delle risorse fornite dai dispositivi, reattività generale, ed indipendenza dalla connessione ad internet, è quello dello sviluppo di applicazioni native.

Ogni sistema per dispositivi mobili è scritto in un determinato linguaggio di programmazione e segue determinati criteri di organizzazione al suo interno, cosa che lo rende diverso e, in generale, incompatibile rispetto agli altri sistemi.

Per ogni sistema operativo, però, sono forniti strumenti di sviluppo ed ambienti standardizzati di programmazione e testing atti alla creazione di applicazioni specifiche per quel tipo di sistema, che ne seguano, quindi, i criteri di progettazione e la grammatica.

Una applicazione scritta in maniera da essere supportata da un sistema operativo senza alcun tipo di intermediario che funga da interfaccia è detta applicazione nativa.

Per come nasce, una applicazione nativa può essere costruita ad hoc sul dispositivo, avendo accesso a tutte le sue funzionalità e sfruttandone le caratteristiche al meglio, inoltre, poiché viene eseguita direttamente da esso, senza bisogno di elementi intermedi, sarà inevitabilmente più veloce e reattiva rispetto ad applicazioni progettate seguendo approcci differenti.

Altro vantaggio dello sviluppo di una applicazione nativa rispetto quelle web, è che il codice viene scaricato al momento dell'installazione e risiede, a regime, completamente sul dispositivo, perciò essa non necessiterà di una connessione alla rete per entrare in esecuzione, cosa irrinunciabile per le applicazioni web.

Ultimo elemento importante a vantaggio delle applicazioni progettate appositamente per specifici sistemi è quello relativo alla visibilità una volta che il prodotto deve entrare in circolo: le applicazioni native, a differenza di quelle web, possono entrare negli store virtuali ufficiali, e quindi usufruire di tutta la pubblicità che

le case produttrici fanno di quegli store (fondamentalmente Google Play per Android e App Store per Apple), inoltre queste applicazioni hanno la possibilità di inviare notifiche push agli utenti che le hanno installate sul proprio dispositivo, in modo tale da fornire loro informazioni relative al software installato ed eventualmente pubblicizzare altri prodotti.

Lo svantaggio principale legato alle applicazioni native è quello derivante dal mancato supporto della portabilità, che questo tipo di progettazione implica intrinsecamente.

Dal momento che una applicazione nativa è concepita per funzionare alla perfezione, ma solo su un particolare tipo di sistema operativo, fornire il supporto a più sistemi comporta la riscrittura dell'intero codice per ogni tipo cui si vuole fornire questo supporto, operazione che comporta oneri elevati in termini di tempo e, in caso ci si debba avvalere, come spesso accade, dell'opera di più sviluppatori specializzati nei singoli sistemi, anche di denaro.

All'estremo opposto rispetto all'approccio dello sviluppo di applicazioni native si trova quello della produzione di applicazioni web.

Una applicazione web consiste in una ottimizzazione per sistemi mobili di un sito internet, di conseguenza sarà eseguita all'interno di un browser e non richiederà alcuna installazione sul dispositivo.

Questo tipo di applicazioni, come i siti web di cui costituiscono una variante, sono scritte usando HTML, CSS e JavaScript, ed è proprio questo che permette di eseguirle su tutti i dispositivi utilizzando lo stesso codice sorgente: ad eseguire il codice non sarà direttamente il dispositivo, ma il browser, concepito per essere sempre in grado di interpretare quei linguaggi di programmazione.

Per lo sviluppo di una applicazione web, quindi, sarà necessaria una sola programmazione, oltretutto in un contesto generalmente più semplice rispetto a quello dei consueti linguaggi di programmazione, condizione che porta numerosi vantaggi in termini di tempi di realizzazione e costi.

Nei termini visti fin ora, la semplicità che caratterizza le applicazioni web è quanto di meglio si possa desiderare, ma questa maggiore semplicità generale porta con sé una serie di aspetti negativi rispetto alle applicazioni native.

Per prima cosa una applicazione web necessiterà di una connessione ad internet attiva per tutto il tempo in cui sarà in esecuzione, e ciò ne limita l'uso ad ambienti in cui

esiste la possibilità di accedere alla rete, inoltre il fatto che ad eseguirle sia un browser pone un limite superiore alla velocità di esecuzione e alla disponibilità delle risorse a disposizione della app (sicuramente non maggiori di quelle destinate al browser).

A questi lati negativi si aggiunge anche il fatto di non poter interfacciare l'applicazione con funzionalità specifiche dei dispositivi, né inviare notifiche di tipo push all'utente, in quanto le web app non possono interagire direttamente con l'hardware e il software del dispositivo su cui sono in esecuzione.

Un'altra limitazione consiste nel fatto che questo tipo di applicazione non può essere pubblicato negli store digitali, e di conseguenza non può avvantaggiarsi di tutta la pubblicità che ne deriverebbe.

L'ultimo approccio alla creazione di applicazioni per dispositivi mobili è quello delle applicazioni ibride.

Questo stile di progettazione di applicazioni è quello nato più di recente, e, come dice il nome stesso, si propone di superare le limitazioni degli approcci precedenti fornendo una soluzione che presenti caratteristiche intermedie tra i due.

Il nucleo di una applicazione ibrida è molto simile a quello di una applicazione web, e quindi è scritto in HTML, JavaScript e CSS, ed è valido per tutti i sistemi per i quali la applicazione verrà implementata, ma il codice della applicazione ibrida non si esaurisce a questo: oltre al nucleo web, infatti, vengono fornite una serie di API in codice nativo per l'interfacciamento del nucleo web con il sistema operativo del dispositivo per cui si vuole realizzare l'applicazione.

Queste API traducono le istruzioni del codice sorgente in linguaggio nativo per il sistema operativo, permettendo un interfacciamento con software e hardware simile a quello delle applicazioni native.

Ovviamente tutto il codice di supporto al nucleo HTML dovrà cambiare a seconda del dispositivo per il quale si vuole rilasciare la applicazione, e, in generale, ne serviranno tante versioni quanti saranno i tipi di sistema operativo per cui si vorrà fornire il supporto, ma l'aiuto di strumenti automatizzati come Apache Cordova/Phonegap o Appcelerator Titanium rende pressoché indolore il passaggio da un sistema all'altro, poiché essi forniscono in maniera automatizzata tutto il codice di interfacciamento necessario, riducendo, di fatto, la programmazione di una applicazione ibrida a quella della sua sola componente web.

A questo punto le applicazioni ibride presentano una possibilità di interfacciamento con i dispositivi su cui sono eseguite molto maggiore rispetto alle applicazioni web, ma sono, nel contempo, molto più semplici da realizzare e mantenere rispetto alle applicazioni native, e questo, a patto di accettare una complessità comunque maggiore rispetto alle prime ed una velocità e reattività inferiori rispetto alle seconde, le rende una soluzione davvero interessante.

Nella realizzazione del progetto si è seguito, appunto, l'approccio di sviluppo ibrido, facendo riferimento al software Apache Cordova per quanto riguarda le API di interfacciamento tra il codice sorgente in HTML ed il sistema operativo dei dispositivi.

Tipo APP	INTEGRAZIONE COL DEVICE	TEMPI e COSTI DI SVILUPPO	FUNZIONAMENTO OFFLINE	MODALITA' DI DISTRIBUZIONE	APPROVAZIONE DALLO STORE
APP NATIVA	Sì	Alti	Sì	Store	Sì
APP IBRIDA	Sì	Medi	Alcune parti	Store	Sì
WEB APP	Limitato	Bassi	No	Solo Web	No

Figura 1.1: Schematizzazione delle principali caratteristiche degli approcci di programmazione per le applicazioni.

1.3.3 ICT e scuola in Italia

Per quel che riguarda l'introduzione delle nuove tecnologie nell'ambiente scolastico, ed il suo adeguamento alle mutate condizioni della vita quotidiana, l'Italia, benché abbia compiuto dei passi avanti e siano state promosse, almeno dal punto di vista organizzativo, delle iniziative tese alla innovazione della scuola, è sempre rimasta un po' indietro rispetto ad altri paesi avanzati europei, e quello di una mancata integrazione vera delle nuove tecnologie nell'ambiente scolastico e nella mentalità secondo cui la scuola è organizzata è tuttora un problema non completamente risolto.

I primi calcolatori nelle scuole sono stati introdotti nei primi anni '80, già qualche anno in ritardo rispetto altri paesi europei, con la commercializzazione dei primi personal

computer, ed ebbero inizialmente diffusione limitata quasi esclusivamente ad istituti di scuole superiori ad indirizzo tecnico o professionale, che, avendo mediamente più risorse economiche, riuscivano meglio a far fronte alla spesa per questi strumenti, allora non indifferente e lasciata gravare sui singoli istituti.

Una diffusione importante delle tecnologie informatiche, inoltre, è stata inizialmente impedita anche dai requisiti troppo tecnici che esse richiedevano all'utilizzatore, tanto che si iniziò, contestualmente alla introduzione dei pc, ad organizzare anche corsi di alfabetizzazione informatica, rivolti principalmente ai docenti di materie tecniche e scientifiche.

Nel 1985 venne lanciato il primo programma su scala nazionale per la diffusione e la gestione delle nuove tecnologie nel nostro paese: il *Piano Nazionale Informatica* (*PNI1*).

L'obiettivo di questo programma era quello di fornire agli insegnanti competenze informatiche di base per metterli in condizione di affrontare le modifiche ai programmi di insegnamento che in questo periodo si iniziavano a diffondere, benché ancora in via sperimentale, soprattutto in matematica e fisica.

L'effetto prodotto dal lancio del PNI1 fu una forte spinta verso la sperimentazione da parte dei docenti nell'innovazione degli insegnamenti; innovazione che si indirizzava verso lo sviluppo di competenze interdisciplinari negli studenti, e che coinvolse, con successivi progetti di ampliamento del PNI1, anche le scuole medie ed elementari, con l'obiettivo di introdurre le nuove leve ad un uso consapevole delle risorse informatiche e promuovere un collegamento dell'informatica con le materie di studio, soprattutto quelle scientifiche e matematiche.

Negli anni successivi (seconda metà degli anni '80 fino ai primi anni '90), si verificò una evoluzione dei sistemi e della loro integrazione nei programmi di studio: si iniziò, soprattutto negli istituti tecnici, ad introdurre software professionale (CAD, progettazione, gestione elettronica dei documenti, etc.) nei piani didattici delle materie volte alla formazione di tecnici professionisti, che in un contesto di progresso tecnologico diffuso avrebbero fatto riferimento a quegli strumenti anche in ambito lavorativo.

In questi anni, però, iniziò a svilupparsi anche una nuova ottica in relazione al ruolo che i computer potevano avere nell'educazione, ottica che avrebbe preso piede nel decennio successivo.

Nel 1991 partì il secondo progetto per l'informatica, il *PNI2*, questa volta esteso anche alle discipline di tipo linguistico e letterario.

Il grosso cambio di prospettiva che ebbe luogo in questi anni e che caratterizzò il PNI2 stava nell'iniziare a concepire il calcolatore non più semplicemente come uno strumento finalizzato ad uno scopo specifico, secondo un approccio di tipo meccanicistico, ma come mezzo di cui esplorare le possibilità in relazione all'insegnamento.

Nel PNI2 si voleva spostare l'attenzione dal computer in sé alle possibilità che esso offriva in relazione all'attività di insegnamento ed agli aspetti concettuali che la caratterizzavano, quindi non si voleva più solo fornire elementi di base di informatica nei percorsi scolastici, ma si voleva usare le nuove tecnologie nel contesto della didattica delle materie.

Nel 1997 ebbe inizio il *Programma di Sviluppo delle Tecnologie Didattiche* (*PSTD*) promosso dal Ministero della Pubblica Istruzione, volto, attraverso l'attivazione di progetti pilota, all'introduzione degli studenti alla multimedialità, al miglioramento della professionalità dei docenti in ambito informatico, e ad un rafforzamento dell'uso delle *ICT* (*Information and Communication Technologies*) nei curricula scolastici.

Gli anni a cavallo tra la fine del secolo scorso e l'inizio dell'attuale furono anche gli anni in cui iniziò a diffondersi Internet nell'ambiente scolastico: si svilupparono i primi usi didattici della rete, e iniziarono le sperimentazioni sullo sviluppo di corsi di formazione a distanza.

Nel frattempo continuavano le iniziative volte alla formazione in campo informatico del personale docente, come il programma ForTic (2003/2004), promosso dal MIUR (Ministero dell'Istruzione, dell'Università e della Ricerca).

Nonostante tutto questo la distribuzione delle nuove tecnologie nelle scuole è rimasta molto disomogenea, ed è rimasto basso il livello della loro vera integrazione con i programmi didattici.

Ci sono state diverse altre iniziative, come il piano DigiScuola (2005-2007), miranti alla riduzione del divario che si è nel tempo venuto a generare tra la diffusione dell'uso delle tecnologie dell'informazione nella vita quotidiana e la loro diffusione sempre inadeguata nell'ambito scolastico, ma sostanzialmente ancora nel 2011/2012 il nostro paese rimaneva fanalino di coda tra i paesi avanzati europei, con il minor accesso alla banda larga per gli studenti, indipendentemente dal grado di istituto [SOS13].

Nel 2007 è partito il piano Scuola Digitale, che è tutt'ora operativo e si prefigge di modificare gli ambienti scolastici con una introduzione massiccia di nuove tecnologie.

In questo contesto, secondo i dati pubblicati dal Miur tra il 2008 e il 2012, con il progetto *Azione LIM*, si sono introdotte 35.144 *LIM* (*Lavagna Interattiva Multimediale*) e si sono formati 72.357 docenti al loro uso sia tecnico che didattico.

Nel 2009 è iniziato anche il progetto *Classi 2.0*, che prevede la realizzazione di laboratori nelle classi per offrire agli studenti ambienti di apprendimento innovativi e che fino al 2012 ha coinvolto 416 classi nei vari ordini.

Sempre nel 2012 sono stati sottoscritti altri accordi per l'assegnamento di ulteriori 1.931 LIM e la formazione di 905 classi 2.0 e 23 Scuole 2.0.

Secondo i dati prodotti dall'Osservatorio tecnologico del Miur nell'analisi condotta per l'anno scolastico 2014/2015 circa la *dematerializzazione dei servizi* (siti e portali, comunicazione scuola-famiglia, registro elettronico, gestione dei contenuti didattici multimediali), la *dotazione tecnologica dei laboratori e delle biblioteche* (connessioni, computer, LIM e proiettori interattivi) e la *dotazione tecnologica delle aule* (connessioni, dispositivi fissi e mobili in dotazione a studenti e docenti, LIM e proiettori interattivi):

- il 99,3% degli istituti ha un proprio sito web, il 58,3% fa uso di comunicazioni scuola-famiglia online, il 69,2% si serve di registri di classe elettronici, e il 73,6% usa quello del docente.
- I laboratori nelle scuole sono in totale 65.650, con una media di 7,8 per istituto, e di questi l'82,5% è connesso in rete, il 43,6% è dotato di LIM, e il 16,9% di proiettore interattivo.
- Il 70% delle classi è connesso alla rete (in modalità cablata o wireless), anche se generalmente con una connessione inadatta alla didattica digitale, il 41,9% è dotato di LIM, e il 6,1% di proiettore interattivo.

Per quel che riguarda gli strumenti che si intende fornire alle scuole, il prossimo passo in programma è quello di portare fibra ottica e wi-fi in tutti gli istituti entro il 2020, come prevosto dal *Piano Banda Ultra Larga*, in modo da consentire, ad esempio, l'uso di soluzioni cloud per la didattica, o la fruizione di contenuti di apprendimento multimediali, e contestualmente effettuare una vasta opera di cablaggio LAN e wireless (già programmato per quest'anno) per consentire l'accesso alla rete in ogni spazio scolastico.

Altri punti in programma riguardano la digitalizzazione della burocrazia, la collaborazione con ricercatori ed esperti per costruzione di progetti di ricerca, la diffusione tra le nove leve di nuove competenze tipicamente legate al contesto attuale e futuro, come il problem solving, il pensiero laterale, la capacità di muoversi nell'ambiente digitale e la possibilità, e, a partire da dicembre 2015, la apertura dele scuole all'utilizzo di dispositivi elettronici personali durante le attività didattiche (BYOD – Bring Your Own Device).

Da ultimo, si porrà attenzione anche alla formazione dei docenti affinché siano in grado di utilizzare i mezzi che ci si propone di mettere loro a disposizione, cercando, in questo modo, di massimizzare l'integrazione concreta delle nuove tecnologie nell'ambiente scolastico e nell'istruzione delle nuove leve.

2 Specifiche del progetto e strumenti di sviluppo utilizzati

2.1 Presentazione e contestualizzazione del progetto di tesi

2.1.1 Contesto in cui si inserisce il progetto di tesi

Il contesto in cui questo progetto di tesi affonda le sue radici e quello della collaborazione tra docenti dell'Università di Bologna e docenti dell'Istituto Comprensivo Statale 7 "L. Orsini" di Imola, che prevede la realizzazione da parte degli studenti universitari di una applicazione prototipo che applichi principi di gamification e edutainment per stimolare e facilitare l'apprendimento di nozioni di matematica, ed il testing sul campo di tale applicazione da parte delle classi medie dell'istituto, che forniranno un feedback sulla sua efficacia in un contesto applicativo.

I dispositivi target del progetto sono quelli mobili, poiché al giorno d'oggi essi assumono un ruolo sempre più importante nella vita privata e, specialmente a seguito dei nuovi progetti statali proposti nell'ambito della operazione *Scuola 2.0*, anche in quella scolastica, dove iniziano a ricoprire ruoli prima appannaggio esclusivo di libri e quaderni.

Il sistema elaborato come prototipo in questo progetto di tesi è concepito per essere una parte della applicazione più ampia, che è stata progettata per comporsi di una serie di mini-giochi volti a facilitare l'apprendimento di specifici argomenti, ognuno dei quali viene concepito come un modulo a sé stante, e per interagire con gli altri dovrà interfacciarsi ad un substrato di interconnessione coerente fornito dall'applicazione.

In questo modo il sistema mira a massimizzare la facilità di modifica, aggiornamento ed ampliazione: elementi molto importanti in vista di implementazioni future.

2.1.2 Il ruolo del Tangram

La sezione del progetto finale di cui ci si è occupati è quella relativa ad uno dei mini-giochi che verranno presentati all'utente, ed in particolare viene presentato il gioco del Tangram.

La versione mostrata in questa occasione è una versione di esempio che si è provveduto a rendere indipendente, ma a progetto ultimato sarà solo una parte di un'entità più complessa, che ne gestirà il lancio in base alle necessità.

A tal scopo si è provveduto a progettare il modulo in maniera tale che richieda il minimo interfacciamento con un eventuale sistema che lo voglia inglobare, così da semplificarne il più possibile la gestione dall'esterno: in particolare, l'intera applicazione viene vista dall'esterno come una singola funzione cui passare un numero minimo di parametri, relativi all'eventuale stato del gioco, al livello di difficoltà e al tema prescelto.

Gli obiettivi che ci si pone introducendo il gioco del Tangram sono quelli di facilitare la familiarizzazione degli utenti con alcune figure geometriche e con i movimenti rigidi sul piano cartesiano, e fornire loro la possibilità di fare sperimentazioni sui concetti di equi-estensione di figure piane, confronti di aree e conservazione delle superfici.

Il gioco del Tangram ha origini incerte sia dal punto di vista temporale, sia da quello geografico: si ritiene sia nato in Cina, alcuni sostengono addirittura intorno a 3000 anni fa, anche se la redazione dei primi libretti di regole e soluzioni risale solo al XIX secolo [CAS90], e la prima volta in cui compare la descrizione di un gioco simile risale al 1742, in un libro pubblicato in Giappone [ARC15].

In antichità si riteneva che chi riuscisse a padroneggiare l'uso di questo gioco avesse la chiave per ottenere talento e saggezza, tanto che, tra i vari nomi sotto cui era noto, vi era anche quello di "Le sette pietre della saggezza" [MAT04].

Il Tangram si compone di sette tessere, dette *tan*, aventi le forme di un quadrato, un parallelogramma, e 5 triangoli rettangoli isosceli (2 grandi, 1 medio e 2 piccoli), ricavate a partire da un quadrato di dimensione maggiore secondo un procedimento che ne divide l'area solo lungo le diagonali o lungo segmenti congiungenti i punti che dividono i vari lati/diagonali a metà o in quarti [CIT13].

Per come il gioco è costruito esistono, dunque, rapporti fissi tra i lati delle varie figure e, di conseguenza, tra le loro aree: in particolare essi sono confrontabili in rapporti multipli di 2.

Si può dire, ad esempio, che l'area di uno dei triangoli più grandi vale il doppio di quella del triangolo di dimensioni medie, e che questa, a sua volta, ha dimensioni doppie rispetto a quella di uno dei triangoli più piccoli, ma identiche rispetto a quelle del quadrato e del parallelogramma, e ragionamenti analoghi si possono fare anche per i lati.

Altra particolarità geometrica derivante dalle modalità di organizzazione del gioco è legata agli angoli presentati dalle figure, che possono essere solo multipli di 45°: in particolare 45°, 90° o 135°.

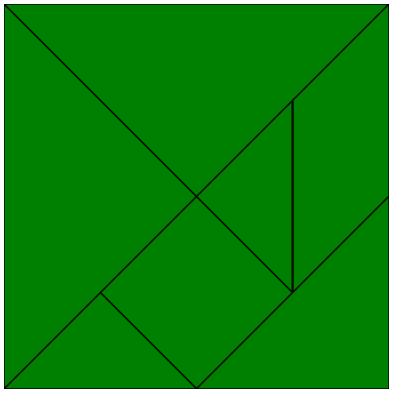

Figura 2.2: Quadrato iniziale Tangram

I tan sono inizialmente disposti in modo tale da formate il quadrato di partenza (*figura 1.2*), ed il giocatore deve riuscire a combinarli senza sovrapposizioni in modo tale da riprodurre un'immagine fornitagli come modello (*figura 1.3*), che può essere puramente geometrica o ispirata ad elementi reali. Unici strumenti a sua disposizione per il posizionamento delle figure sono la traslazione, la rotazione e, solo per il parallelogramma, il ribaltamento.

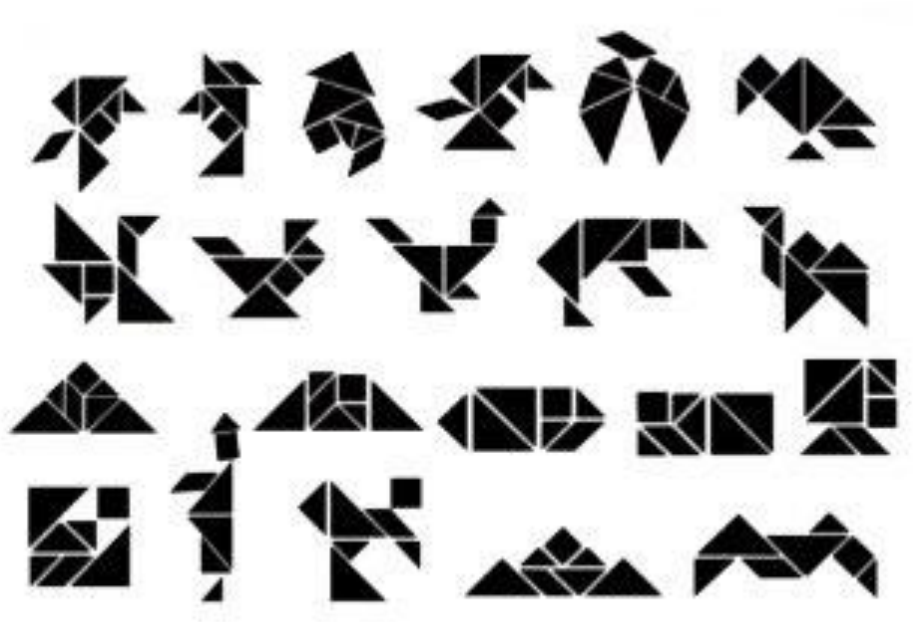

Figura 2.3: Esempi di soluzioni del Tangram

La stampa dei primi libretti contenenti regole e soluzioni risale solo al XIX secolo, e le sole immagini riportate erano quelle tradizionalmente utilizzate per il gioco, ma il Tangram ebbe, successivamente grande diffusione anche in Europa [CAS90], e gode ancora oggi di un discreto successo, tanto che si trovano anche moltissime applicazioni web-based e per dispositivi mobili che lo riproducono per piattaforme informatiche.

Nell'ambito di una applicazione che si ponga come obiettivo quello di fornire nel contempo giochi di intrattenimento per i ragazzi della scuola media, ed una piattaforma di allenamento nell'ambito dell'aritmetica, della logica e della geometria, si è ritenuto che questo gioco rispondesse alle esigenze del progetto nel suo complesso, date le caratteristiche, che sin dalla sua origine, lo rendono tanto un gioco quanto uno strumento di allenamento al pensiero matematico.

2.2 Gestione del contesto di gioco

Il prototipo sviluppato in questo progetto di tesi ha come obiettivo quello di dare la possibilità di essere utilizzato come applicazione autonoma in sede di dimostrazione e presentazione, ma garantire, nel contempo, un livello di modularità che ne consenta l'inserimento all'interno di progetti di carattere più ampio.

Quando si pensa al supporto della compatibilità con altri sistemi, si deve necessariamente pensare che questi potrebbero seguire modelli di gestione delle dinamiche di gioco complessi, come quelli basati su una *saga* o sull'*accudimento* di un avatar o di creature digitali ed in questo contesto tali sistemi potrebbero prevedere livelli a difficoltà crescente, ed organizzazioni delle partite anche molto articolate, che potrebbero addirittura coinvolgere più applicazioni elementari per generare videogiochi più articolati.

Per fare fronte a queste eventualità e mantenere la massima flessibilità nella gestione delle dinamiche di gioco, si è ritenuto opportuno organizzare la gestione delle sessioni di utilizzo secondo l'unita fondamentale di partite elementari che coinvolgono un solo livello.

In questa sede all'utente viene fornita la possibilità di giocare un numero grande a piacere di partite successive tutte indipendenti tra loro, selezionando il livello di difficoltà che preferisce tra i tre proposti in un menù dedicato, ma si preserva la possibilità, ai fini di una successiva integrazione all'interno di software più elaborati, di gestire le dinamiche di gioco in maniera differente, demandando alla eventuale applicazione di grado superiore la selezione automatizzata del livello di difficoltà, del numero di partite elementari da svolgere per completare il gioco e, in generale, di tutto quanto sia relativo allo *stato* del sistema.

2.3 Tecnologie e metodi di progettazione e sviluppo utilizzati per la realizzazione del progetto

Come accennato nel paragrafo 1.3.2, la tipologia di applicazioni cui fa capo quella sviluppata in questo progetto è quella delle applicazioni ibride: si procederà, in questo paragrafo, ad una presentazione degli strumenti e delle tecnologie utilizzati nella realizzazione di tale applicazione.

Dal momento che le applicazioni ibride si caratterizzano per la presenza di un nucleo concepito in modalità analoghe a quelle dello sviluppo di applicazioni web, si è necessariamente dovuto fare riferimento alle tecnologie di programmazione web, ed in particolare ai linguaggi HTML5, CSS3 e JavaScript, utilizzando anche il framework JavaScript KineticJS, dedicato alla gestione di elementi di grafica 2D e basato sull'uso dell'elemento *canvas*, introdotto per la prima volta con HTML 5.

Per l'interfacciamento del nucleo web dell'applicazione con il sistema operativo dei sistemi mobili si è fatto, invece, riferimento al framework Apache Cordova, tra i più quotati per lo sviluppo di applicazioni ibride, nella sua versione incorporata come plugin nell'ambiente di sviluppo Microsoft Visual Studio 2015.

2.3.1 HTML 5

Il linguaggio *HTML* (*Hyper Text Markup Language*) è un linguaggio per la marcatura di testi derivante dal più generale *SGML* (*Standard Generalized Markup Language*) e costituisce lo standard per la produzione di testo per il web [BRA14a].

La caratteristica principale di questo tipo linguaggio è la definizione di un insieme di elementi detti *marcatori* o *tag*, tramite i quali è possibile definire ed identificare le diverse parti della pagina, inserire all'interno del testo elementi multimediali quali immagini, suoni e animazioni, e, più in generale, determinare tutti gli aspetti relativi al *rendering* che non riguardino la *presentazione*, per gestire la quale si fa uso degli appositi *fogli di stile CSS* (*Cascading Style Sheet*), di cui si parlerà nel seguito.

Nei documenti l'uso dei marcatori, che possono essere innestati gli uni dentro gli altri secondo le dinamiche definite dalla grammatica del linguaggio, porta alla formazione di una struttura ad albero che rispecchia la gerarchia di contenimento delle sezioni di marcatura.

L'elemento radice dell'albero è costituito dal tag `<html>`, che definisce i limiti del documento ed ospita al suo interno tutto il contenuto della pagina e tutta la logica di rendering. Il contenuto di tale tag è organizzato in due elementi principali: l'intestazione, contenuta nel tag `<head>`, ed il corpo del documento, contenuto nel tag `<body>`. La struttura base che risulta per un documento HTML è la seguente:

```
<!DOCTYPE html>
<html>
    <head>
        <title> Titolo del documento </title>
        Contenuto dell'head
    </head>
    <body>
        Contenuto del body
    </body>
</html>
```

Nell'esempio si identificano tre elementi principali:

- La *dichiarazione del tipo di documento*, che non fa propriamente parte del linguaggio HTML (infatti sta all'esterno del tag `<html>`), ma e necessaria per identificare quale specifica grammatica segue un particolare documento, e quindi fornisce gli estremi per interpretarlo e stabilirne la correttezza e la validità.
- Una sezione di *intestazione* contenuta tra l'apertura e la chiusura del tag `<head>`, il cui unico elemento visibile è il titolo, mentre per il resto contiene informazioni utili ai fini della gestione e dell'interpretazione del documento ed i riferimenti a tutti gli script che andranno caricati prima dell'esecuzione del corpo della pagina.
- Una sezione di *contenuti* tra l'apertura e la chiusura del tag `<body>`, in cui è ospitato il vero contenuto che comparirà nella pagina finale ed i riferimenti agli script ed alle risorse che saranno necessari per il suo funzionamento.

I marcatori vengono interpretati da programmi appositamente studiati (i *web browser*) che si incaricano del reperimento di tutte le risorse esterne richieste dal documento, della loro integrazione nel testo della pagina, e della strutturazione dei contenuti secondo quanto stabilito nel codice, mostrando all'utente il risultato finale.

Nella maggior parte dei casi le risorse non risiedono nella macchina su cui è in esecuzione il browser, quindi il compito del programma, oltre a quello di visualizzare in maniera corretta i documenti scritti in HTML, è quello di fungere da *client* per reperire tali risorse dal *WWW* (*World Wide Web*) alla locazione indicatagli nel codice della pagina.

Il protocollo seguito dai browser per comunicare con i server ed effettuare questa operazione è il protocollo HTTP.

Tra i browser più diffusi al momento della scrittura di questa tesi si trovano *Google Chrome*, *Mozilla Firefox*, *Microsoft Internet Expolorer*, *Opera* e *Safari* [W3C15].

Il linguaggio HTML nasce nel 1989, anche se la prima versione veramente utilizzata è, nel 1994, HTML 2.0, nato per Mosaic, primo browser ad alta diffusione da cui derivò Netscape [BRA14a].

Inizialmente lo sviluppo e l'estensione di HTML fu in carico al *CERN* (*Conseil Européen pour la Recherche Nucléaire* / *Organizzazione Europea per la Ricerca Nucleare*), dove lavorava lo scienziato inglese Tim Berners-Lee, inventore del WWW nel 1989, e all'*IETF* (*Internet Engineering Task Force*), ma con la nascita del *W3C*

(*World Wide Web Consortium*), la gestione dell'evoluzione di questo linguaggio passò in mano a quest'ultimo [BRA14a].

Dopo un primo tentativo semi-abortito di estendere HTML 2.0 (HTML 3.0), un approccio più pragmatico portò, nel 1997, alla nascita di HTML 3.2, evoluto, nello stesso anno, in HTML 4.

Dopo una ulteriore evoluzione di HTML 4 in HTML 4.01 (1999), il W3C decise di smettere di evolvere questo tipo di linguaggio, e di crearne, piuttosto, uno nuovo che ne traducesse le caratteristiche in un formato compatibile con i canoni dell'XML: fu così che nel 2000 nacque XHTML 1.0, che non aggiungeva nulla ad HTML 4.01, se non la possibilità di serializzazione comportata dal rispetto dei canoni XML.

Successivamente a questo si iniziò a lavorare allo sviluppo di un nuovo linguaggio non compatibile con HTML 4, né con XHTML 1.0: XHTML 2.

Poiché questo linguaggio sarebbe stato, di fatto, limitato alle sole tecnologie più nuove, violando il principio di *retrocompatibilità* che ritenevano dovesse stare alla base delle tecnologie web, Apple, Mozilla, Opera e Google si unirono, nel 2004, nel *WHATWG* (*Web Hypertext Application Technology Work Group*), e si opposero allo sviluppo di XHTML.

Dopo il respingimento, nel 2004, della proposta di rinnovamento di HTML da parte dei W3C, il WHATWG decise di incaricarsi dello sviluppo e dell'evoluzione di tale linguaggio in parallelo rispetto all'attività del consorzio, il quale avrebbe iniziato a collaborare tre anni dopo per lo sviluppo di HTML 5, complice una innovazione nell'ambito della gestione delle *form* che aveva riportato l'attenzione sul vecchio linguaggio [W3C14].

Dalla collaborazione tra i due gruppi di lavoro, il 28 ottobre 2014 nacque la *HTML 5 Recommendation*, che costituisce una sorta di standard codificato per l'ultima versione del linguaggio di markup di Internet.

Le principali innovazioni introdotte in HTML 5 rispetto alla versione precedente sono focalizzate al miglioramento del disaccoppiamento tra la struttura del testo, affidata al markup, le caratteristiche di presentazione della pagina, affidate ai fogli di stile, ed il contenuto, costituito dal testo vero e proprio.

In particolare, in HTML 5 si riscontra:

- un *miglioramento della modularità degli elementi costitutivi di una pagina web*, in modo tale da consentire la scrittura di un codice più *strutturato* ed *ordinato*;

- una *struttura del testo più rigorosa e completa*, con suddivisioni esplicite in capitoli e paragrafi e l'introduzione di *tag specifici per ogni sezione della pagina*;
- il supporto alla memorizzazione locale di dati scaricati tramite il browser, in modo tale che applicazioni che richiedano la archiviazione di dati offline non necessitino dell'installazione di plugin ad hoc;
- la introduzione di un meccanismo detto *Html5Storage*, che punta a rimpiazzare i cookies;
- la riduzione del contenuto del tag `<!doctype>` ad un semplice `<!doctype html>` in luogo delle stringhe molto lunghe che caratterizzavano le intestazioni precedenti ad HTML 5;
- la multimedialità gestita in modo nativo dal linguaggio tramite i tag `<audio>` e `<video>` laddove prima era necessario l'uso di plugin esterni come Flash;
- la deprecazione di alcuni tag presenti nelle versioni precedenti del linguaggio quali `<frame>`, quelli dedicati alla formattazione del testo e quelli relativi a tutti gli aspetti di presentazione;
- l'introduzione della struttura *Canvas*, che permette la gestione di grafica vettoriale in sezioni di pagina web (tecnologia usata anche all'interno di questo progetto);
- un maggior supporto alla gestione degli errori, laddove i browser in precedenza non erano in grado di distinguere errori nel codice, e si limitavano a dare risultati imprevedibili in loro presenza.

2.3.2 CSS3

Il linguaggio HTML nasce con lo scopo di descrivere il *contenuto* di un documento e la sua *struttura*, ma non è concepito per occuparsi delle relative *caratteristiche di visualizzazione*.

Il prototipo WWW di Tim Berners-Lee del 1989 concepiva un linguaggio di stile che consentisse agli *utenti* di definire personalmente aspetti relativi alla presentazione dei documenti HTML sui propri dispositivi, e i primi browser (Mosaic), seppure limitando le opzioni alla scelta della dimensione e del tipo dei font usati, permettevano una gestione di tali aspetti impostata secondo questo modello.

Con il successo di WWW e di HTML, e la contestuale diffusione dei siti, tra gli autori di pagine web iniziarono a comparire esperti di marketing, grafici e tipografi [BRA14b], e si rese sempre più importante una gestione centralizzata degli aspetti di presentazione dei contenuti.

Inizialmente questo venne fatto inserendo le informazioni di presentazione all'interno del documento HTML sotto forma di tag e attributi, tanto che nella specifica di HTML 3.2 venne introdotta una grande quantità di elementi dedicati esclusivamente alla definizione delle caratteristiche tipografiche e di formattazione del testo, ma questa non si dimostrò una scelta particolarmente felice.

I problemi principali derivanti da questo approccio sono la *replicazione del codice*, poiché è necessario ripetere le istruzioni di formattazione in ogni pagina ed in ogni sezione che ne fa uso, la conseguente *non riusabilità del codice*, dal momento che un documento che debba essere visualizzato secondo modalità diverse deve essere necessariamente riscritto ex novo, la *difficoltà di modifica e manutenzione* dei documenti, poiché qualunque modifica alla formattazione deve essere replicata con un intervento importante su tutte le pagine interessate, e la *limitatezza delle possibilità offerte dai marcatori introdotti in HTML*, che non consentono un controllo preciso di molti aspetti del layout.

La necessità di riprodurre i contenuti HTML su molteplici dispositivi e addirittura su media differenti (per esempio un supporto video digitale o una periferica per la stampa su carta, o, ancora, una che soddisfi requisiti di accessibilità particolari, quali la riproduzione vocale di testo per utenti non vedenti o la stampa braille), e di farli interagire con diverse applicazioni stimolò la separazione netta tra gli aspetti relativi al contenuto e quelli di presentazione, favorendo la diffusione di linguaggi di stile specializzati.

Tra i vari linguaggi di questo tipo quello che ha avuto più seguito, fino a diventare, di fatto, lo standard in accoppiamento ad HTML, è il Cascading StyleSheet (CSS) [W3C15b]

Con la combinazione di HTML (dalla versione 4) e dei fogli di stile CSS, nel linguaggio di marcatura rimangono solo i tag relativi ai contenuti e alla loro strutturazione, mentre le caratteristiche di presentazione vengono affidate ai fogli di stile [BRA14b].

Questo introduce un ulteriore livello di complessità nella progettazione, specialmente per ottenere risultati uniformi su tutti i browser, ma garantisce grossi

vantaggi in termini di riutilizzabilità di medesimi schemi di presentazione in contesti e documenti diversi, e semplifica la modifica e la manutenzione, dal momento che è necessario apportare modifiche al solo foglio di stile per applicare le variazioni a tutti i documenti che ne fanno uso.

Particolarità di CSS è quella di incoraggiare la presenza di molteplici fogli di stile che vengono applicati *in cascata*, permettendo sia all'autore che all'utente di avere il controllo sulle caratteristiche di stile del documento, mettendo a disposizione un meccanismo per applicare regole di presentazione a più livelli di dettaglio tramite dichiarazioni successive con target sempre più restrittivo.

Applicando regole a target generale si definiscono proprietà di presentazione per tutti gli elementi di un certo tipo, e facendole seguire da altre con bersaglio più ristretto è possibile personalizzare l'aspetto solo di elementi mirati sovrascrivendo quanto impostato in precedenza.

Un'altra caratteristica vincente del linguaggio CSS è la sua indipendenza dall'insieme degli elementi e degli attributi di HTML, caratteristica che ne permette l'utilizzo in accoppiamento con tutte le versioni di HTML e con altri linguaggi dalle caratteristiche simili (l'uso di tag di marcatura), come XML.

Per quanto riguarda la struttura di un documento CSS, essa è costituita da un insieme di *regole* composte da un *selettore* ed una o più *dichiarazioni* del tipo "*proprietà: valore*" racchiuse tra parentesi graffe e seguite da "*;*".

Il selettore, a seconda di quanto sono restrittivi i filtri applicati alla selezione, può identificare gruppi di elementi che spaziano da una certa categoria (espressa dal tag del linguaggio di marcatura cui si accoppia) fino ad un elemento specifico (ad esempio ponendo un filtro sull'attributo *id*), o anche tutti gli elementi contenuti in una certa sezione o nell'intero documento tramite l'uso del *selettore universale* "*". Ciò che ne risulta è un codice del tipo:

```
selettore {
      proprietà : valore;
      proprietà : valore;
      .
      .
      proprietà : valore;
}
```

Quando analizza il documento da riprodurre, il browser crea una struttura ad albero, chiamata *document model,* rispecchiante la struttura della pagina, identifica il tipo di medium di destinazione, e carica tutti i fogli di stile definiti per il documento in relazione a quel particolare tipo di medium, applicandone le regole al document model.
A ciascun elemento dell'albero è assegnato un unico valore per ciascuna delle proprietà legate allo stile.
Tale valore risulta dalla applicazione in cascata di tutte le regole di ciascun foglio di stile relative a quel particolare attributo, in modo tale che faccia fede solo la definizione stabilita per ultima.
HTML prevede quattro modi diversi per collegare i fogli di stile alle pagine web, che in ordine crescente di separazione contenuto-presentazione (e quindi di riusabilità, semplicità di gestione e preferibilità) sono:

- Posizionamento diretto delle regole di stile nel tag di riferimento tramite l'attributo *style*, con sintassi `<tag style="regola di stile; …; regola di stile;>…</tag>`
- Inserimento delle regole di stile all'interno di un tag `<style>` apposito che vale per il documento corrente, con sintassi `<style type="text/css"> regole di stile; </style>`
- Importazione del contenuto di un file CSS separato all'interno di un tag `<style>` analogo al caso precedente tramite la clausola @import, con sintassi `<style type="text/css"> @import url(percorso_al_file.css) </style>`
- Indicazione nel tag `<link>`, con sintassi `<link type="text/css" rel="stylesheet" href=" percorso_al_file.css ">`

Per concludere la trattazione del CSS si presenterà brevemente l'evoluzione delle sue versioni dalla prima fino a quella attuale.
Nella fase iniziale della loro vita i browser non offrivano il supporto alla definizione di fogli di stile, e si caratterizzavano per il fatto di lasciare il controllo degli aspetti di presentazione nelle mani dell'utente finale, in maniera simile, benché implementata in maniera molto meno completa, a quella di un editor di testo.
La storia del CSS iniziò nel 1994, in un contesto in cui, con la diffusione del WWW e dei siti web, gli autori richiedevano maggiori possibilità di gestione della veste grafica

dei loro documenti, mentre i browser ed il linguaggio di codifica permettevano modifiche minime [WIU99].

La prima volta in cui un progetto di CSS ottenne lo stato di *recommendation* dal W3C fu nel 1996, con CSS1[W3C08].

Ancora si trattava di uno strumento supportato da pochi browser, e per di più in maniera parziale: il primo a supportarlo in parte fu Microsoft Internet Explorer 3, realizzato nel 1996, seguito da Netscape Navigator 4 e successivamente anche da Opera, nel 1998 [WIU99].

Sempre nel 1998 divenne standard W3C CSS2 [W3C98], che nasceva come naturale evoluzione di CSS1 e godeva di un discreto supporto da parte dei browser [MIL10].

Con CSS2 venne introdotto il supporto a media multipli, con fogli di stile specifici per il mezzo, e si aveva a disposizione un linguaggio sofisticato e complesso per quel che riguardava il posizionamento delle sezioni di documento HTML, la gestione di font (anche scaricati dalla rete), l'impaginazione di tabelle, l'internazionalizzazione dei contenuti e la gestione dell'interfaccia utente [BRA14b].

Un miglioramento di CSS2 avvenne nel 2004, con l'introduzione di CSS2.1, che sarebbe diventato *recommendation* W3C solo nel 2011[W3C14b].

Esso risolveva alcuni errori della versione precedente ed apportava varie correzioni, aggiungendo alcune funzionalità rispetto a CSS2 e rimuovendone alcune non supportate dai browser [BRA14b].

A partire dal 2009 iniziò ad affermarsi CSS3 [W3C15c], che estendeva le funzionalità di CSS2.1 e si presentava, pur mantenendo una totale retrocompatibilità, in una nuova forma organizzata in oltre 50 moduli, laddove le precedenti versioni erano concepite come enormi specifiche singole.

Al momento la maggior parte dei moduli CSS3, benché largamente utilizzati nella pratica, non ha ancora status di recommendation W3C, e gli unici ufficializzati sono *Color* (2011), *Namespace* (2011), *Media Queries* (2012), *Selectors Level 3* (2011), *Style Attributes* (2013) [MDN15].

Nel frattempo è in fase di progettazione anche CSS4 [W3C11].

2.3.3 JavaScript

JavaScript nacque nel 1995 ad opera di Brendan Eich, ingegnere presso Netscape, e venne rilasciato per la prima volta con la versione 2.0 del browser Netscape Navigator

all'inizio del 1996, dapprima sotto il nome di LiveScript, e subito dopo con il nome attuale.

Fornendo funzionalità di calcolo e manipolazione dei documenti che non coinvolgevano il server, sin dall'inizio JavaScript permise la modifica dinamica delle pagine web tramite la sola interazione con il browser (quindi totalmente lato client), e per questo negli anni '90 ci si riferiva all'accoppiata HTML-JavaScript con il nome di *DHTML* (*Dynamic HTML*) [CHI14].

Di pochi mesi successivo al lancio di JavaScript da parte di Netscape, fu quello della controparte prodotta da Microsoft, rilasciata con Internet Explorer 3 sotto il nome di *JScript*.

A questo punto si decise di sottoporre il linguaggio all'azione di un ente di standardizzazione industriale: l'*ECMA International*, dall'opera del quale nel 1997 nacque lo standard *ECMA-262*, che rappresentava le specifiche del linguaggio *ECMAScript* [ECM15].

Su ECMAScript si sarebbero basate non solo le successive evoluzioni di JavaScript, ma anche altri linguaggi di scripting (ActionScript, WMLScript e QtScript) [CHI14].

Nel 1999 ECMAScript subì un importante aggiornamento come ECMAScript Edition 3 e un altro, dopo una abortita quarta edizione, nel 2009 come ECMAScript Edition 5: la versione attuale è la Edition 6, rilasciata nel giugno 2015 [ECM15].

Dopo un periodo in cui le sempre maggiori esigenze di interattività e dinamicità portarono il grande successo di alcune tecnologie concorrenti, come i plugin *Flash*, i controlli *ActiveX* e gli *Applet Java* (tecnologie che nonostante imponessero limitazioni in termini di compatibilità fornivano la possibilità di realizzare funzionalità ed effetti grafici di maggiore impatto rispetto a quanto possibile con JavaScript), un ritorno in primo piano coinvolse il linguaggio JavaScript con l'introduzione della tecnologia *Ajax*, che permetteva la comunicazione asincrona con il server tramite script [MDN08].

Con il rinnovato interesse verso il linguaggio e le sue nuove possibilità di utilizzo nacque il cosiddetto *Web 2.0*, e fiorirono numerose librerie volte alla semplificazione di alcune delle attività più comuni e alla uniformazione dei browser, appianando le diversità più forti che ancora sussistevano tra essi al fine di rendere possibile una programmazione unificata e più semplice [CHI14].

Citando Andrea Chiarelli dal sito HTML.it, "*L'evoluzione del linguaggio da una parte e l'avvento di HTML5 dall'altra hanno ulteriormente amplificato le possibilità applicative del linguaggio, anche al di fuori del semplice Web browser*" [CHI14], tant'è che oggi JavaScript offre possibilità di utilizzo sia lato client sia lato server, tanto in applicazioni desktop quanto in contesto mobile, ed è ben lontano dall'essere solo un semplice collante tra il codice HTML e l'utente, come invece accadeva all'inizio.

Benché la maggior parte della enorme diffusione di questo linguaggio sia dovuta "al fiorire di numerose librerie nate allo scopo di semplificare la programmazione sul browser" [CHI14], condizione che lo vede ancora applicato secondo le modalità originali, sono nati anche molti framework lato server ed in ambiente mobile che lo supportano come linguaggio principale [CHI14].

Si può, in generale, identificare, per JavaScript, una struttura divisa in tre sezioni principali (figura 2.2):

- Un *nucleo*, che comprende il codice standard così come stabilito dal documento ECMA-262 e costituisce la parte comune tra la versione del linguaggio orientata alla codifica client-side e quella orientata al lato server.
- Una sezione dedicata alle *espansioni finalizzate alla codifica client-side*, che si accoppia con quella di nucleo e ne espande le funzionalità.
- Una sezione dedicata alle *espansioni finalizzate alla codifica server-side*, anch'essa destinata all'accoppiamento con il nucleo per espanderlo in una determinata direzione.

Parti comuni a JavaScript client-side e server-side sono *il linguaggio di base*, gli *oggetti predefiniti di uso comune* come Array, Date, e Math (differiscono quelli specifici di lati client e lato server), ed il meccanismo *LiveConnect*, che permette l'utilizzo di codice Java da JavaScript e viceversa.

Prendendo la definizione iniziale del linguaggio, ancora valida per la sua parte di nucleo, benché, come detto, nella sua evoluzione il linguaggio si sia aperto alla possibilità di esecuzione lato server, addirittura permettendo anche il non-embedding e la compilazione, JavaScript è un linguaggio di *scripting client-side embedded eseguito da un browser* [BRA14c].

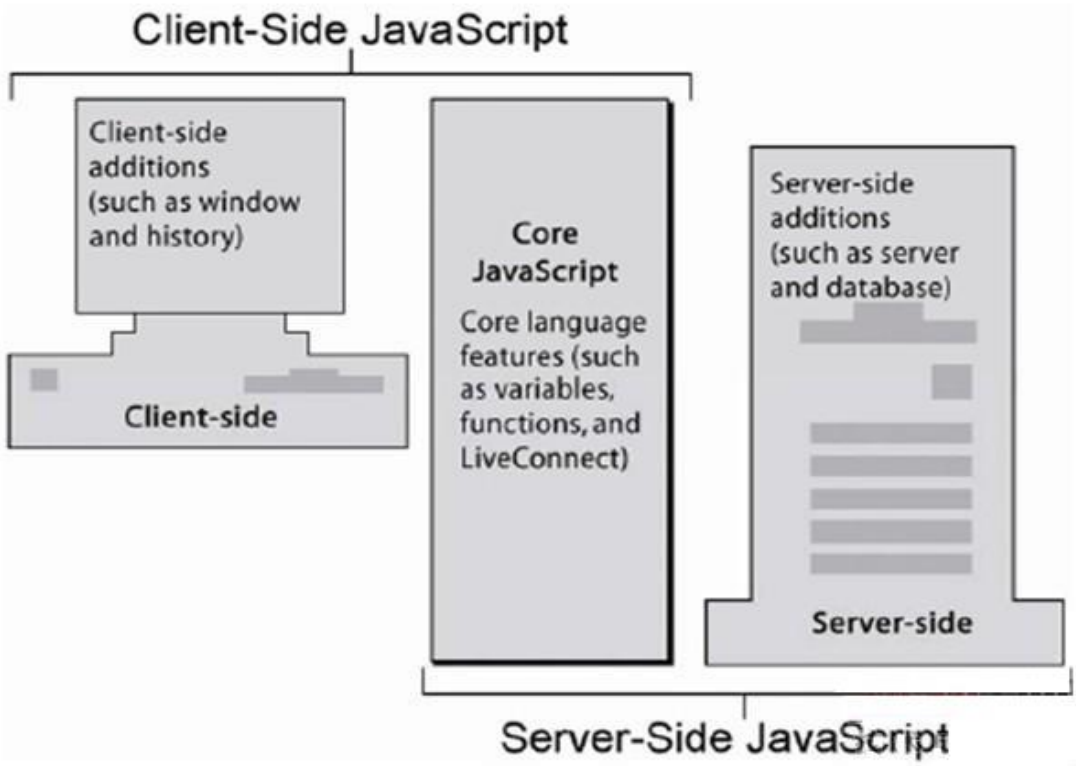

Figura 2.2: Struttura sezioni linguaggio JavaScript

Separando tale definizione nelle sue componenti fondamentali si ricava che:

- *Scripting*: si tratta di un linguaggio interpretato, e non necessita, quindi, della compilazione di tutte le istruzioni da parte di un software apposito a monte della esecuzione dell'intero codice (come avviene, per esempio, per Java), ma ogni istruzione viene letta, tradotta ed eseguita dal browser web nel momento in cui è necessaria la sua esecuzione.
- *Client-side*: si tratta di un codice che, benché utilizzato in contesto web, viene eseguito solo lato client, cioè dal browser sul dispositivo in cui esso è in esecuzione, e non richiede l'interazione con il server.
- *Embedded*: si tratta di codice incorporato nell'HTML tramite l'uso di appositi tag di marcatura `<script>`, e quindi non è possibile una sua esecuzione disgiunta da quella del codice della pagina web in cui è inserito.

Per quel che riguarda la sintassi, JavaScript assomiglia a C, di cui è una estensione [BRA14c], e per la gestione degli oggetti, benché utilizzino politiche differenti, a Java.

Importante differenza rispetto sia a C sia a Java riguarda la *tipizzazione*, il meccanismo, cioè, di assegnamento del tipo alle variabili, che per JavaScript è *dinamica*, e permette alle variabili di assumere in modo dinamico valori di tipi differenti in base alle necessità, mentre per gli altri linguaggi è *statica*, e impone, per questo, alle variabili l'assunzione di valori appartenenti ad un solo tipo di dato.

Si conclude l'introduzione al linguaggio di scripting (che non vuole essere una lezione teorica approfondita sulla sua grammatica), evidenziando alcune differenze importanti che intercorrono tra i linguaggi JavaScript e Java, che non sono simili tra loro quanto il nome porterebbe a pensare. Innanzitutto Java è un linguaggio di tipo *compilato*, mentre JavaScript, come detto in precedenza, è *interpretato*, inoltre Java è un linguaggio completamente *Object-Oriented* (*OO*), e prevede l'uso di *classi* e l'inclusione dei meccanismi base della programmazione OO, primo tra tutti quello *dell'ereditarietà*, mentre JavaScript è solo *Object-Based*, e non si basa sul concetto di classe di oggetti, ma su quelli di *tipo di oggetto* e di *oggetto prototipo*, che permettono comunque la definizione di oggetti e la loro estensione con metodi o campi aggiuntivi, ma non permettono il supporto di *ereditarietà* e *definizione di sottotipi* (in inglese *subtyping*). Per concludere si riprende quanto già accennato in precedenza, ricordando i differenti approcci usati per la tipizzazione delle variabili, che è *statica* nel caso di Java e *dinamica* in quello di JavaScript, con le conseguenze precedentemente citate. Si termina proponendo uno specchietto riassuntivo delle principali differenze evidenziate tra Java e JavaScript [BRA14], mostrato in figura 2.3.

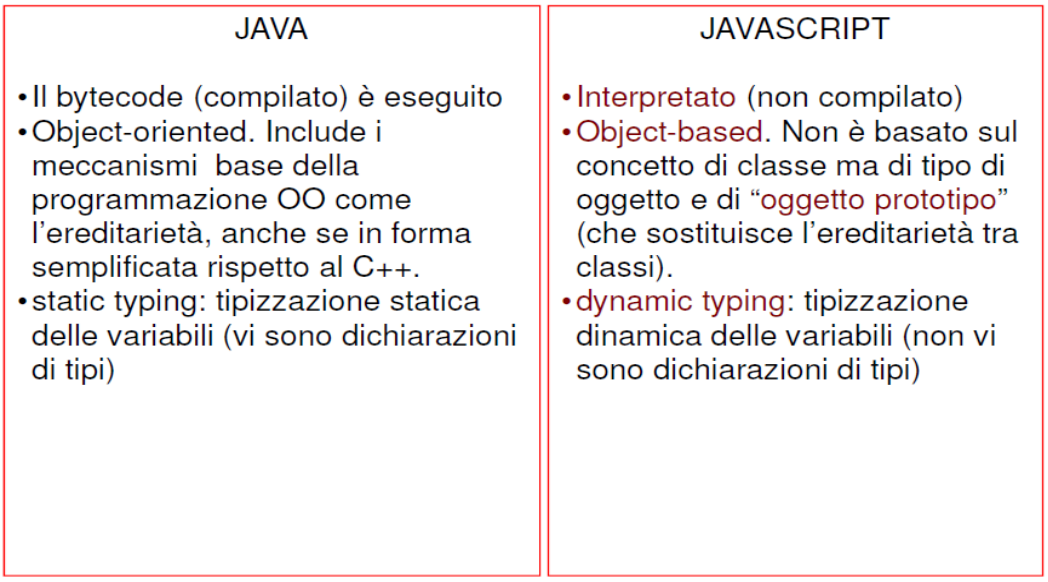

Figura 2.3: Differenze principali tra Java e JavaScript

2.3.4 KineticJS e Canvas

L'elemento *canvas* (tag `<canvas>`) è un nuovo componente introdotto per la prima volta come standard con HTML5 [W3C14c].

Questo tag identifica un'area nel documento composta da una matrice di pixel su cui si può intervenire settando le componenti di colore RGB e la trasparenza, permettendo, quindi, il disegno di grafici vettoriali tramite linguaggi di scripting come JavaScript [BEO11].

Importante nella definizione del canvas è il settaggio degli attributi di *altezza*, *larghezza* e *id* del tag: i primi due sono indispensabili per fornire una dimensione effettiva al riquadro destinato al disegno vettoriale, l'ultimo per poterlo indirizzare all'interno degli script che dovranno disegnarne il contenuto.

Nato nel 2004 come codice proprietario prodotto da Apple per le necessità dei widget della dashboard di OsX, ed inserito nel MacOs X WebKit ad uso degli sviluppatori, negli anni successivi l'elemento canvas è stato utilizzato anche da altri produttori e browser, così da essere accettato prima dal WHATWG, e poi anche dal W3C, entrando nelle specifiche per la redazione dello standard HTML5 [PAG11].

Benché si tratti di uno strumento potente, per come è concepito, canvas non presenta una organizzazione interna orientata agli oggetti, cosa che ne rende complesso l'utilizzo per la creazione di immagini contenenti più elementi, e ancor più per la gestione delle animazioni; fortunatamente, però, esistono molte librerie ed API di interfacciamento concepite per facilitarne l'uso che consentono un buono sfruttamento delle sue potenzialità ed una gestione più semplice del codice.

Tra questi software si annovera anche il framework utilizzato per la realizzazione di questo progetto: KineticJS.

KineticJS è un framework JavaScript orientato agli oggetti destinato allo sviluppo di software tanto per contesti desktop, quanto per contesti mobile, ed è concepito per gestire con un buon livello di performance animazioni, transizioni, innestamento di nodi, stratificazione di *layer* multipli e l'intercettazione e la gestione di eventi.

Fino a dicembre 2014 questo software è stato sviluppato e mantenuto dal suo creatore Eric Rowel, che lo ha distribuito a licenza gratuita sia sul proprio sito, sia sulla piattaforma GitHub, ma ora, nonostante rimangano reperibili, a beneficio di chi ne ha fatto uso per i propri siti, tutte le versioni rilasciate, il progetto non viene più sviluppato attivamente, e vede il suo erede diretto in KonovaJS, nato come suo *fork*.

La struttura di base di KineticJS è lo *stage*, che si compone di *layer* definiti dall'utente.

Per ogni layer vengono istanziati due elementi *canvas*: uno visibile che costituisce la scena vera e propria, ed un altro nascosto, finalizzato all'intercettazione ad alto rendimento degli eventi.

Ogni layer può contenere forme geometriche (*shapes*), gruppi di forme, o anche gruppi composti da altri gruppi.

Tutte le strutture, dallo stage alla singola figura geometrica, sono gestite come nodi virtuali, in modo tale che il documento è caratterizzato da una struttura ad albero non dissimile da quella formata dai nodi in una pagina HTML (per un esempio di veda la *figura 2.2*).

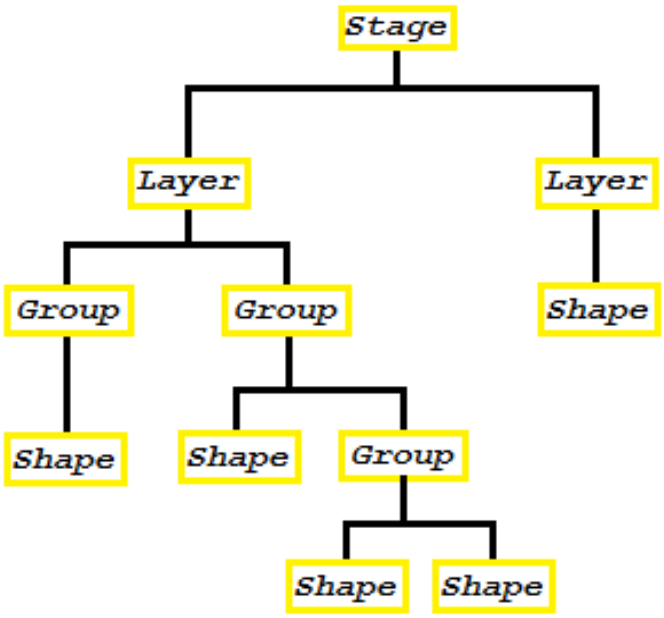

Figura 2.2: Esempio di struttura ad albero formata dai nodi di un file realizzato con KineticJS

Ogni nodo è un oggetto caratterizzato da una serie di attributi e proprietà che ne consentono la personalizzazione e la trasformazione.

Il framework contiene già la definizione di una serie di forme geometriche di base, ma è sempre possibile definire figure ad hoc istanziando un oggetto della classe *Shape* ed implementandone il metodo *draw*.

Una volta popolato lo stage di oggetti, è possibile collegare ad essi degli *event listeners*, che permettono di renderli reattivi definendone il comportamento a fronte di determinati eventi generati dall'utente o da altre parti dell'applicazione.

Tra le varie caratteristiche di questo framework alcune di particolare interesse per l'applicazione sviluppata in questo progetto di tesi sono:

- L'implementazione del *drag and drop* e la possibilità di stabilire dei confini per l'area di trascinamento delle figure.
- Il fatto di essere un framework concepito sia per desktop sia per dispositivi mobili, elemento importante per il tipo di applicazione che si intende produrre, dal momento che consente l'intercettazione e la gestione di eventi per entrambi gli scenari di utilizzo.
- L'orientazione agli oggetti, e la possibilità di generare gruppi di figure, cosa che facilita molto la gestione delle tessere e la possibilità di aggiungere elementi grafici solidali con il loro movimento.

2.3.5 Apache Cordova

Apache Cordova nasce come versione open source di un software di successo per la produzione di applicazioni ibride chiamato PhoneGap, prodotto originariamente da una azienda canadese di nome Nitobi Software.

Nel 2011 Adobe Systems acquistò il progetto PhoneGap, e contestualmente la Nitobi donò il codice sorgente alla associazione Apache [CHI13].

Oggi il progetto è portato avanti da Adobe in due differenti versioni: il codice di PhoneGap donato dalla Nitobi viene rilasciato in modalità open source con licenza Apache 2.0 sotto il nome di Apache Cordova, mentre con il nome originale di PhoneGap viene rilasciata la versione a pagamento, che presenta la possibilità di usufruire di servizi aggiuntivi.

Le applicazioni prodotte utilizzando Apache Cordova sono organizzate in quattro sezioni fondamentali:

- Il *nucleo* dell'applicazione è costituito da una vera e propria *applicazione web* eseguita localmente.

  In questa sezione vengono usati HTML5 e CSS3 per la strutturazione ed il rendering del contenuto dell'applicazione, e JavaScript per quanto riguarda la definizione della sua dinamica e della sua logica di funzionamento.
- La *WebView* è il contenitore entro il quale è messa in esecuzione la sezione web dell'applicazione.

Questo elemento occupa l'intero schermo del dispositivo e, come un browser, permette di visualizzare i risultati dell'interpretazione di HTML e CSS, e di eseguire il codice JavaScript contenuto nell'applicazione.

- Tramite JavaScript è possibile accedere ad un *insieme di API* concepite per interfacciare la sezione web dell'applicazione con le funzionalità native del sistema ospitante.

  Il framework implementa lo stesso insieme di API su tutte le diverse piattaforme mobili supportate, e fornisce, quindi, un substrato di software standard al quale è possibile interfacciare le applicazioni [CHI13].

  In Apache Cordova sono inclusi in modo nativo una serie di plugin per l'interfacciamento con alcuni elementi di base del sistema che eseguirà l'applicazione, come la fotocamera, l'accelerometro, o il file system, ma è sempre possibile estendere le funzionalità e l'interfacciamento tramite l'inserimento di plugin di terze parti.
- L'ultima sezione è costituita dal *sistema operativo* installato sul dispositivo, che sarà il materiale esecutore di tutte le operazioni per cui sarà richiesto un interfacciamento tra la applicazione web e le funzionalità specifiche del sistema su cui è in esecuzione.

Le uniche parti della applicazione che dipendono direttamente dalla specifica piattaforma mobile sono la WebView e le API di interfacciamento, ed è proprio questo che Apache Cordova mette a disposizione dello sviluppatore, permettendogli di focalizzare la propria attenzione sul codice web da inserire nel nucleo, indipendente dalla piattaforma specifica.

In questo modo viene fornito uno strumento potente per produrre con sforzo minimo applicazioni che, quantunque generalmente più lente rispetto alle applicazioni native, sono sviluppate in un solo momento per tutte le piattaforme supportate, e possono, nel contempo, essere ammesse negli store virtuali alla stessa stregua delle applicazioni sviluppate in linguaggio specifico per ogni dispositivo.

Al momento della realizzazione del progetto le piattaforme mobili supportate da Apache Cordova sono Android, iOS, Blackberry, Bada, Tizen e Windows Phone [CHI13].

Figura 2.1: Struttura di una applicazione realizzata con Apache Cordova

2.4 Gestione di temi e ambientazioni di gioco multipli

Il programma sviluppato offre il supporto di più temi e ambientazioni, in modo tale da prestarsi in maniera nativa ad ampliamenti successivi e adattamenti ad esigenze e gusti diversi.

Affinché questo sia possibile, tutti gli aspetti relativi alla presentazione, sia per quanto riguarda il codice HTML, sia per quanto riguarda gli aspetti grafici del gioco, sono parametrizzati e separati dalle sezioni che si occupano della struttura e della gestione della dinamica dell'applicazione, in modo tale che per inserire nuovi temi ci si possa limitare a fornire le eventuali risorse multimediali richieste, modificare il file JavaScript *themes.js* ed il foglio di stile, e inserire gli opportuni riferimenti per il caricamento delle informazioni nei file html e JavaScript che dovranno farne uso.

All'interno del file *themes.js* i vari temi sono concepiti come funzioni che saranno invocate dal motore effettivo del gioco.

L'applicazione si limita a creare le strutture di base per le tessere e le soluzioni nei loro aspetti generali ed indipendenti dalla veste grafica prescelta, e demanda all'invocazione delle funzioni di applicazione del tema i settaggi di colori, immagini e tutto ciò che sia inerente l'aspetto estetico del gioco.

Non vengono posti limiti alla complessità delle operazioni che possono venire svolte all'interno delle funzioni di settaggio, né vengono posti limiti per quanto riguarda l'interfaccia che tali funzioni sono tenute a rispettare, purché i risultati prodotti dalla loro esecuzione abbiano un significato coerente con le dinamiche del gioco proposto.

È già possibile riscontrare un esempio di quanto detto nei due soli temi gestiti al momento della presentazione.

Allo stato attuale l'applicazione presenta una opzione di stile base, che si è deciso di mantenere minimale dal punto di vista della grafica, ed una a tema zombie, che immagina le sette tessere del tangram come zattere montate da altrettanti zombie e la soluzione come un disegno formato da un'ombra visibile in trasparenza sotto il livello dell'acqua.

Se, da un lato, la funzione di impostazione del tema minimale si limita a settare semplicemente colori uniformi per il riempimento ed i bordi dei tan, delle figure che compongono l'immagine target, e del pannello di sfondo, la funzione relativa al tema zombie imposta immagini di sfondo per ogni tessera, introduce nel sistema nuovi elementi (le immagini degli zombie) che dovranno avere movimenti e comportamenti coerenti con quelli di altri (le relative tessere/zattere), ed inserisce un livello intermedio tra le tessere e le immagini di target (la texture semi-trasparente dell'acqua).

Tutto questo si riflette in un cambiamento della struttura interna di memorizzazione delle tessere, che dopo il lancio della funzione di settaggio del tema zombie non conterrà più i soli oggetti tessera, ma gruppi di oggetti che faranno riferimento ai sistemi zattera-zombie, e tutto ciò avviene in maniera trasparente all'utente e senza modifiche al file contenente la dinamica dell'applicazione.

In conclusione si riporta che vengono fornite funzioni di supporto (implementate ma non utilizzate) che consentono, qualora servisse, il disegno di una griglia sullo sfondo della schermata di gioco cui è possibili allineare le tessere con un effetto calamita (comunque presente anche in assenza di tale griglia).

3 Implementazione e fase di test

In questa sezione ci si occuperà dell'esposizione nel dettaglio dell'implementazione interna dell'applicazione, incominciando dalla descrizione della sua architettura e dei suoi componenti, per passare alla presentazione delle dinamiche di gioco e dei casi d'uso gestiti, e finendo con una sezione relativa ai test effettuati, ai problemi riscontrati e alle eventuali soluzioni proposte.

3.1 Architettura dell'applicazione

L'applicazione prodotta in questo progetto di tesi segue il modello di progettazione ibrido: si compone, cioè, di un nucleo sviluppato secondo il modello delle applicazioni web, e di una sezione di interfacciamento con il sistema operativo del dispositivo su cui viene eseguita.

Considerando questa ultima sezione, il codice della quale viene inserito in maniera automatica con l'uso del software Apache Cordova, alla stregua di uno strumento, e non avendo prodotto direttamente nulla di quanto la compone, ci si concentrerà sul nucleo web della applicazione, che costituisce la parte originale sviluppata in questo progetto.

L'applicazione è orientata agli eventi, che costituiscono il principale meccanismo di interazione tra l'utente ed il sistema, e segue il paradigma di progettazione ad oggetti, molto efficace per gli scopi di interattività che ci si prefigge in questo progetto.

Il nucleo dell'applicazione è composto da sei file: *index.html*, *game.html*, *engine.js*, *levels.js*, *themes.js* e *index.css*, organizzati come verrà esposto nel seguito.

Il primo file ad entrare in esecuzione è, come di norma, *index.html*, che in questo progetto codifica un menu iniziale da proporre all'utente per la raccolta di informazioni necessarie all'esecuzione, quali il *tema prescelto* (al momento dell'esposizione ne vengono proposti due), il *livello di difficoltà*, che determina anche il massimo livello di

punti ottenibile con la vittoria della partita (3000 per difficoltà bassa, 4000 per difficoltà media, 5000 per difficoltà elevata), e i dati relativi al giocatore (nome, cognome, id e stato iniziale del punteggio): informazioni che in una versione definitiva innestata in un progetto più ampio verranno reperite da un database gestito dall'applicazione di livello superiore, ma in questo momento vengono richieste all'utente al solo fine di mostrarne la gestione da parte del sistema, benché non abbiano propriamente un significato.

Con la pressione del bottone *Inizia* il controllo passa da *index.html* alla pagina codificata da *game.html*: quella destinata al gioco vero e proprio.

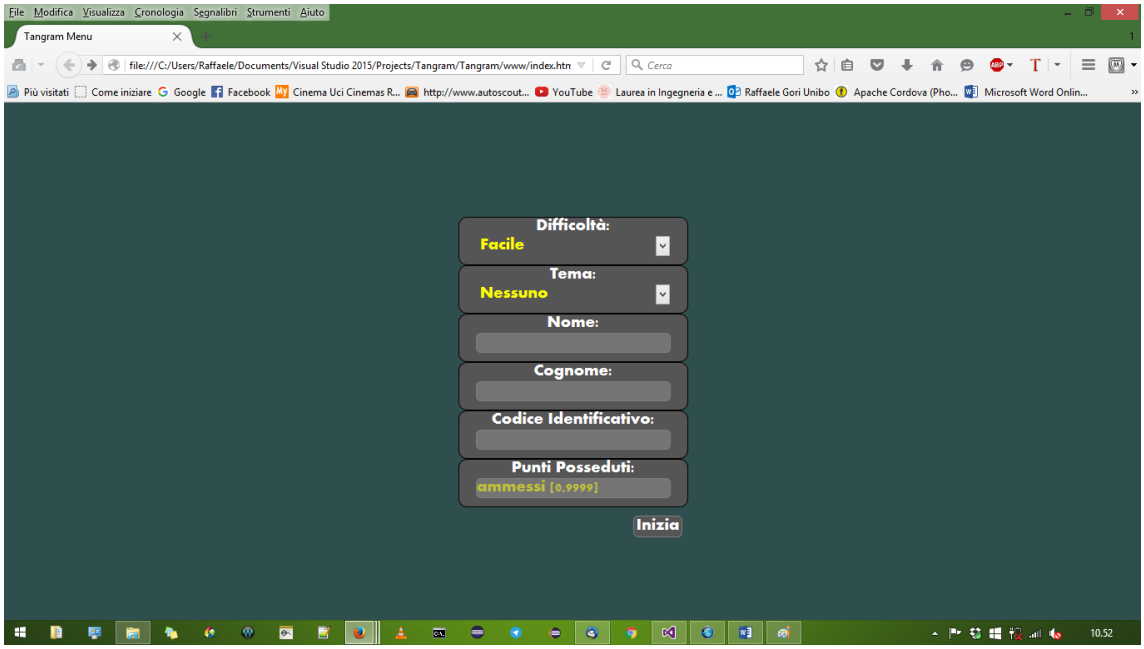

Figura 3.1: Cattura della finestra di menù in esecuzione su browser Mozilla Firefox

Questa pagina carica dapprima i file JavaScript di supporto (*levels.js*, *themes.js*) e quello relativo al framework per la gestione dell'elemento *canvas* (*kinetic-v5.1.0.min.js*), e poi quelli di supporto alla gestione delle API di interfacciamento per la parte ibrida (*cordova.js* e *platformOverrides.js*) e quello contenente il motore del sistema (*engine.js*), che codifica la effettiva logica del gioco e farà uso del codice contenuto nei primi due file di supporto e nella libreria del framework.

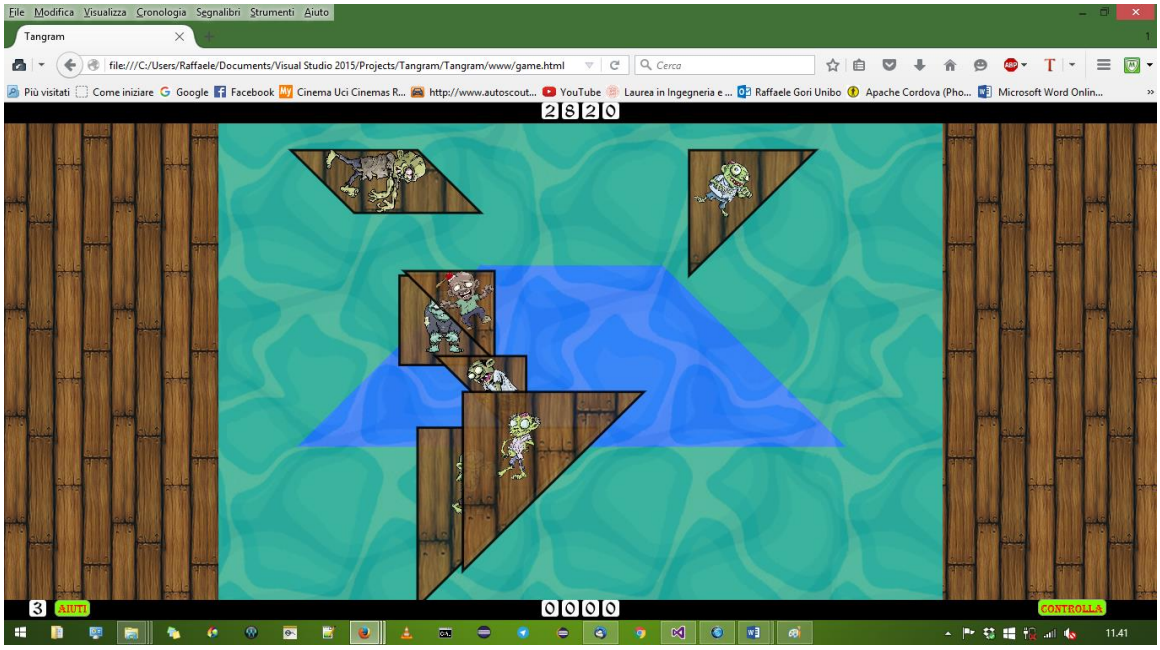

Figura 3.2: Cattura della finestra di gioco in esecuzione su browser Mozilla Firefox

In figura 3.3 una rappresentazione schematica dell'architettura e delle relazioni di utilizzo e caricamento.

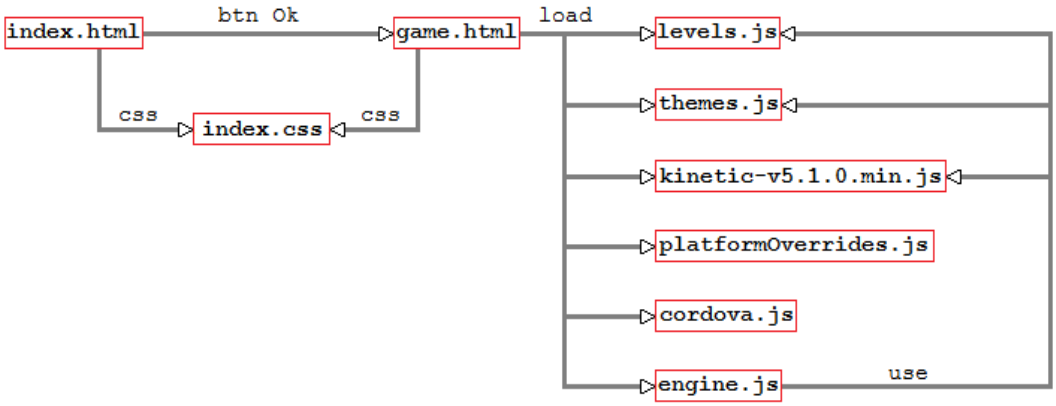

Figura 3.3: Flusso del caricamento e delle dipendenze tra i file della applicazione

3.1.1 Analisi nel dettaglio dei singoli blocchi

Si presenta, ora, una analisi nel dettaglio dei file realizzati (gli altri, come già detto, si considereranno alla stregua di strumenti e si invita, pertanto, alla consultazione

delle rispettive documentazioni per informazioni più approfondite di quanto non si potrebbe fornire in questa sede):

index.html

E' la pagina html destinata al menù di selezione dei valori iniziali da passare al motore dell'applicazione.

Essedo, l'applicazione, pensata per essere un mini-gioco ed appoggiarsi ad uno strumento che le funga da sostegno in background per tutto quanto riguarda la gestione dei dati, questa pagina ha ragione di esistere solo in fase di test e di presentazione per permettere all'applicazione di avere tutto quanto le risulta necessario per l'esecuzione.

Per la presentazione *index.html* fa riferimento all'unico foglio di stile *index.css* e si compone di una semplice sezione di `<head>` in cui viene caricato tale foglio di stile, ed una sezione di `<body>`, in cui compare un `<div>` deputato al pannello dei comandi.

Tale pannello contiene sei `<fieldset>` ed il bottone per il proseguimento verso la schermata di gioco.

I `<fieldset>` consentono la selezione del tema e del livello di difficoltà, e l'inserimento di nome, cognome, codice identificativo e punteggio pregresso.

All'interno del `<body>` è contenuta anche una sezione di `<script>` in cui è determinato in che modo il bottone dovrà reagire alla pressione.

In questo script è contenuta la definizione della funzione *validate()*, che di fatto è ciò che materialmente mette in esecuzione il gioco.

In particolare, questa funzione salva come campi dell'oggetto *sessionStorage* introdotto con HTML5 i valori settati dall'utente per i campi proposti nel menù, e aggiunge un valore, espresso in numero di punti, per il livello coerentemente con la difficoltà prescelta (3000 per difficoltà facile, 4000 per difficoltà media e 5000 per difficoltà elevata), punti che verranno decrementati alla velocità di 10 ogni 2 secondi fino a quando il livello non sia stato vinto o non si sia raggiunta quota 0 punti.

Come ultima operazione lo `<script>` carica la pagina *game.html* al posto di quella attuale, passandole il controllo e permettendo l'inizio della partita.

game.html

E' la pagina html destinata alla visualizzazione della schermata di gioco.

Nella sezione di <head> vengono caricati gli script di supporto *themes.js* e *levels.js*, e quello contenente il framework JavaScript KineticJS, utilizzato per la gestione della grafica 2D del gioco (*kinetic-v5.1.0.min.js*).

La sezione di <body> si presenta suddivisa in tre sottosezioni: un <header>, un centrale ed un <footer>.

L'<header> contiene la definizione di quattro immagini per la rappresentazione delle cifre del punteggio iniziale (si arriva a rappresentare numeri solo fino alle migliaia) ed una sezione di <script> che regola la dinamica della loro rappresentazione (il punteggio scala di 10 unità ogni 2 secondi, e le immagini che lo rappresentano vengono aggiornate di conseguenza).

Chiusa la sezione di <header>, vengono caricati gli script necessari alla gestione ibrida dell'applicazione: *cordova.js* e *platformOverrides.js*, e quello che contiene il motore del gioco: *engine.js*, di cui si parlerà in dettaglio nel seguito.

Il centrale funge da *container* per il <canvas> generato dinamicamente dal framework ed ospita al suo interno due <script> destinati rispettivamente al collegamento al pannello di una funzione di ridimensionamento, che interviene al variare delle dimensioni della finestra, ed all'invocazione della stessa funzione all'apertura del gioco al fine di garantire che lo *stage* abbia sempre dimensioni coerenti con quelle della finestra quando il gioco viene lanciato.

Ultimo elemento del <body> della pagina è un <footer> destinato alla rappresentazione del punteggio totale e del numero di aiuti disponibili, ed inoltre ad ospitare i bottoni per la richiesta di aiuti e per il lancio della routine di convalida della soluzione.

Il <footer> contiene, dunque, cinque immagini, quattro delle quali rappresentano il punteggio totale capitalizzato dal giocatore (funzione che ha senso solo in un contesto in cui si tenga traccia dello stato del gioco tra le partite, cosa che in modalità demo non avviene, ma viene implementata comunque in vista di espansioni ed usi futuri), e la quinta il numero residuo di aiuti disponibili, cui si aggiungono i due bottoni per richiedere gli aiuti e lanciare la funzione di convalida della soluzione.

A queste strutture si aggiunge l'ultimo elemento che compone la pagina: uno <script> definito in testa al <body> contenente la definizione della funzione *numToImage(num)*, che prende in ingresso un valore numerico (per poter funzionare non

deve superare le 4 cifre) e fornisce in output l'array dei riferimenti ai *path* di reperimento delle immagini corrispondenti ad ogni cifra, stabiliti per il tema attuale.

In questo modo è possibile associare al campo *src* di ognuno dei tag `<img>` definiti nell'`<header>` e nel `<footer>` il percorso alla risorsa che dovranno contenere.

***index*.css**

E' il file contenente le regole di stile per entrambi i file html presentati sopra (gli unici che compaiano nell'applicazione), e non presenta alcuna caratteristica di particolare interesse, se non, forse, il fatto che si è cercato di definire comportamenti dinamici che si adattino ad una gamma di schermi quanto più vasta e variegata possibile, dal momento che il target principale è quello rappresentato dai dispositivi mobili (smartphone e tablet).

engine.js

E' il cuore vero e proprio dell'applicazione: in questo file vengono generati tutti gli oggetti che comporranno la schermata di gioco propriamente detta e vengono definiti tutti i comportamenti delle tessere a fronte delle interazioni con l'utente, codificati all'interno di *event listeners* collegati agli oggetti interattivi.

Data la complessità di una descrizione verbale di quanto elaborato nel codice, si comincerà introducendo una tabella informativa sui significati delle variabili e delle funzioni presenti nel file, cercando di fornire uno strumento nel contempo completo e ben comprensibile, che funga da guida per affrontare, poi, in maniera più agevole e profittevole la descrizione del funzionamento del codice.

VARIABILI DI SUPPORTO	
theme	Riferimento al tema selezionato dall'utente.
solved	Booleano che contiene l'esito dell'invocazione della funzione controllo. Viene utilizzato dallo script di *game.html* deputato alla diminuzione periodica del punteggio ottenibile per sapere quando il livello è superato e fermare il decremento.

shapePixel	Sorta di fattore di scala interno per gli elementi contenuti nel canvas che viene utilizzato per gestire i rapporti tra le loro dimensioni e quelle delle unità della griglia utilizzata per la funzione calamita. Costituisce un valore di compromesso per ottenere un allineamento abbastanza preciso, ma non basato su unità infinitesime.
SCENE_BASE_WIDTH ***e*** ***SCENE_BASE_HEIGHT***	Valori utilizzati per stabilire la risoluzione di base dello stage, definendo un rapporto base-altezza che rimane costante, nel contesto della funzione di ridimensionamento del pannello di gioco al variare delle dimensioni della finestra.
cnt	Valore utilizzato per limitare superiormente il numero di aiuti invocabili in ogni partita ad una quota stabilita (al momento impostata al valore delle tessere) anche in presenza di un numero di aiuti disponibili maggiore.
numAiuti	Variabile che contiene il numero di richieste di aiuto capitalizzato dal giocatore fino al momento presente.
dblClk	Booleano usato nel gestore dell'evento di *click/tap* per controllare che nel periodo di attesa non si sia verificato l'evento di doppio *click/tap,* e permettere una gestione separata dei due eventi.

VARIABILI RELATIVE ALLO STAGE	
stage	Variabile che indica il contenitore delle quinte di gioco. Si appoggia sul `<div>` *container* in *game.html*.
layer	Effettiva quinta del gioco. Questa è destinata alla rappresentazione del livello di sfondo.
tansLayer	Quinta del gioco deputata alla presentazione dei sette tan.
rect	Pannello per il colore dello sfondo (bianco per il tema neutro, verde per quello ad argomento zombie).
squareTan, triangle1Tan, triangle2Tan, triangle3Tan, triangle3Tan, triangle5Tan, parallelogramTan	Variabili corrispondenti ad oggetti *Line* del framework KineticJS utilizzate per rappresentare le tessere a disposizione del giocatore.
tans	Array contenente i riferimenti a tutti gli oggetti tessera per facilitarne la gestione.
xCent e yCent	Valori delle coordinate del centro dello stage approssimati a multipli di *shapePixel*.
squareSol, triangle1Sol, triangle2Sol, triangle3Sol, triangle3Sol, triangle5Sol, parallelogramSol	Oggetti gemelli di quelli utilizzati per i tan, eccezion fatta per il fatto che hanno colore diverso e non sono reattivi agli eventi di *click/tap* e *drag&drop*. Vengono utilizzati per generare la figura target sullo sfondo del pannello di gioco disponendoli in base al primo array di coordinate contenuto in *solCheck[imageNum]*.
sol	Array similare a *tans* utilizzato per indirizzare in maniera più agevole le tessere di soluzione.

FUNZIONI DI SUPPORTO	
stampa()	Funzione di supporto finalizzata ad una più facile produzione degli array di coordinate utilizzati per le soluzioni. Quando invocata stampa su console l'array delle coordinate e degli angoli di rotazione attuali dei tan sullo stage già in formato valido per l'inserimento nell'array delle soluzioni con un semplice *copy&paste*, in modo da consentire di inventare nuove soluzioni disponendo le tessere sul piano invece di fare calcoli complessi. Viene lasciata, anche se non utilizzata durante la partita, come utile strumento per eventuali ampliamenti futuri.
controllo()	Funzione di validazione della soluzione proposta dall'utente. Recuperati i valori di coordinate e angoli di rotazione di tutte le tessere al momento dell'invocazione, li compara con tutti gli elementi contenuti nell'array delle soluzioni possibili per quell'immagine target (*solCheck[imageNum]*) e restituisce un booleano corrispondente all'esito della ricerca di corrispondenze.

helpReq()	Funzione di gestione delle richieste di aiuto. Se la richiesta avviene in una condizione in cui siano disponibili aiuti, ed il valore della variabile di controllo *cnt* sia inferiore alla soglia stabilita (in questo momento i numero di tessere soluzione, così da impedire lo *"spreco"* di aiuti), il sistema sceglie un numero casuale compreso tra 0 e l'indice massimo dell'array *sol*, e se la tessera soluzione corrispondente a quell'indice nell'array non è già stata colpita dalla funzione aiuto (ha ancora colore di riempimento originale), ne cambia il colore di riempimento ad uno stabilito dal tema corrente applicato al gioco. Fatto questo i valori di *cnt* e *numAiuti* vengono rispettivamente incrementato e decrementato.

resizeCanvas()	Funzione di ridimensionamento dello *stage* affinché mantenga lo stesso rapporto tra base e altezza al variare delle dimensioni della finestra, permettendole, quindi, di essere liberamente ridimensionata senza precludere la possibilità di giocare. Per prima cosa viene salvato un puntatore al campo *container* dello *stage*, dopo di che da questo vengono ricavate le misure di base e altezza. Tali misure vengono rese *pari* al fine di evitare effetti di sfocatura al ridimensionamento e vengono applicate allo stage come suoi nuovi valori di altezza e larghezza. A questo punto vengono calcolati i rapporti tra la altezza attuale e quella contenuta nella variabile *SCENE_BASE_HEIGHT*, e tra la larghezza attuale e quella contenuta in *SCENE_BASE_WIDTH*¸ viene scelto il rapporto minore come nuovo fattore di scala per lo *stage* (in modo da garantire che non avvengano distorsioni dell'immagine) ed in fine vi viene applicato.

quit()	Funzione di terminazione della partita che si occupa del salvataggio come parametri dell'oggetto *sessionStorage* di tutti i valori di interesse e della notificazione all'utente della vittoria. Ultima operazione effettuata dalla funzione è quella del trasferimento del controllo dalla pagina di gioco *game.html* a quella di menù *index.html*.

Partendo dalla struttura della quinta di gioco, si ha come elemento di base l'oggetto *stage*, che costituisce la base del pannello e si appoggia, per come è stabilito dalla sintassi del framework utilizzato, sull'apposito `<div>` avente id *container* definito nel file *game.html*.

Stage fungerà da contenitore per i successivi oggetti generati.

In particolare, sono stati definiti due livelli di *"tele"* sovrapposte (oggetti *layer* e *tansLayer*): il primo ha lo scopo di contenere tutto ciò che comporrà lo sfondo della schermata di gioco, che rimarrà pressoché statico, mentre il secondo è concepito per ospitare gli oggetti che costituiscono i tan, che sono caratterizzati da un alto livello di interattività e richiedono maggiore dinamicità e frequenti operazioni di ridisegno del pannello.

Mantenendo separati i layer si è cercato di limitare l'impatto che le frequenti invocazioni dei metodi di ridisegno del canvas, a seguito dello spostamento delle tessere o della loro rotazione, hanno sulle prestazioni generali, specialmente per dispositivi non particolarmente potenti o aggiornati (se ne parlerà più approfonditamente nel paragrafo 3.3).

Gli oggetti che popolano il layer deputato al background sono il pannello per la selezione del colore di sfondo (*rect*), generato usando un oggetto della classe *Rect* fornito dal framework e settandone altezza e larghezza ai valori posseduti dallo *stage*, e le sette tessere usate per produrre l'immagine di sfondo che costituirà il target da replicare (*squareSol*, *triangle1Sol*, *triangle2Sol*, *triangle3Sol*, *triangle4Sol*, *triangle5Sol*, *parallelogramSol*), generate come oggetti della classe *Line* offerta da KineticJS e memorizzate nell'array *sol*.

Si è deciso di utilizzare tessere gemelle dei tan a disposizione del giocatore per due motivi principali: per prima cosa questa modalità permette di fornire uno strumento per la rappresentazione di ogni soluzione indipendentemente da quale essa sia, possibilità che non sarebbe stata offerta disegnando le immagini come unici oggetti, inoltre lo stesso strumento risulta utile anche nella realizzazione del meccanismo di aiuto, in quanto permette, quando richiesto, di cambiare il colore di riempimento di una delle tessere di soluzione, rendendo nota la sua posizione al giocatore, che potrà, così, posizionare con certezza il tan corrispondente.

Si è ritenuto, pertanto, che questo duplice vantaggio compensasse il maggior livello di complessità introdotto dalla gestione di sette oggetti ulteriori in un pannello di gioco che già rischia di risultare un po' congestionato.

L'ultimo elemento a comparire nel layer deputato allo sfondo è, solo per il tema ad argomento zombie, un secondo oggetto di classe *Rect* (*rect2*) definito nel file *themes.js*, che fungerà da base per l'applicazione della texture dell'acqua usata per rappresentare il mare in cui galleggiano le zattere/tessere.

Passando al livello destinato ad ospitare le tessere, gli unici oggetti che compaiono sono, appunto, quelli memorizzati nell'array *tans*.

Essi sono generati in modalità identiche a quelli destinati alla rappresentazione dell'immagine obiettivo.

Tutti i 14 oggetti *Line* hanno valori settati solo per i campi indipendenti dal tema prescelto (ad esempio i valori delle coordinate *x* e *y* di posizionamento sul piano, o di quelle di ogni vertice, salvate come coordinate relative alla posizione indicata dalle prime), mentre viene demandato alle funzioni di applicazione del tema il settaggio dei rimanenti campi ai valori desiderati per il tema particolare.

```
var squareTan = new Kinetic.Line({
            points: [0 * shapePixel, 0 * shapePixel,
              10 * shapePixel, 0 * shapePixel,
              10 * shapePixel, 10 * shapePixel,
              0 * shapePixel, 10 * shapePixel],
            closed: true
});
```

L'applicazione del tema, però, potrebbe arrivare ad avere effetti pesanti sugli oggetti contenuti nell'array *tans*, che potrebbero arrivare anche a cambiare tipo: mentre il tema base lascia inalterata la classe degli oggetti tessera e si limita a settare i campi

lasciati liberi all'atto della loro creazione, il tema zombie crea gruppi di figure composti dai sistemi tessera-zombie, e li sostituisce nell'array alle semplici tessere.

Definizione di base del gruppo:

```
var squareGroup = new Kinetic.Group({
    x: 25 * shapePixel,
    y: 22 * shapePixel,
    draggable: true,
    offsetX: 5 * shapePixel,
    offsetY: 5 * shapePixel
});
```

Definizione dell'immagine da applicare allo zombie:

```
var imgZombie = new Image();
imgZombie.src = 'images/cartoon-zombie.gif';
```

Definizione dell'oggetto zombie:

```
var zombie0 = new Kinetic.Line({
    points: [0 * shapePixel, 0 * shapePixel,
             10 * shapePixel, 0 * shapePixel,
             10 * shapePixel, 10 * shapePixel,
             0 * shapePixel, 10 * shapePixel],
    fillPatternImage: imgZombie,
    fillPatternOffsetX: -15 * shapePixel,
    fillPatternOffsetY: -10 * shapePixel,
});
```

Aggiunta di tutti gli elementi tessera e di tutti gli elementi zombie ai rispettivi gruppi:

```
for (var i = 0; i < groups.length; i++) {
    groups[i].add(tans[i]);
    groups[i].add(zombies[i]);
}
```

Indipendentemente dalla natura degli elementi definiti negli array *tans* e *sol*, però, le funzioni di applicazione del tema, dopo aver applicato le modifiche stabilite, li inseriscono nei layer di competenza (*tansLayer* per gli elementi di *tans* e *layer* per quelli di *sol*) invocandone i metodi *add*, e se tutto è stato progettato in maniera coerente il sistema funzionerà senza inconvenienti e senza richiedere alcun intervento al di fuori della funzione di applicazione del tema.

Ultima invocazione necessaria alla creazione della struttura della quinta di gioco è quella del metodo *add* di *stage* per inserirvi in ordine i livelli *layer* e *tansLayer*.

Dopo aver definito tutti gli oggetti necessari all'esecuzione del gioco ed aver invocato le funzioni di applicazione del tema, il codice prevede che vengano definite le opportune politiche di gestione degli eventi per gli elementi che lo richiedono.

Per ogni tessera del gioco a disposizione dell'utente vengono definiti gli *action listener* deputati alla gestione delle condizioni di *trascinamento, selezione* (evento *mousedown*), e *singolo click/tocco*.

In particolare, al termine del *drag&drop* i valori delle coordinate della tessera al momento del rilascio vengono normalizzati ad un multiplo di quello inserito nella variabile *shapePixel* ed impostati come nuove coordinate di partenza per l'oggetto su cui è stato lanciato l'evento.

Non appena una tessera viene selezionata, invece, l'*action listener* collegatole per la gestione dell'evento *mousedown* la porta in primo piano invocando la funzione *moveToTop()* offerta da KineticJS.

Da ultimo, quando su una tessera viene effettuato un *click/tap* singolo l'*action listener* di gestione ne provoca la rotazione di 45° in senso orario invocando su di essa il metodo *rotate(deg)*, fornito dal framework, con argomento *45*.

Solo per la tessera a forma di parallelogramma, poi, essendo l'unica non simmetrica, viene definito un gestore per l'evento di *doppio click/tap* che ne inverte la scala lungo l'asse delle ascisse, ribaltandola, di fatto, rispetto a quello delle ordinate[1].

Ultimo *action listener* previsto dal codice è quello collegato all'oggetto *window*, che modella la finestra del browser, per intercettare l'evento di ridimensionamento e lanciare la funzione *resizeCanvas* (funzione di supporto di cui ci occuperemo nelle righe immediatamente successive) al fine di ridimensionare contestualmente anche l'elemento *stage* e tutti i suoi nodi figli.

L'ultima sezione del codice è, infine, quella destinata alla definizione delle funzioni di servizio *stampa(), controllo(), helpReq(), resizeCanvas()* e *quit()*.

La funzione *stampa()* è stata implementata ed utilizzata per rendere più semplice l'opera di definizione delle coordinate relative alle soluzioni, anche se non è strettamente necessaria allo svolgimento della partita. Utilizzando un ciclo *for* per recuperare i valori delle coordinate sugli assi di ascisse e ordinate e gli angoli di rotazione assunti da tutte

[1] Con asse delle ascisse si vuole intendere quello parallelo alle basi (i lati maggiori) e come asse delle ordinate quello loro perpendicolare.

le tessere dell'array *tans* al momento in cui viene invocata, la funzione normalizza le coordinate in modo che esprimano in termini multipli di *shapePixel* lo scostamento dal centro dello *stage* (normalizzato anch'esso), e stampa tutti i valori su console in formato già valido per l'inserimento negli array usati per la disposizione delle tessere soluzione e per il controllo di validità, lasciando a chi volesse inserire un nuovo elenco di coordinate solo il compito di copiare e incollare i valori dalla console del browser al file *levels.js*.

La normalizzazione apportata è stata concepita in un contesto in cui non si era ancora deciso di fissare il rapporto tra le dimensioni ad un valore fisso, ma si presta ad essere un elemento di robustezza per eventuali sviluppi futuri che prevedano diverse implementazioni della gestione della scala del canvas, consentendo, magari, la gestione di più formati.

Anche in questo caso le soluzioni proposte rimarrebbero valide a patto di fornire valori adeguati per l'identificazione di un centro normalizzato dello *stage* ed una unità di base in *shapePixel*.

Passando alla funzione *controllo()*, si deve dire che essa è uno degli elementi più importanti del codice, poiché costituisce il meccanismo di convalida che permette di stabilire quando il livello è stato superato e passare, quindi, in uno stato di terminazione della partita o, se necessario, di preparazione di un nuovo livello.

```
function controllo() {
            var check = false;
            for ( var i=0; i < solCheck[imageNum].length; i++) {
                  if
                  ((solCheck[imageNum][i][0] * shapePixel) + xCent
                        == tans[0].x() &&
                  (solCheck[imageNum][i][1] * shapePixel) + yCent
                        == tans[0].y() &&
                  (solCheck[imageNum][i][2]) % 90
                        == (tans[0].rotation() % 90) &&

                  ((
                  solCheck[imageNum][i][3] * shapePixel + xCent
                        == tans[1].x() &&
                  solCheck[imageNum][i][4] * shapePixel + yCent
                        == tans[1].y() &&
                  solCheck[imageNum][i][5] % 360
                        == tans[1].rotation() % 360 &&

                  solCheck[imageNum][i][6] * shapePixel + xCent
                        == tans[2].x() &&
                  solCheck[imageNum][i][7] * shapePixel + yCent
                        == tans[2].y() &&
```

```
solCheck[imageNum][i][8] % 360 ==
tans[2].rotation()%360
) ||

(
solCheck[imageNum][i][3] * shapePixel + xCent
      == tans[2].x() &&
solCheck[imageNum][i][4] * shapePixel + yCent
      == tans[2].y() &&
solCheck[imageNum][i][5] % 360
      == tans[2].rotation() % 360 &&

solCheck[imageNum][i][6] * shapePixel + xCent
      == tans[1].x() &&
solCheck[imageNum][i][7] * shapePixel + yCent
      == tans[1].y() &&
solCheck[imageNum][i][8] % 360
      == tans[1].rotation() % 360
)) &&

((
solCheck[imageNum][i][9] * shapePixel + xCent
== tans[3].x() &&
solCheck[imageNum][i][10] * shapePixel + yCent
      == tans[3].y() &&
solCheck[imageNum][i][11] % 360
      == tans[3].rotation() % 360 &&

solCheck[imageNum][i][12] * shapePixel + xCent
      == tans[4].x() &&
solCheck[imageNum][i][13] * shapePixel + yCent
== tans[4].y() &&
solCheck[imageNum][i][14] % 360
      == tans[4].rotation() % 360
) ||

(
solCheck[imageNum][i][9] * shapePixel + xCent
      == tans[4].x() &&
solCheck[imageNum][i][10] * shapePixel + yCent
      == tans[4].y() &&
solCheck[imageNum][i][11] % 360
      == tans[4].rotation() % 360 &&

solCheck[imageNum][i][12] * shapePixel + xCent
      == tans[3].x() &&
solCheck[imageNum][i][13] * shapePixel + yCent
      == tans[3].y() &&
solCheck[imageNum][i][14] % 360
      == tans[3].rotation() % 360
)) &&

solCheck[imageNum][i][15] * shapePixel + xCent
      == tans[5].x() &&
solCheck[imageNum][i][16] * shapePixel + yCent
      == tans[5].y() &&
solCheck[imageNum][i][17] % 360
      == tans[5].rotation() % 360 &&

solCheck[imageNum][i][18] * shapePixel + xCent
      == tans[6].x() &&
```

```
                solCheck[imageNum][i][19] * shapePixel + yCent
                      == tans[6].y() &&
                solCheck[imageNum][i][20] % 180
                      == tans[6].rotation() % 180 &&
                solCheck[imageNum][i][21]
                      == tans[6].scaleX()
                )

                {
                 check = true;
          rect.fill('#'+(Math.random()*0xFFFFFF<<0).toString(16));
                 layer.draw();
                break;
                }
          }
          return check;
    }
```

Questa funzione non prevede alcun valore in input e restituisce in output un valore booleano *check*.

Tale valore è inizializzato a *false* e corrisponde all'avvenuto riscontro di una corrispondenza tra le posizioni e rotazioni attuali delle tessere ed i valori contenuti in uno degli array che codificano le soluzioni ammissibili per la specifica immagine di target.

In particolare, all'interno di un ciclo *for* che scandisce tutti gli elementi dell'array viene innestato una grossa condizione *if* che controlla che i valori di ascissa, ordinata e rotazione di ogni tessera corrispondano a tre a tre con i valori contenuti nell'array in esame.

Ad esempio, gli indici da 0 a 2 dovranno corrispondere con i valori *x*, *y* e *rotation* della tessera quadrata, quelli da 3 a 5 con gli stessi valori di una di quelle triangolari di dimensioni maggiori, e così via fino al parallelogramma, che viene controllato per ultimo e per il quale è computato nel confronto anche il valore di scala lungo l'asse delle ascisse.

Tale campo può avere valori +1 o -1, e permette di avere il controllo anche sul fatto che sia richiesto o meno il ribaltamento della tessera per fornire una soluzione corretta.

Se la condizione di confronto non è verificata l'indice di scorrimento dell'array delle soluzioni viene incrementato e si procede con l'analisi del suo elemento successivo, mentre qualora la soluzione proposta sia compatibile con una di quelle contenute nell'array, il sistema associa alla variabile *check* il valore *true*, cambia il colore di riempimento dell'oggetto *rect* portandolo ad un colore scelto in modalità pseudo-casuale per aumentare l'effetto grafico associato alla vittoria, e ridisegna il layer di sfondo

(*layer*), dopodiché interrompe il ciclo invocando la funzione *break*, ed infine fornisce in output il valore di *check* precedentemente settato.

Si riporta che per la gestione della presenza di tessere identiche tra loro, e la conseguente ininfluenza di un eventuale scambio di posizione tra esse per la definizione di una soluzione corretta non si sono inserite nuove soluzioni nell'array, ma si è fatto riferimento all'inserimento di clausole *or* nell'elenco di confronti collegati da *and* secondo uno schema concettuale del tipo:

if (quadrato AND ((triangoloGrande1 AND triangoloGrande2) OR (triangoloGrande2 AND triangoloGrande1)) AND ((triangoloPiccolo1 AND triangoloPiccolo2) AND (triangoloPiccolo2 AND triangoloPiccolo1)) AND triangoloMedio AND parallelogramma).

Altro elemento di particolarità è quello relativo alla memorizzazione delle coordinate.

Poiché la formattazione delle coordinate nell'array è quella particolare definita in relazione alla funzione *stampa()* (le coordinate sono salvate come distanza dal centro dello stage normalizzato a *shapePixel* e vengono a loro volta normalizzate a multipli di *shapePixel*), per poterle confrontare con i valori di ritorno delle funzioni *x()* ed *y()*, che esprimono le normali coordinate in pixel partendo dall'origine degli assi, la funzione moltiplica per *shapePixel* i valori letti dall'array, e vi aggiunge il valore di *xCent* o *yCent* a seconda che si tratti di una ascissa o di una ordinata.

Ultimo elemento degno di attenzione riguarda la gestione delle rotazioni.

I campi *rotation* degli oggetti forniti dal framework utilizzato non prevedono range di valori periodici con periodo 360, come sarebbe comodo per gestire i gradi di rotazione nel contesto applicativo, di conseguenza la rotazione di un oggetto fatto ruotare completamente due volte non risulterà, per esempio, complessivamente nulla, ma pari a 720°.

Per gestire un numero indefinito di rotazioni delle tessere da parte dell'utente e permettere, nel contempo, un confronto dei valori dei loro campi *rotation* con i numeri fissi, e generalmente minimi, con cui si sono rappresentate le rotazioni nella soluzione, si sono dovute apportare normalizzazioni di tali valori: si è provveduto all'eliminazione di ogni multiplo di 360° (operazione *rotation%360*) per tutte le tessere eccetto il

parallelogramma, per la forma del quale bastano 180°, ed il quadrato, che ne richiede solo 45°.

Per quanto riguarda la funzione *helpReq(),* essa fa uso delle due variabili *cnt* e *numAiuti* definite in testa al file, rappresentanti rispettivamente il numero di aiuti utilizzati nel livello corrente e quelli disponibili in generale al giocatore.

Se il numero di aiuti utilizzati è inferiore ad una quota stabilita (in questo caso il numero delle tessere) e contemporaneamente il numero di aiuti disponibili è superiore a 0, il sistema sceglie in modalità pseudo-casuale (usando la funzione *rand*) un numero tra 0 e 7, e controlla se la tessera corrispondente al valore contenuto nell'array *sol* alla posizione indicata da quel numero è già stata colpita da una precedente invocazione ispezionandone il campo relativo al colore riempimento.

In caso la tessera risulti già colpita viene calcolato un nuovo indice secondo le stesse modalità, e tale ciclo viene reiterato fino a quando il controllo non riporti una tessera intatta.

Quando finalmente questo avviene, il campo relativo al colore di riempimento viene settato ad un valore stabilito dal tema e viene invocata la funzione di ridisegno del livello di sfondo (*layer*).

Come ultime operazioni si decrementa il valore della variabile contenente il numero di aiuti disponibili, si incrementa quello della variabile contenente il numero di aiuti utilizzati e si invoca la funzione *numToImage(num)* passandole in input il nuovo numero di aiuti residui al fine di aggiornare l'immagine relativa al loro contatore.

Si passa, ora, alla funzione *resizeCanvas().*

Questa funzione fa uso delle variabili *SCENE_BASE_WIDTH* e *SCENE_BASE_HEIGHT*, che assume come valori base per la definizione di un formato per la finestra di gioco che dovrà rimanere invariato al variare delle sue dimensioni.

Come prima cosa la funzione recupera i valori di larghezza e altezza del *container* di *stage*, li rende numeri pari decrementandoli in caso non lo siano e li applica a *stage* come valori dei suoi campi *width* ed *height*.

Questo passo viene compiuto al fine di prevenire effetti di sfocatura che i valori dispari potrebbero introdurre al momento del ridimensionamento del pannello.

Dopo questa operazione vengono calcolati i rapporti tra le dimensioni appena imposte a *stage* ed i valori corrispondenti salvati nelle variabili *SCENE_BASE_WIDTH*

e *SCENE_BASE_HEIGHT*, e viene scelto il minore tra i due come fattore di scala da applicare ad entrambe le dimensioni dell'oggetto.

In questo modo viene conservato il formato e vengono evitate eventuali distorsioni, benché il prezzo da pagare sia quello di dimensioni generalmente inferiori a quelle che si avrebbero occupando tutta l'area possibile applicando fattori di scala diversi per altezza e larghezza.

L'ultima operazione effettuata prima del ridisegno della finestra è la centratura dello *stage* ridimensionato.

Il valore della ascissa di base dello *stage* viene impostato alla metà della larghezza del *container*, cui viene sottratta la metà del valore contenuto in *SCENE_BASE_WIDTH* moltiplicato per il fattore di scala applicato a *stage*, e similmente il valore dell'ordinata viene calcolato applicando gli stessi ragionamenti alle altezze.

In chiusura della trattazione del codice di *engine.js* trattiamo brevemente la funzione *quit()*.

Questa funzione viene richiamata dall'*action listener* del bottone di controllo in *game.js* qualora la verifica della soluzione proposta dia esito positivo, ed è caratterizzata da un codice piuttosto semplice.

Per prima cosa vengono salvati come attributi dell'oggetto *sessionStorage* i valori di *nome*, *cognome* e *id*, ed il *punteggio aggiornato*, calcolato come la somma del punteggio iniziale con il quantitativo di punti rimasto al termine del livello, dopodiché viene mostrata una finestra di *alert* in cui ci si complimenta con il giocatore per la vittoria, e quando viene premuto il tasto *Ok* il sistema carica nuovamente la finestra di menù codificata in *index.html*, ed il gioco ricomincia.

levels.js

E' il file JavaScript contenente la logica secondo la quale viene selezionata l'immagine di target da riprodurre nella partita e vengono settati i valori necessari al controllo della correttezza di quanto realizzato dall'utente.

Sulla base degli stessi valori avviene anche la disposizione degli oggetti creati in *engine.js* deputati alla costruzione della soluzione (quelli contenuti nell'array *sol*).

Ognuna delle soluzioni valide per ogni immagine target è memorizzata nel file sotto forma di un array contenente le coordinate *x* e *y* sul piano, calcolate come multipli dell'unità di base *shapePixel* rispetto a quelle del centro, anch'esse normalizzate a

multipli di *shapePixel* (*xCent* e *yCent* definite in *engine.js*), e l'angolo di rotazione di ognuna delle sette tessere che compongono la soluzione.

In più, solo per il parallelogramma, viene inserito in ultima posizione il valore del fattore di scala sull'asse delle ascisse (+1 o -1), che permette di capire se per riprodurre correttamente la soluzione il parallelogramma deve essere ribaltato, dal momento che esso è l'unica figura non simmetrica tra quelle che compongono le tessere.

Poiché esistono, in generale, *n* soluzioni valide per ogni immagine, queste n soluzioni sono contenute in un array apposito, che è, quindi, un array di array.

Un ulteriore livello di raggruppamento, poi, si ha quando si vuole selezionare un pool di immagini possibili per un livello di difficoltà che sia sottoinsieme dell'universo delle immagini proposte.

Questa situazione comporta un terzo ed ultimo livello di raggruppamento, che è stato gestito tramite un ulteriore array contenuto nella variabile *solCheck*, che conterrà le coordinate e l'angolo di rotazione di tutte le tessere per ogni soluzione possibile di ogni immagine relativa ad un certo livello di difficoltà.

In base al valore di difficoltà impostato, salvato nella variabile *difficulty*, viene settato il valore dell'array *solCheck* ad uno degli elenchi precompilati di coordinate e rotazioni relativi a tutte le immagini valide per quel livello di difficoltà.

A questo punto viene impostato il valore della variabile *imageNum* ad un numero scelto in base ad una funzione random tra 0 e la lunghezza di *solCheck*: questo valore indica, ora, quale tra le immagini possibili è stata scelta per la partita, e quindi, usandolo come indice per identificare un particolare elemento di *solCheck*, a quale pool di soluzioni fare riferimento per la rappresentazione della figura e per il controllo della correttezza di quanto prodotto dal giocatore.

themes.js

E' il file deputato alla gestione dei temi.

Nella sezione iniziale vengono definite le variabili globali di cui si farà uso in tutto il sistema per l'applicazione del tema, a partire da dieci variabili deputate alla memorizzazione dei *path* di reperimento delle immagini usate per rappresentare le cifre di punti e aiuti in *game.html*, per proseguire con tutte quelle deputate alla gestione di colori, opacità e contorni di ogni tessera.

Oltre alle variabili, il file contiene le funzioni prettamente deputate all'applicazione del tema, la parte di logica più importate del file, e le funzioni di supporto loro affiancate.

In particolare vengono introdotte le funzioni di supporto destinate all'assegnamento delle variabili per i *path* alle immagini cifra, e per il disegno di una griglia sullo sfondo del pannello di gioco, funzionalità implementata, ma non utilizzata soprattutto a causa di problemi di performance sui dispositivi meno potenti.

Essendo definiti solo due temi, uno di tipo minimale che si è concepito come tema di base, ed uno più articolato sviluppato su argomento zombie, le funzioni di applicazione di temi sono solo le due loro destinate, ma la definizione di nuovi temi per sviluppi futuri è semplice come la scrittura di una nuova funzione di applicazione concettualmente simile a quelle già inserite.

Per come è concepito il motore dell'applicazione definito in *engine.js*, una parte molto importate della definizione degli oggetti che popoleranno il pannello di gioco è affidata alle funzioni di applicazione del tema, senza l'intervento delle quali non sarebbe possibile giocare alcuna partita.

Per quel che riguarda la funzione di applicazione del tema nullo, *noneTheme()*, essa setta i valori di colori, posizioni ed offset per l'individuazione dei centri di tutte le tessere e del pannello di sfondo, in modo tale che il risultato sia una finestra di gioco a fondo bianco che presenti una figura obiettivo non interattiva di colore grigio, e sette tan di colore azzurro bordati di blu per riprodurla.

Non si è voluto settare valori quali la responsività al trascinamento e l'offset per le tessere all'interno del file *engine.js*, in cui esse vengono definite, per lasciare la massima libertà nella gestione dei temi, delegando al file di definizione solo il settaggio dei parametri indispensabili e sempre validi, come le coordinate relative dei vertici rispetto alle coordinate di posizione e la posizione iniziale stessa.

Si è immediatamente tratto vantaggio della libertà concessa alla definizione di temi con la seconda funzione di applicazione, quella per il tema ad argomento zombie: *zombieTheme()*.

Per questo tema ci si è immaginati che le tessere del Tangram siano in realtà zattere di legno dall'aspetto simile a vecchi bancali, e che sette zombie si trovino a montarle per attraversare un tratto di mare.

A questo punto non risultano più sufficienti gli oggetti tessera, ma si rende necessario organizzare un sistema più complesso, che coinvolge le sette tessere, altrettante immagini per gli zombie, ed un egual numero di oggetti che fungano da base per l'applicazione di tali immagini, in modo da renderle oggetti concreti all'interno del pannello di gioco.

Per fare questo la funzione *zombieTheme()* setta tutte le variabili globali introdotte all'inizio ai valori stabiliti per il tema, dopodiché crea una variabile locale deputata al fattore di scala per le figure degli zombie affinché possano essere alloggiate all'intero dell'area delle tessere senza dare problemi di alcun tipo, e genera gli oggetti di tipo *group* che alla fine dell'esecuzione conterranno i sistemi zombie-zattera.

Vengono, poi, generati nove oggetti immagine: sette di zombie, uno per la texture a tavole di legno delle zattere, ed uno contenente la texture dell'acqua da usare per rappresentare il mare.

Altri otto oggetti vengono, poi, generati per fungere da substrati per l'applicazione delle immagini appena citate, eccezion fatta per la texture delle zattere, che andrà applicata agli oggetti tan già definiti: sono oggetti appartenenti alla classe *Rect* fornita dal framework KineticJS.

Gli oggetti destinati a rappresentare i sette zombie vengono, infine, inseriti in un array *zombies* per facilitarne l'indirizzamento e permetterne la gestione facendo uso di cicli *for*.

In maniera simile alla funzione di applicazione del tema neutro, vengono settati i parametri di posizione e centro per le tessere a disposizione del giocatore e di quelle di soluzione, e tramite cicli *for* i tan vengono aggiunti ai gruppi creati in precedenza in accoppiamento con gli oggetti zombie, così da costituire gruppi solidali zombie-zattera.

Ultime istruzioni contenute nella funzione sono la sostituzione degli oggetti gruppo a quelli tessera nell'array *tans* generato da *engine.js*, l'aggiunta di tutti gli elementi ai layer corrispondenti, e l'invocazione della funzione di supporto *digitAssign ()*, che assegna alle variabili per la rappresentazione delle cifre i path relativi alle immagini selezionate per il tema, passati in input.

3.2 Fasi di gioco e casi d'uso

Il flusso del gioco è piuttosto lineare: dapprima al giocatore viene proposto un menù in cui è richiesta la selezione del livello di difficoltà e del tema, e la compilazione

di quattro campi: il *nome*, il *cognome*, il *numero dei punti inizialmente posseduti*, ed uno relativo ad un eventuale *codice identificativo*, che vengono richiesti nella versione demo del programma, ma a regime saranno forniti da altre applicazioni o recuperati da un database dedicato.

Una volta compilato il menù, l'utente può accedere al gioco tramite la pressione del bottone *Inizia* (l'unico bottone presente nella schermata iniziale).

A questo punto viene caricata la schermata di gioco vera e propria, che presenta all'utente tre tipi di interazione possibile:

- spostare e ruotare le sette tessere del gioco interagendo con loro via mouse se l'applicazione è lanciata su browser, o tramite il touchscreen se lanciata su dispositivo mobile;
- premere il bottone di richiesta di aiuti, che rende mostra la posizione che deve essere occupata da una delle tessere per completare la soluzione;
- premere il bottone che lancia il controllo della soluzione proposta, cosa che ha senso fare, però, solo quando si ritiene di aver risolto il livello.

Una volta che l'utente ha completato correttamente la soluzione ed ha invocato la funzione di controllo, gli viene mostrata una finestra di congratulazioni, e poi il controllo viene riportato alla pagina del menù iniziale.

Inutile dire che in ogni momento, sia che si esegua l'applicazione su browser, sia che la si esegua su dispositivo mobile, è possibile utilizzare gli appositi meccanismi per il caricamento della pagina precedente come meccanismo di "*indietro*", allorché se la pagina attuale è quella di gioco codificata in *game.html* il controllo viene riportato a quella di menù, mentre se la pagina caricata è quella di menù rispettivamente si verificano il passaggio in background dell'applicazione o il caricamento della pagina visitata precedentemente a seconda del fatto che si esegua come app per dispositivo mobile o web browser.

Si presentano, di seguito, i diagrammi dei casi d'uso e degli stati che rappresentano il sistema, ed alcuni snapshot relativi alle fasi di gioco.

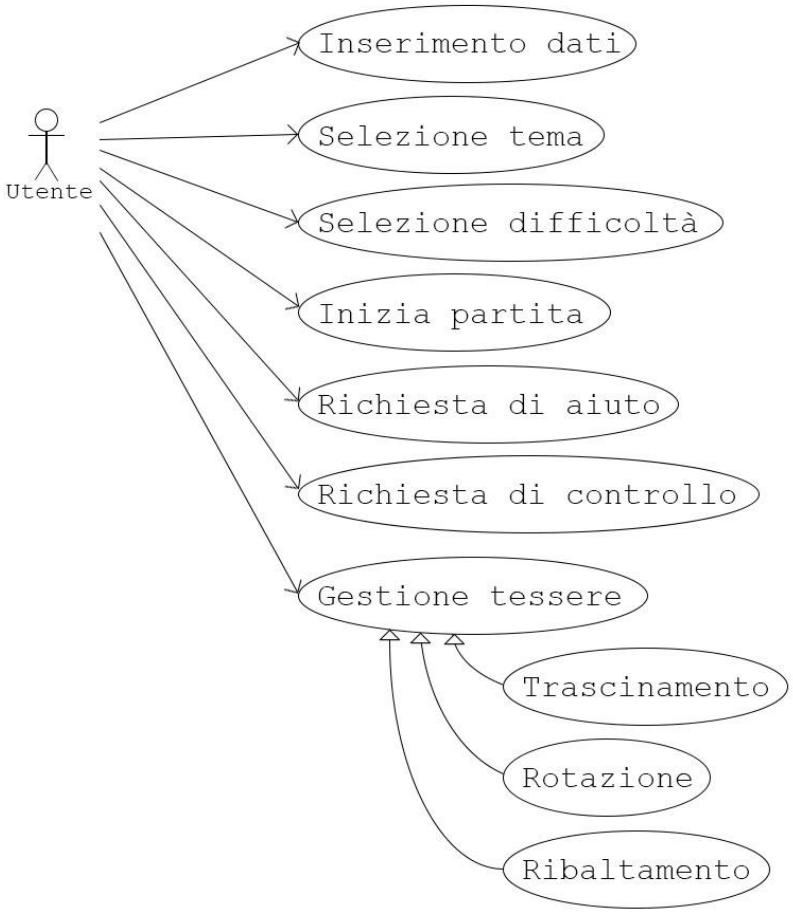

Figura 3.4: Diagramma dei casi d'uso dell'applicazione

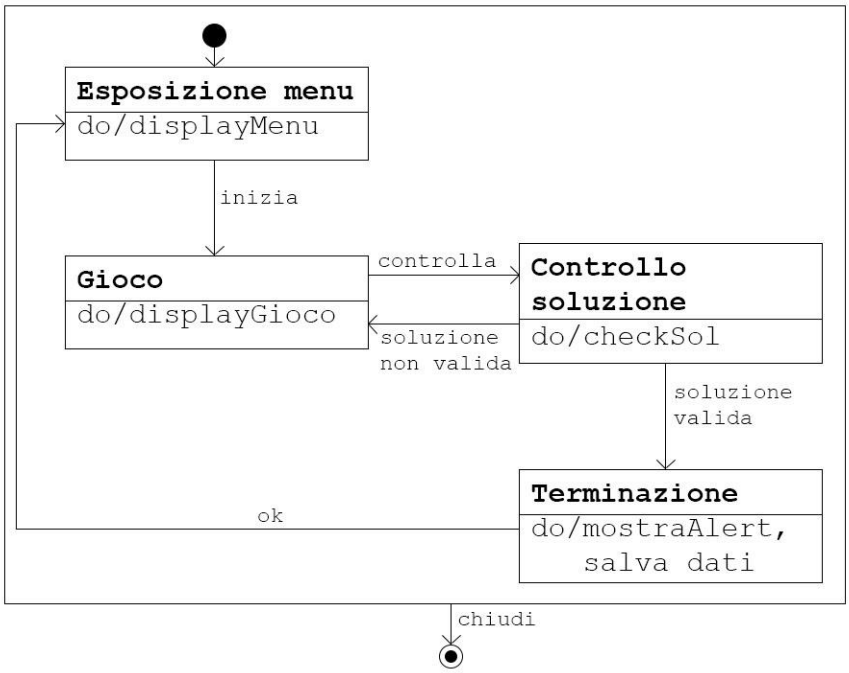

Figura 3.5: Diagramma degli stati dell'applicazione

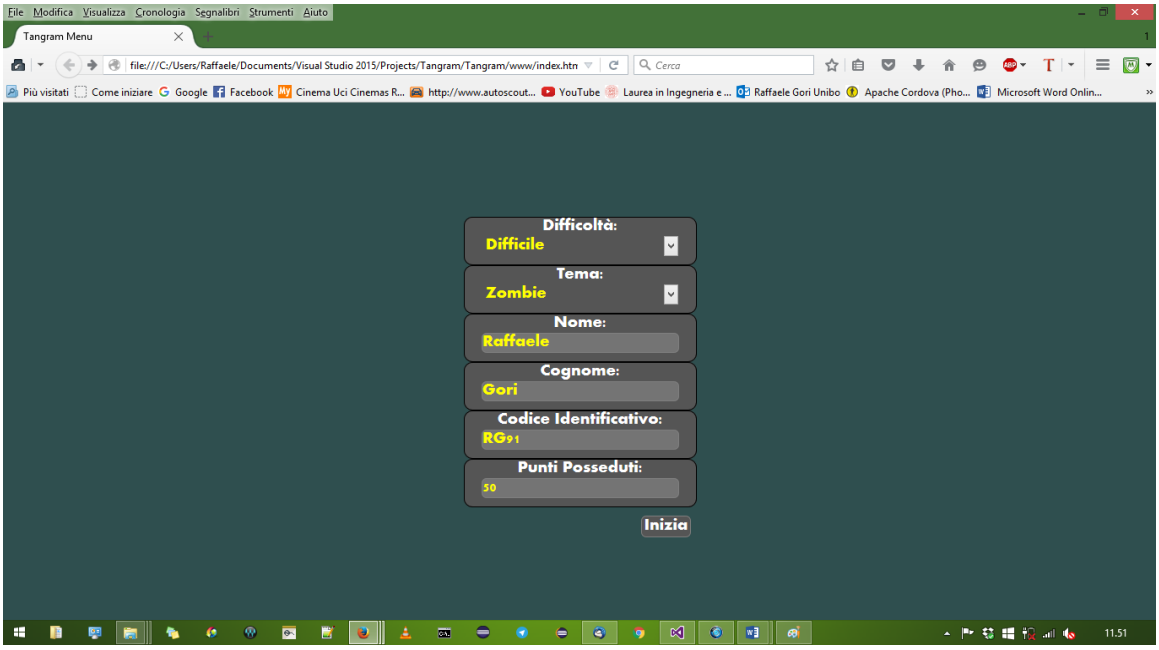

Figura 3.6: Inserimento dei dati e selezione di tema e livello di difficoltà

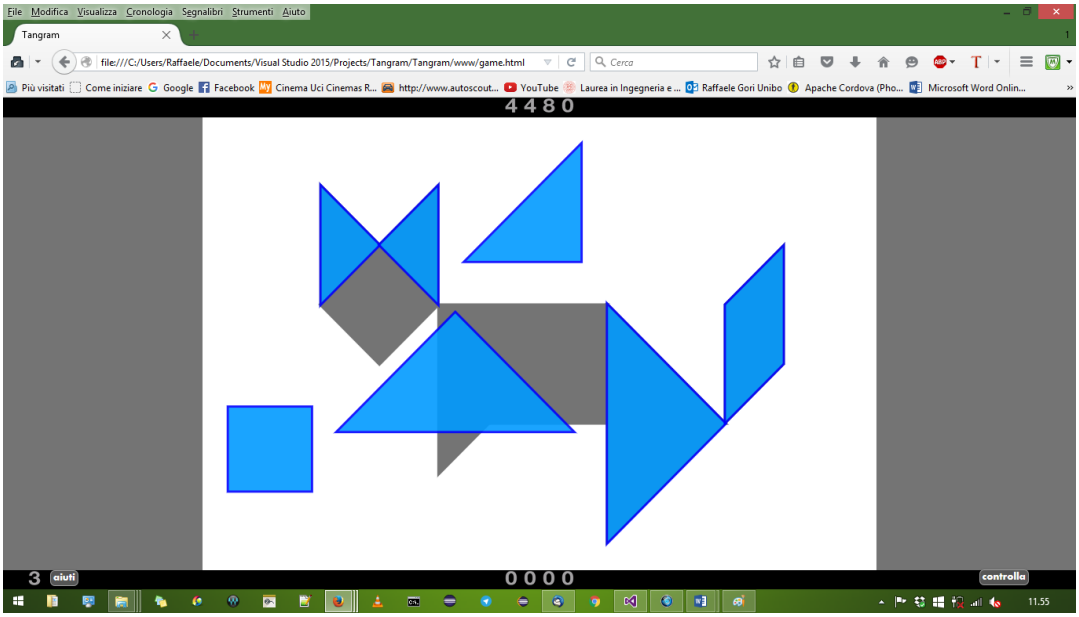

Figura 3.7: Momento di gioco con disposizione parziale delle tessere

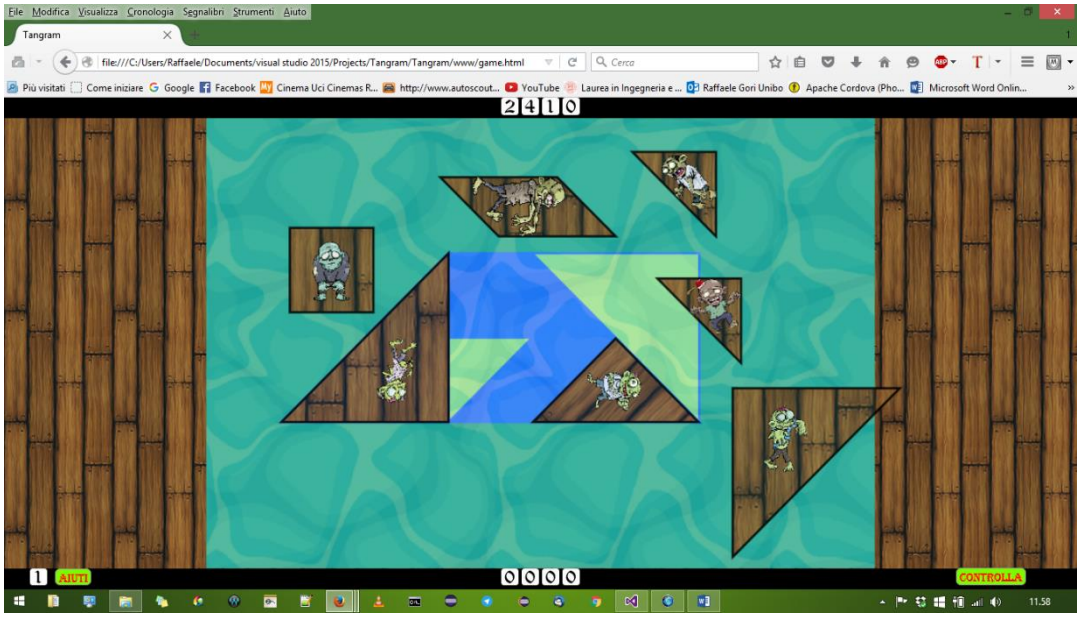

Figura 3.8: Esempio di richiesta di aiuti

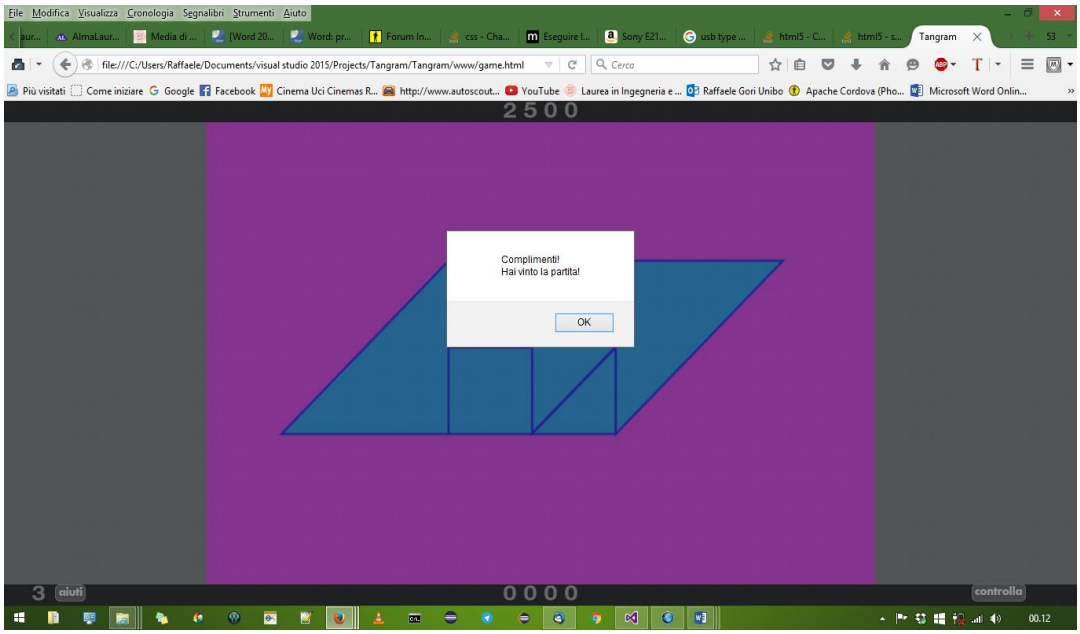

Figura 3.9: Schermata di vittoria

3.3 Test effettuati, bug riscontrati, soluzioni apportate e miglioramenti possibili

I test che sono stati condotti sulla applicazione oggetto di questa tesi si sono basati sull'uso dei seguenti strumenti:

- Ambiente di sviluppo Microsoft Visual Studio Community 2015, nella versione 14.0.23107.0D14REL
- Browser Mozilla Firefox, nella release 41.0.0.5784

- Browser Microsoft Internet Explorer, nella release 11.0.9600.17840
- Browser Google Chrome, nella release 46.0.2490.86
- Smartphone Sony XPeria E4 Dual e2115
- Emulatore di dispositivi mobili Apache Ripple in sinergia con l'espansione Apache Cordova di Visual Studio.

Nonostante si ritenga che sarebbe stato quantomeno utile, se non necessario, condurre test più approfonditi che coinvolgessero un numero decisamente superiore di dispositivi mobili e che coprissero un insieme quanto più vasto e variegato possibile di marche, caratteristiche hardware, dimensioni ed età (l'unico device fisico utilizzato è uno smartphone con schermo da 5" ed 1GB di RAM prodotto nel 2015), nei limiti dei test condotti è stato comunque possibile rilevare qualche elemento di criticità e qualche debolezza che potrebbero fornire spunti per elaborazioni e sviluppi futuri.

Per prima cosa i test condotti sul nucleo dell'applicazione utilizzando i web browser hanno evidenziato una generale compatibilità di quanto prodotto con tali programmi, ad eccezione del solo Internet Explorer, che mostra problemi nella gestione dell'oggetto *sessionStorage*, introdotto con il recente HTML5 ed usato per l'immagazzinamento temporaneo dei dati, quando le pagine non vengano ricevute da un server attraverso il protocollo *HTTP* [STA13].

Ad ogni modo non si ritiene che questa condizione sia preoccupante in vista di un ipotetico scenario di esecuzione web diverso da quello di test, poiché in tali condizioni il codice dell'applicazione risiederebbe su un server, e non sulla macchina locale, e di conseguenza anche Internet Explorer dovrebbe riuscire correttamente nell'esecuzione.

Non si sono, invece, riscontrati problemi di compatibilità nell'esecuzione dell'applicazione sui dispositivi mobili (fisici o emulati) testati, benché rimanga incognito il comportamento che l'applicazione potrebbe mostrare quando eseguita su dispositivi che usino IE come browser predefinito, dal momento che il funzionamento di Apache Cordova prevede l'utilizzo di una WebView per la presentazione di codice che risiede localmente sul dispositivo.

Altro elemento di criticità è la generale tendenza alla lentezza che l'applicazione mostra quando eseguita su dispositivi dalle scarse risorse come lo smartphone utilizzato.

Al fine di limitare il più possibile il numero di livelli sovrapposti e di elementi canvas istanziati dal framework, si era inizialmente organizzato il codice in modo tale che utilizzasse un solo layer per l'intero pannello di gioco; tuttavia tale approccio

provocava un drastico rallentamento dell'esecuzione, specialmente quando coniugato al tentato uso delle funzioni di disegno della griglia, illustrate poc'anzi, e al lancio della routine di controllo dopo ogni interazione con le tessere.

Il problema era reso tanto più grave dal fatto di dover gestire un oggetto aggiuntivo per ogni linea verticale ed orizzontale della griglia, e dal dover compiere un numero rilevante di confronti con cadenza decisamente elevata.

I tempi di risposta al trascinamento delle figure, in questa configurazione, si attestavano introno a qualche secondo (indicativamente tra 3 e 8 a seconda del numero di applicazioni aperte in background e del numero di input dati al touchscreen): tale condizione risultava palesemente incompatibile con i requisiti di interattività richiesti dalla tipologia di applicazione.

Per risolvere, o quantomeno tamponare il problema si è proceduto, allora, ad un'opera di riorganizzazione del codice e della struttura dell'applicazione, arrivando alla condizione attuale.

Si è operata una divisione degli elementi del pannello in due livelli, definendo un layer di sfondo ricco di elementi, ma pressoché statico, ed uno molto dinamico, ma destinato alla gestione dei soli sette oggetti tessera, e per tanto caratterizzato da oneri di gestione contenuti per il dispositivo.

Altri interventi volti ad una migliore fluidità di esecuzione sono stati la rinuncia al disegno della griglia ed il cambio di approccio nella gestione del controllo della soluzione.

Mentre nella concezione iniziale il controllo avveniva in corrispondenza di ogni evento di trascinamento, rotazione o ribaltamento delle tessere indipendentemente dal fatto che ci fossero o meno i presupposti per pensare che la soluzione proposta fosse corretta, ora la funzione è stata collegata alla pressione di un bottone da parte dell'utente.

Questo approccio rende più snella l'esecuzione, poiché sostituisce ad una politica simile a quella di *polling* una gestione del controllo più simile a quella basata sugli *interrupt*, ma presenta come contropartita una minor facilità di utilizzo del sistema da parte dell'utente, che sarà, ora, costretto ad invocare personalmente la funzione nel minore tempo possibile dopo il completamento della soluzione: pena il guadagno di un numero inferiore di punti.

In fase di test ci si è resi conto anche di una serie di inconvenienti dovuti alla grande variabilità nelle dimensioni degli schermi dei dispositivi dai quali l'applicazione dovrà poter essere eseguita.

Mentre lo schermo di un pc supporta bene anche risoluzioni elevate per il pannello del gioco, quando l'applicazione deve essere eseguita su un dispositivo con schermo di diagonale pari a 5", o addirittura inferiore, si verifica il fatto che le risoluzioni troppo alte comportano un rimpicciolimento eccessivo delle tessere, che invece devono rimanere sempre trascinabili, posizionabili e ruotabili con facilità.

Coniugando questa situazione con la scelta di gestire un formato singolo e fisso per il pannello di gioco, si è stati costretti a scegliere valori di risoluzione per il pannello piuttosto bassi (600x400) e fattori di scala per le tessere abbastanza elevati (*shapePixel* vale 7.5px), condizioni che non compromettono la qualità dell'immagine o la giocabilità sugli schermi più grandi, ma evidenziano la mancanza di una funzione di bounding per le tessere, che al momento possono superare i confini dell'area destinata al pannello di gioco, ed hanno tanto maggiori possibilità di trovarsi a farlo quanto più è limitato lo spazio di manovra a disposizione del giocatore.

L'implementazione della funzione di bounding viene proposta come spunto per sviluppi futuri, così come l'elaborazione dell'attuale approccio di gestione delle risoluzioni, permettendo non solo la gestione di un maggior numero di formati per il pannello di gioco, ma con essa anche una migliore occupazione dell'area dello schermo rispetto a quanto fatto dalla attuale funzione *resizeCanvas()*.

Parlando della disposizione delle tessere nel pannello di gioco si deve parlare anche dell'impossibilità di gestire in maniera totalmente precisa la loro giustapposizione utilizzando un effetto calamita, pressoché indispensabile per poter effettuare controlli precisi sulle posizioni in vista della validazione della soluzione.

Presentando angoli sempre multipli di 45°, le figure sono caratterizzate da lati espressi in termini di $\sqrt{2}$, cosa che ne rende impossibile il posizionamento e l'accostamento in maniera precisa all'interno di una griglia di posizioni possibili, definita sulla base di unità discrete e non infinitesimali: per questo motivo si è stati costretti a scegliere come base per la griglia un valore di compromesso, che non garantisce accostamenti esatti, ma, in unione a valori adeguati per lo spessore dei contorni permette una giustapposizione comunque soddisfacente.

Si è notato, ad ogni modo, che nessuna delle applicazioni incontrate sulla rete, che abbiano caratteristiche simili a quella presentata in questo progetto, garantisce un accostamento esatto, o se lo fa non fornisce alcun effetto calamita né controlli di validazione delle soluzioni proposte, elementi imprescindibili per contesti che richiedano la divisione del gioco in livelli, o che prevedano condizioni di terminazione vittoriosa della partita, come quello proposto dalla applicazione di livello superiore, di cui quella presentata è solo una sezione.

Si conclude la sezione ed il capitolo parlando delle difficoltà che si riscontrano nella gestione della tessera a forma di parallelogramma.

La forma della tessera, in una sfortunata concomitanza con le particolari dimensioni dei suoi lati in relazione all'unità di base della griglia, ha comportato l'assestamento del suo centro di rotazione ad una misura incompatibile con il posizionamento guidato dall'effetto calamita, generando una condizione in cui i movimenti complessi di rotazione, combinata con il ribaltamento, non consentono una soluzione di codice elegante che garantisca un movimento composto della tessera quando essa viene fatta ruotare dopo essere stata ribaltata.

Nel codice si presenta una duplice possibilità di gestione dell'elemento: una codifica elegante che genera una rotazione della tessera non gradevole a vedersi, ed una codifica più complicata che garantisce, però, un comportamento accettabile, lasciando ad eventuali sviluppi futuri di stabilire quale delle due politiche vada applicata, o se addirittura non convenga una rifattorizzazione di tutto il modello prestando maggiore attenzione alla posizione dei centri di rotazione delle figure.

Un ultimo elemento di imperfezione riscontrato è quello relativo alla gestione dei molteplici eventi che possono coinvolgere la tessera parallelogramma: l'unica per cui siano definiti eventi sia per il singolo che per il doppio *click/tap*.

Dopo aver constatato che una gestione sincrona di entrambi gli eventi è impossibile, se si vuole evitare di invocare il gestore del *click/tap* singolo anche ogni volta che si verifica l'evento di doppio *click/tap*, si è inserita la routine di gestione del primo caso all'interno di una funzione di ritardo, e se ne è condizionata l'esecuzione al valore di una variabile indicante l'evento di doppio *click/tap* settata a *true* all'inizio della routine di gestione dell'evento, e a *false* alla fine.

In questo modo la reazione al tocco singolo risulta leggermente ritardata, ma è possibile gestire disgiuntamente entrambi gli eventi con una struttura simile a quella dei semafori di mutua esclusione.

Conclusioni

Quello dell'applicazione di dinamiche ludiche all'apprendimento non è un campo semplice, poiché, specialmente se viene coinvolto l'uso di videogiochi e nuove tecnologie, richiede conoscenze ramificate in diversi ambiti, che spaziano dalla psicologia, alla matematica, alla capacità di progettazione e produzione di software.

Venendo a mancare forza sufficiente ad una di queste componenti si corre il rischio di produrre effetti anche contrari rispetto a quello prefissato, che è quello di fornire uno strumento che coniughi il divertimento con l'apprendimento, generando sistemi che non riescono ad incentivare né l'aspetto ludico, né quello educativo.

L'obiettivo di questo progetto era la realizzazione di una applicazione prototipo finalizzata ai dispositivi mobili che riproducesse in formato digitale e attraente l'antico gioco cinese del Tangram, usato per secoli come strumento di allenamento delle abilità logiche e della familiarità con concetti di geometria quali gli spostamenti rigidi di figure sul piano o i confronti di aree.

Finalità tecniche del progetto erano quelle di mantenere una separazione netta tra tutto ciò che riguardasse gli aspetti di presentazione ed il codice contenente la dinamica del gioco, fornendo, nel contempo, un pacchetto in grado di funzionare sia come software singolo, sia come modulo innestato all'interno di una applicazione multi-gioco, alla quale si doveva fornire una interfaccia il più semplice possibile, per garantire alti livelli di modularità generale.

Al fine di rivolgersi ad un vasto target di dispositivi differenti per tipo e caratteristiche, si è progettato il sistema secondo il modello della applicazione ibrida, che coniuga la semplicità e la generalità dell'implementazione tipiche della progettazione finalizzata al web con performance ed interfacciamento al dispositivo simili a quelli delle applicazioni di tipo nativo, permettendo, tra l'altro, di rilasciare una versione web del programma semplicemente estraendone il nucleo.

Il supporto ad un numero virtualmente infinito di temi è stato realizzato parametrizzando ogni aspetto relativo alla presentazione, e definendo in maniera fissa solo il numero minimo di elementi indispensabili all'esecuzione del programma, al fine di lasciare la massima libertà nella gestione dei temi a funzioni deputate.

Tali funzioni, oltre a poter settare i parametri stabiliti, possono creare al loro interno altri oggetti aggiuntivi e modificare liberamente il pannello di gioco.

Per quanto riguarda l'interfacciamento verso l'esterno e la modularità dell'applicazione, si è provveduto a scambiare con il sistema ospite solo un numero minimo di dati, che vengono richiesti in input e restituiti aggiornati in output, gestendo, per il resto, in maniera autonoma l'intera esecuzione.

L'applicazione prodotta è solo una versione prototipo, e come tale lascia inevitabilmente spazio a future migliorie, sia dal punto di vista delle prestazioni del codice in quanto software, sia da quello della effettiva efficacia come strumento di edutainment, ma mentre il primo aspetto può essere testato in laboratorio, per avere un feedback relativo al secondo occorrerà attendere l'esito della presentazione del gioco agli alunni delle classi medie cui tale applicazione sarà sottoposta nel corso del 2016 per una fase di test *"sul campo"*.

Ad ogni modo il software è stato prodotto perché si presti all'implementazione su sistemi diversi, limitando l'operazione di migrazione tra sistemi mobili all'installazione delle opportune API fornite dal programma cui è affidata la gestione dell'interfacciamento con tali dispositivi, e garantendo la pressoché totale compatibilità del codice del nucleo con tutti i browser per rendere possibile e quasi indolore l'eventuale rilascio di una versione web del modulo.

In conclusione del volume di tesi, si vuole riflettere sul fatto che quello delle applicazioni rivolte al mobile è un mercato in fortissima espansione e costante evoluzione e, considerando la presenza sempre più capillare delle nuove tecnologie *smart* in ogni aspetto della vita quotidiana, non c'è motivo di credere che tale campo rallenterà la sua crescita.

Si è scelto di sviluppare questo progetto di tesi seguendo una curiosità personale verso il mondo dei dispositivi mobili ed i nuovi approcci alla programmazione di applicazioni cross-platform, e si è lieti avere avuto la possibilità di approfondire lo studio di questo aspetto dell'informatica e di averlo fatto cimentandosi con la produzione di un

sistema rivolto all'insegnamento attraverso il divertimento, che si ritiene sia la forma più naturale e proficua di trasmissione del sapere.

Bibliografia

[ARC15] Archimedes' Lab, *What is the tangram*, sito: archimedeslab.org http://archimedes-lab.org/tangramagicus/pagetang1.html

[BEO11] Simone Beonati, *Canvas*, sito *www.html.it*, 2011 *http://www.html.it/pag/19303/canvas1/*

[BOL07] Giorgio Bolondi, *Giochi Matematici*, 2007, in: Corso di formazione per insegnanti offerto dall'istituto comprensivo di Villa Santina (UD) in collaborazione con il Centro Provinciale Servizi Scolastici *http://csaf.provincia.udine.it/data/servizi/formazione_doc/docs/upload/DIDATTICA%20DELLA%20MATEMATICA/GIOCHI%20MATEMATICI.pdf*

[BRA14a] Mario Bravetti, Slides *Il linguaggio HTML* di *Tecnologie Web/Internet* presso il Corso di Laurea in Ingegneria e Scienze Informatiche dell'Università di Bologna, sede di Cesena, AA 2013/2014

[BRA14b] Mario Bravetti, Slides *Cascading Style Sheet (CSS)* di *Tecnologie Web/Internet* presso il Corso di Laurea in Ingegneria e Scienze Informatiche dell'Università di Bologna, sede di Cesena, AA 2013/2014

[BRA14c] Mario Bravetti, Slides *Javascript* di *Tecnologie Web/Internet* presso il Corso di Laurea in Ingegneria e Scienze Informatiche dell'Università di Bologna, sede di Cesena, AA 2013/2014

[CAN03] Caterina Cangià, *Videogiochi e insegnamento /apprendimento: una sinergia inesplorata*, in: *Orientamenti Pedagogici* 50 (2003) 4, 737-755 *http://www.thesisternet.it/download/GRUPPO01~LUMSA_VIDEOGIOCHI/CONTRIBUTO_03.pdf*

[CAR96] Francesco Carlà, *Space Invaders: la vera storia dei videogames*, II Ed, p.13, Castelvecchi, Roma, 1996

[CAS90] Laura Catastini, *Il pensiero allo specchio*, Cap. III, La nuova Italia, 1990

http://www.mat.uniroma2.it/mep/Libro/capitoloIII.pdf

[CEC12] Alessio Ceccherelli, *Videogiochi e apprendimento tra medium e messaggio. Considerazioni sull'uso didattico dei videogiochi,* 2012 (Rivista Scuola IaD numero 6 - 2012, sezione Ricerca e Tecnologia)

[CHI13] Andrea Chiarelli, *Introduzione ad Apache Cordova*, sito *www.html.it*, 2013

http://www.html.it/pag/42121/introduzione-ad-apache-cordova/

[CHI14] Andrea Chiarelli, *Introduzione a JavaScript*, sito *www.html.it*, 2014

http://www.html.it/pag/45343/introduzione-a-javascript/

[CIT13] Sara Citterio, Eleonora Mossini, Isabella Lamoglie, *Problemi di Tangram*, 2013

https://www.google.it/url?sa=t&rct=j&q=&esrc=s&source=web&cd=13&ved=0ahUKEwiTkPmop7DJAhWFLhoKHaJgAJ4QFghVMAw&url=https%3A%2F%2Fmatelsup2-2013.wikispaces.com%2Ffile%2Fview%2FG-07-Problemi%2Bdi%2BTANGRAM.pptx&usg=AFQjCNFtdmGFYG74sXTiR3PbaVv3Z126ow

[DEM08] Erik D.Demaine, Susan Hohenberger, David Liben-Nowell, *Tetris is Hard, Even to Approximate*, 2008
http://arxiv.org/pdf/cs/0210020v1.pdf

[ECM15] ECMA, *Standard ECMA-262, EcmaScript® 2015 Language Specification, 6th edition (June 2015), 2015*
http://www.ecma-international.org/publications/standards/Ecma-262.htm

[KID15] Kidga, documentazione di Math Effect su GooglePlay, https://play.google.com/store/apps/details?id=com.kidga.mathrush&hl=it

[MAT04] *Tangram*, sito: Math.it, 2004

http://www.math.it/tangram/tangram.htm

[MIA04] Lidio Miato, *La teoria vygotskijana*, 2004, Iprase del Trentino,

http://try.iprase.tn.it/old/in05net/upload/pub/materiali/P4t4n217_Teoria_Vygotskij.pdf

[MIL10] Paolo Milazzo, Slides *5. Cascading Style Sheets (CSS)* di *Corso di Web Programming* presso il Corso di Laurea in Informatica Applicata dell'Università di Pisa, AA 2010/2011
http://www.di.unipi.it/~milazzo/teaching/AA1011-WebProg/slides/5-CSS.pdf

[MDN08] Mozilla Developer Network, *Ajax*, 2008
https://developer.mozilla.org/it/docs/AJAX

[MDN14] Mozilla Developer Network, *Una reintroduzione al JavaScript (Tutorial JS)*, 2015
https://developer.mozilla.org/it/docs/Web/JavaScript/Una_reintroduzione_al_JavaScript

[MDN15] Mozilla Developer Network, *CSS3*, 2015,
https://developer.mozilla.org/en-US/docs/Web/CSS/CSS3

[NES04] Romano Nesler, Mauro Fontanari , *Didattica assistita dalle nuove tecnologie*, 2004

[PAG11] Sandro Paganotti, *Canvas, approccio procedurale e librerie*, sito *www.html.it*, 2011
http://www.html.it/pag/19560/canvas-approccio-procedurale-e-librerie/

[SOS13] Commissione Europea, *Survey of schools: ICT in Edcation – Benchmarking access, use and attitudes to technology in Europe's schools*, 2013
https://ec.europa.eu/digital-agenda/sites/digital-agenda/files/KK-31-13-401-EN-N.pdf

[STA13] Stackoverflow, Session storage not working in IE, 2013
http://stackoverflow.com/questions/16212347/session-storage-not-working-in-ie

[W3C98] W3C, *Cascading Style Sheets, level 2 CSS2 Specification, 1998*
http://www.w3.org/TR/REC-CSS2/

[W3C08] W3C, *Cascading Style Sheets, level 1,W3C Recommendation 17 Dec 1996, revised 11 Apr 2008*, 2008
http://www.w3.org/TR/REC-CSS1/

[W3C11] W3C, *Selectors Level 4, W3C Working Draft 29 September 2011*, 2011
http://www.w3.org/TR/2011/WD-selectors4-20110929/

[W3C14a] W3C, *HTML5, A vucabulary and associated APIs for HTML and XHTML, W3C Reccomendation 28 October 2014*, paragrafo *1.4 History*, 2014
http://www.w3.org/TR/html5/introduction.html#history-0

[W3C14b] W3C, *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification W3C Recommendation 07 June 2011, edited in place 17 December 2014 to point to new work*, 2014
http://www.w3.org/TR/CSS21/

[W3C14c] W3C, *HTML5, A vucabulary and associated APIs for HTML and XHTML, W3C Reccomendation 28 October 2014*, 2014
http://www.w3.org/TR/html5/

[W3C15a] *Browser Statistics*, sito *www.w3schools.com*, 2015
http://www.w3schools.com/browsers/browsers_stats.asp

[W3C15b] W3C, *Cascading Style Sheets home page*, 2015
http://www.w3.org/Style/CSS/

[W3C15c] W3C, *CSS Current Status*, 2015
http://www.w3.org/standards/techs/css#w3c_all

[WIU99] Håkon Wium Lie, Bert Bos, *Cascading Style Sheets, designing for the Web*, II Ed, cap 20, Addison Wesley, 1999

[YIL01] Lay, Moule e Quinney citati in Yildirim, Yasar Aksoy e Ozden, *Coparrison of Hypermedia Learning and Traditional Instruction on Knowledge Acquisition and Return*, in: *The Journal of Educational Research*, vol 94, IV Ed, 2001