Documentazione

1 Introduzione 3

1.1 Informazioni sul progetto 3

1.2 Abstract 3

1.3 Scopo 3

2 Analisi 4

2.1 Analisi del dominio 4

2.2 Analisi e specifica dei requisiti 4

2.2.1 Spiegazione elementi tabella dei requisiti: 5

2.3 Use case 5

2.4 Pianificazione 5

2.5 Analisi dei mezzi 5

2.5.1 Software 6

2.5.2 Hardware 6

3 Progettazione 6

3.1 Design dell’architettura del sistema 6

3.2 Design dei dati e database 6

3.3 Design delle interfacce 6

3.4 Design procedurale 6

4 Implementazione 7

5 Test 7

5.1 Protocollo di test 7

5.2 Risultati test 8

5.3 Mancanze/limitazioni conosciute 8

6 Consuntivo 8

7 Conclusioni 8

7.1 Sviluppi futuri 8

7.2 Considerazioni personali 8

8 Glossario 8

9 Bibliografia 9

9.1 Bibliografia per articoli di riviste: 9

9.2 Bibliografia per libri 9

9.3 Sitografia 9

10 Allegati 9

# Introduzione

## Informazioni sul progetto

Allievi: Michael Dobeson, Luca Fumasoli

Classe: I3AC

Docente responsabile: Guido Montalbetti

Data inizio: 09.09.2022

Data fine: 23.12.2022

## Abstract

*Pac-Man is a well-known videogame which has stood the test of time and cemented itself has a classic.*

*In this game the player has to control Pac-Man in a maze that has a pill on every cell and the players goal is to eat all the pills. In the meantime, there are 4 ghosts following the player and if they catch you, you lose a life. If the player loses al his lives the game is over.*

*There are also super-pills in the maze which make Pac-Man invincible and capable of eating the ghosts.*

*In our variation of Pac-Man there is only one ghost and the maze is randomly generated each game.*

## Scopo

Lo scopo di questo progetto è di avere una variazione di Pac-Man con 1 fantasma e dei labirinti randomizzabili per continuare a rendere le partite diverse ed eliminare la familiarità che il giocatore potrebbe avere con la mappa forzandolo ad improvvisare ogni partita.

# Analisi

## Analisi del dominio

Questo capitolo dovrebbe descrivere il contesto in cui il prodotto verrà utilizzato, da questa analisi dovrebbero scaturire le risposte a quesiti quali ad esempio:

* Background/Situazione iniziale
* Quale è e come è organizzato il contesto in cui il prodotto dovrà funzionare?
* Come viene risolto attualmente il problema? Esiste già un prodotto simile?
* Chi sono gli utenti? Che bisogni hanno? Come e dove lavorano?
* Che competenze/conoscenze/cultura posseggono gli utenti in relazione con il problema?
* Esistono convenzioni/standard applicati nel dominio?
* Che conoscenze teoriche bisogna avere/acquisire per poter operare efficacemente nel dominio?
* …

## Analisi e specifica dei requisiti

Il progettista, dopo aver ricevuto il mandato, in collaborazione con il committente redige una lista di requisiti. Durante questi incontri, tramite interviste (da inserire nei diari), il progettista deve cercare di rispondere alle seguenti domande:

* Quali sono i bisogni del committente?
* Quali funzioni deve svolgere il prodotto?
* Come devono essere implementate?
* L’utente, come vorrebbe/dovrebbe interagire con il prodotto?
* Come verrà utilizzato il prodotto?
* Che tipo di interfaccia si immagina?
* Che prestazioni minime deve fornire il prodotto?
* Che grado di sicurezza deve avere il prodotto?
* …

In base alla lista dei requisiti e all’analisi degli stessi, il progettista redige una *specifica dei requisiti* in cui elenca e descrive in modo dettagliato quali sono le funzionalità che il prodotto fornirà. La specifica dovrebbe essere abbastanza dettagliata da poter essere utilizzata come base per lo sviluppo, ma non troppo; ad esempio non dovrebbe contenere dettagli di implementazione, o definizioni dettagliate dell’interfaccia grafica a meno che questi non siano considerati cruciali. Non si deve scordare che i requisiti non rappresentano delle attività bensì delle caratteristiche che il prodotto dovrà possedere.

Priorità 2/3 = opzionale

|  |  |
| --- | --- |
| **ID: REQ-01** | |
| **Nome** | Generazione Mappa |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Si deve usare l'algoritmo Manhattan Mapper per generare una mappa. |
| **Sotto requisiti** | |
| **001** | Conoscenza base di Unity. |

|  |  |
| --- | --- |
| **ID: REQ-02** | |
| **Nome** | Movimento Pac-Man |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Il giocatore può usare la tastiera per muoversi dentro la mappa. |
| **Sotto requisiti** | |
| **001** | Una matrice su cui può muoversi. |

|  |  |
| --- | --- |
| **ID: REQ-03** | |
| **Nome** | Piazzamento Pillole |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Vengono piazzate le pillole e Super-Pillole, che aumentano il punteggio, in modo casuale dentro la mappa. |
| **Sotto requisiti** | |
| **001** | Una matrice su cui piazzarle. |

|  |  |
| --- | --- |
| **ID: REQ-04** | |
| **Nome** | AI Fantasma |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Usando l'algoritmo Manhattan Mapper l'AI viene piazzato sulla mappa per inseguire il giocatore. |
| **Sotto requisiti** | |
| **001** | Una matrice su cui può muoversi. |
| **002** | La posizione del giocatore. |

|  |  |
| --- | --- |
| **ID: REQ-05** | |
| **Nome** | Super-Pillole |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Quando Pac-Man consuma una Super-Pillola il fantasma scappa (Manhattan inverso) e se viene raggiunto inizia a fuggire più velocemente per un tempo parametrizzabile per poi rinascere e ritornare ad inseguire Pac-Man. |
| **Sotto requisiti** | |
| **001** | Le pillole sono piazzate. |
| **002** | Il giocatore riesce a consumarle. |
| **003** | Un'AI che segue il giocatore. |

|  |  |
| --- | --- |
| **ID: REQ-06** | |
| **Nome** | Grafica |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Un'interfaccia con i bottoni per cominciare, uscire e ricominciare. Dettagli del gioco come le vite, il punteggio e l'Highscore che viene salvato in una variabile globale. |
| **Sotto requisiti** | |
| **001** | Le pillole sono piazzate. |
| **002** | Il giocatore riesce a consumarle. |
| **003** | Un'AI che segue il giocatore. |

|  |  |
| --- | --- |
| **ID: REQ-07** | |
| **Nome** | Vittoria/Perdita/Ricomincia |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | I diversi casi in cui una nuova mappa viene generata ed i personaggi vengono ripristinati:   * Quando tutte le pillole e Super-Pillole sono consumate (Vittoria). * Quando perdi tutte le tue vite per colpa del fantasma che ti colpisce (Perdita). * Quando metti in pausa il gioco e clicchi ricomincia. |
| **Sotto requisiti** | |
| **001** | Le pillole sono piazzate. |
| **002** | Il giocatore riesce a consumarle. |
| **003** | Un'AI che segue il giocatore. |
| **004** | Pulsanti per ricominciare la partita. |

|  |  |
| --- | --- |
| **ID: REQ-08** | |
| **Nome** | Variabili del Gioco |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Prima di iniziare la partita, si potrà modificare delle variabili di gioco (Vite, tempo Super-Pillole, grandezza delle mappe, velocità del fantasma, ecc..). |
| **Sotto requisiti** | |
| **001** | Il gioco finito e funzionante. |

|  |  |
| --- | --- |
| **ID: REQ-09** | |
| **Nome** | Difficoltà |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Si potrà scegliere il livello di difficoltà prima di iniziare la partita. |
| **Sotto requisiti** | |
| **001** | Il gioco finito e funzionante. |
| **002** | Si puo' cambiare le variabili del gioco facilmente. |

|  |  |
| --- | --- |
| **ID: REQ-10** | |
| **Nome** | Seconda Grafica |
| **Priorità** | 3 |
| **Versione** | 1.0 |
| **Note** | Aggiungere una seconda grafica 3D più “realistica”. |
| **Sotto requisiti** | |
| **001** | Il gioco finito e funzionante. |

|  |  |
| --- | --- |
| **ID: REQ-11** | |
| **Nome** | Modalità "Gambling" |
| **Priorità** | 3 |
| **Versione** | 1.0 |
| **Note** | Una modalità di gioco in cui dopo ogni livello nella partita si può scegliere se si vuole salvare il punteggio oppure continuare con doppio punteggio e con la difficoltà alzata. E dopo quel livello chiede se vuoi triplo punteggio eccetera. Però se perdi durante uno dei livelli, perdi il tuo punteggio per sempre. |
| **Sotto requisiti** | |
| **001** | Il gioco finito e funzionante. |
| **002** | Si può cambiare le variabili del gioco facilmente. |

### Spiegazione elementi tabella dei requisiti:

**ID**: identificativo univoco del requisito

**Nome**: breve descrizione del requisito

**Priorità**: indica l’importanza di un requisito nell’insieme del progetto, definita assieme al committente. Ad esempio, poter disporre di report con colonne di colori diversi ha priorità minore rispetto al fatto di avere un database con gli elementi al suo interno. Solitamente si definiscono al massimo di 2-3 livelli di priorità.

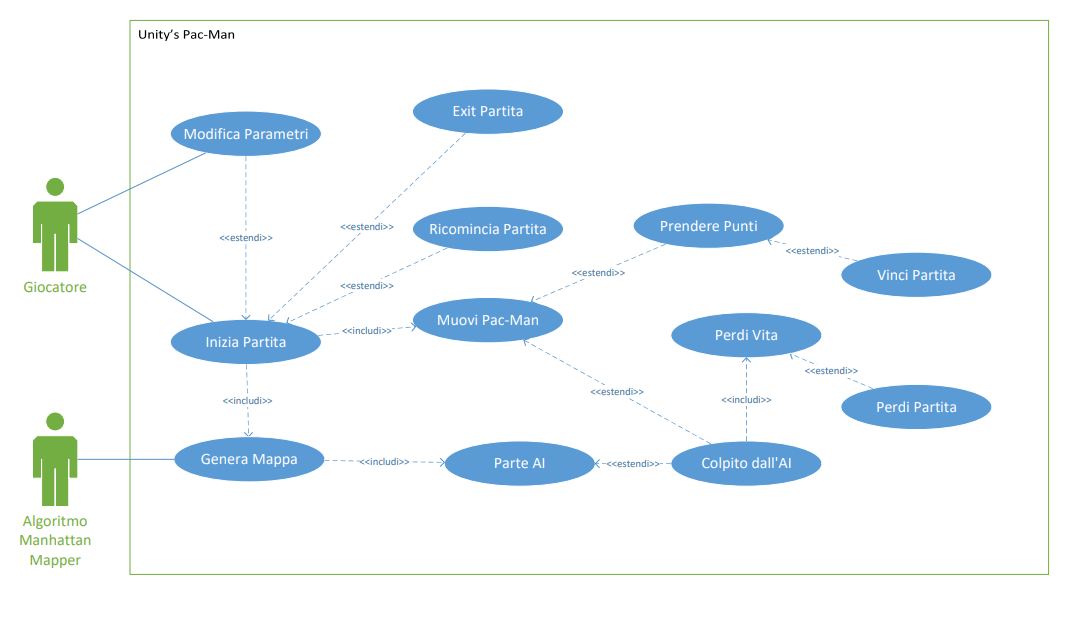
**Versione**: indica la versione del requisito. Ogni modifica del requisito avrà una versione aggiornata.

Sulla documentazione apparirà solamente l’ultima versione, mentre le vecchie dovranno essere inserite nei diari.

**Note**: eventuali osservazioni importanti o riferimenti ad altri requisiti.

**Sotto requisiti**: elementi che compongono il requisito.

## Use case



Ci sono 2 tipi di utente:

1. **Giocatore:** Il giocatore può modificare i parametri del gioco a suo piacimento prima di iniziare la partita. Poi quando avvia la partita muove il Pac-Man per il labirinto. Mangiando le pillole nel labirinto guadagna punti, se li mangia tutti vince la partita. Se viene colpitodal fantasma perde una vita e alla perdita di tutte le sue vite perde la partita.  
   Può anche mettere in pausa la partita e ricominciarla o uscire.
2. **Algoritmo Manhattan Mapper:** Si occupa di generare la mappa randomica e muovere l’AI, usando l’algoritmo trova la strada più corta per arrivare al giocatore e la segue, se invece il giocatore ha consumato una super-pillola trova la strada più efficace per scappare dal giocatore.

## Pianificazione

Prima di stabilire una pianificazione bisogna avere almeno una vaga idea del modello di sviluppo che si intende adottare. In questa sezione bisognerà inserire il modello concettuale di sviluppo che si seguirà durante il progetto. Gli elementi di riferimento per una buona pianificazione derivano da una scomposizione top-down della problematica del progetto.

La pianificazione può essere rappresentata mediante un diagramma di Gantt:

|  |
| --- |
| Figura 1: Esempio di diagramma di Gantt. |

Se si usano altri metodi di pianificazione (p.es. scrum), dovranno apparire in questo capitolo.

## Analisi dei mezzi

### Software

I software che abbiamo usato sono:

* Microsoft office professional plus 2019:
* Microsoft word: per fare la documentazione
* Microsoft visio: per fare lo schema del sito e dei database
* Microsoft Project: per fare il diagramma di Gantt
* Unity hub

### Hardware

Come hardware abbiamo usato i computer scolastici:

* Processore Intel Core i7-9700 CPU @ 3.00GHz
* RAM: 32GB
* Scheda video:

# Progettazione

## Design dell’architettura del sistema

Descrive:

* La struttura del programma/sistema lo schema di rete...
* Gli oggetti/moduli/componenti che lo compongono.
* I flussi di informazione in ingresso ed in uscita e le relative elaborazioni. Può utilizzare *diagrammi di flusso dei dati* (DFD).
* Eventuale sitemap

## Design dei dati e database

Descrizione delle strutture di dati utilizzate dal programma in base agli attributi e le relazioni degli oggetti in uso.

Schema E-R, schema logico e descrizione.

Se il diagramma E-R viene modificato, sulla doc dovrà apparire l’ultima versione, mentre le vecchie saranno sui diari.

## Design delle interfacce

Descrizione delle interfacce interne ed esterne del sistema e dell’interfaccia utente. La progettazione delle interfacce è basata sulle informazioni ricavate durante la fase di analisi e realizzata tramite mockups.

## Design procedurale

Descrive i concetti dettagliati dell’architettura/sviluppo utilizzando ad esempio:

* Diagrammi di flusso e Nassi.
* Tabelle.
* Classi e metodi.
* Tabelle di routing
* Diritti di accesso a condivisioni …

Questi documenti permetteranno di rappresentare i dettagli procedurali per la realizzazione del prodotto.

# Implementazione

In questo capitolo dovrà essere mostrato come è stato realizzato il lavoro. Questa parte può differenziarsi dalla progettazione in quanto il risultato ottenuto non per forza può essere come era stato progettato.

Sulla base di queste informazioni il lavoro svolto dovrà essere riproducibile.

In questa parte è richiesto l’inserimento di codice sorgente - Print Screen - di maschere solamente per quei passaggi particolarmente significativi e/o critici.

Inoltre, dovranno essere descritte eventuali varianti di soluzione o scelte di prodotti con motivazione delle scelte.

Non deve apparire nessuna forma di guida d’uso di librerie o di componenti utilizzati. Eventualmente questa va allegata.

Per eventuali dettagli si possono inserire riferimenti ai diari.

# Test

## Protocollo di test

Definire in modo accurato tutti i test che devono essere realizzati per garantire l’adempimento delle richieste formulate nei requisiti. I test fungono da garanzia di qualità del prodotto. Ogni test deve essere ripetibile alle stesse condizioni.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-01 | **Nome:** | Mappa Generata Casualmente |
| **Descrizione:** | Si proverà a fare generare la mappa senza interruzioni tra il percorso. | | |
| **Procedura:** | 1. Fare partire l'applicativo in Unity. 2. Avviare una partita. 3. Controllare che la mappa è generata correttamente. | | |
| **Risultati attesi:** | Dovrebbe essere possibile raggiungere tutti i punti della mappa casuale, senza punti irraggiungibile in cui le mura bloccano il passaggio. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-002  REQ-02 | **Nome:** | Movimento Pac-Man |
| **Descrizione:** | Si proverà a fare muovere Pac-Man usando dei tasti sulla tastiera. | | |
| **Procedura:** | 1. Fare partire L'applicativo in Unity. 2. Avviare una partita. 3. Usare i diversi tasti seguenti per fare muovere il Pac-Man, in diverse direzioni, di un blocco sulla mappa:    * Tasto "w" o freccia in su: sposta in su.    * Tasto "a" o freccia a sinistra: sposta a sinistra.    * Tasto "s" o freccia in giu': sposta in giu'.    * Tasto "d" o freccia a destra: sposta a destra. | | |
| **Risultati attesi:** | Il Pac-Man si muove in 4 direzioni usando i tasti specificati precedentemente. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-003  REQ-02 | **Nome:** |  |
| **Descrizione:** |  | | |
| **Procedura:** | 1. Fare partire L'applicativo in Unity. 2. Avviare una partita. 3. Usare i diversi tasti seguenti per fare muovere il Pac-Man, in diverse direzioni, di un blocco sulla mappa:    * Tasto "w" o freccia in su: sposta in su.    * Tasto "a" o freccia a sinistra: sposta a sinistra.    * Tasto "s" o freccia in giu': sposta in giu'.    * Tasto "d" o freccia a destra: sposta a destra. | | |
| **Risultati attesi:** |  | | |

## Risultati test

Tabella riassuntiva in cui si inseriscono i test riusciti e non del prodotto finale. Se un test non riesce e viene corretto l’errore, questo dovrà risultare nel documento finale come riuscito (la procedura della correzione apparirà nel diario), altrimenti dovrà essere descritto l’errore con eventuali ipotesi di correzione.

## Mancanze/limitazioni conosciute

Descrizione con motivazione di eventuali elementi mancanti o non completamente implementati, al di fuori dei test case. Non devono essere riportati gli errori e i problemi riscontrati e poi risolti durante il progetto.

# Consuntivo

Consuntivo del tempo di lavoro effettivo e considerazioni riguardo le differenze rispetto alla pianificazione (cap. 1.7) (ad esempio Gantt consuntivo).

# Conclusioni

Quali sono le implicazioni della mia soluzione? Che impatto avrà? Cambierà il mondo? È un successo importante? È solo un’aggiunta marginale o è semplicemente servita per scoprire che questo percorso è stato una perdita di tempo? I risultati ottenuti sono generali, facilmente generalizzabili o sono specifici di un caso particolare? ecc.

## Sviluppi futuri

Migliorie o estensioni che possono essere sviluppate sul prodotto.

## Considerazioni personali

Cosa ho imparato in questo progetto? ecc.

# Glossario

Inserite una semplice tabella con due colonne che spieghi i termini specifici del progetto (lista dei termini in ordine alfabetico A-Z)

Esempio:

|  |  |
| --- | --- |
| **Termine** | **Descrizione** |
| AJAX | **Asynchronous JavaScript And XML**: una tecnica che permette di eseguire richieste ed ottenere dati da una pagina web in modo asincrono. |
| CSS | **Cascading Style Sheets**: linguaggio che permette di definire il layout e la grafica di una pagina web. |

# Bibliografia

## Bibliografia per articoli di riviste:

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo dell’articolo (tra virgolette),
3. Titolo della rivista (in italico),
4. Anno e numero
5. Pagina iniziale dell’articolo.

## Bibliografia per libri

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo del libro (in italico),
3. ev. Numero di edizione,
4. Nome dell’editore,
5. Anno di pubblicazione,
6. ISBN.

## Sitografia

1. URL del sito (se troppo lungo solo dominio, evt completo nel diario),
2. Eventuale titolo della pagina (in italico),
3. Data di consultazione (GG-MM-AAAA).

**Esempio:**

* http://standards.ieee.org/guides/style/section7.html, *IEEE Standards Style Manual*, 07-06-2008.

# Allegati

Elenco degli allegati, esempio:

* Diari di lavoro
* Codici sorgente/documentazione macchine virtuali
* Istruzioni di installazione del prodotto (con credenziali di accesso) e/o di eventuali prodotti terzi
* Documentazione di prodotti di terzi
* Eventuali guide utente / Manuali di utilizzo
* Mandato e/o QdC
* Prodotto
* …