MULTIPLAYER SPACE SHOOTER

Nicholas A. Carroll
Laura Mazzuca
Federico Mustich

1.0 ABSTRACT

Sviluppo di un’applicazione per smartphone per controllare un videogioco spaceshooter multiplayer su PC.
Il gioco consisterà in un multigiocatore 2D con ambientazione sci-fi in cui i giocatori controlleranno delle astronavi e combatteranno tra di loro. Il primo a raggiungere un certo numero di uccisioni sarà decretato il vincitore. I giocatori controlleranno un’astronave, scelta tra varie tipologie di navi, ciascuna con punti di forza e debolezze diverse.

2.0 ANALISI DEI REQUISITI

2.1 Raccolta dei Requisiti

- ➔ L'applicazione consentirà a più giocatori di giocare contemporaneamente ad un videogioco su PC utilizzando Smartphone come controller;
- ➔ Il videogioco dovrà permettere di scegliere tra più tipi di navi spaziali le quali avranno abilità e statistiche differenti tra loro;
- ➔ La scelta della navicella dovrà avvenire a livello di controller, come anche la personalizzazione delle impostazioni di partita;
- ➔ L'obiettivo del gioco sarà eliminare i giocatori avversari (tutti contro tutti) e il primo a raggiungere un certo punteggio sarà il vincitore;
- ➔ Il controller dovrà essere in grado di muovere la posizione e l'orientamento della navicella e di azionare comandi speciali come "attacca", "usa power up", ecc.;
- ➔ Necessaria bassa latenza nella connessione tra Smartphone e PC;
- ➔ La connessione dovrà avvenire via Internet;
- ➔ Le abilità da implementare dovranno essere supervelocità, invulnerabilità e attacco ad area/cono. Ogni navicella ne avrà una sola, in base alle sue statistiche ;
- ➔ Le statistiche dovranno essere velocità, resistenza e forza;
- ➔ I power up dovranno essere medikit, incremento velocità, proiettile speciale e scudi;
- ➔ Il numero massimo di giocatori di una partita dovrà essere 4.

2.2 Analisi del dominio

2.2.1 Vocabolario

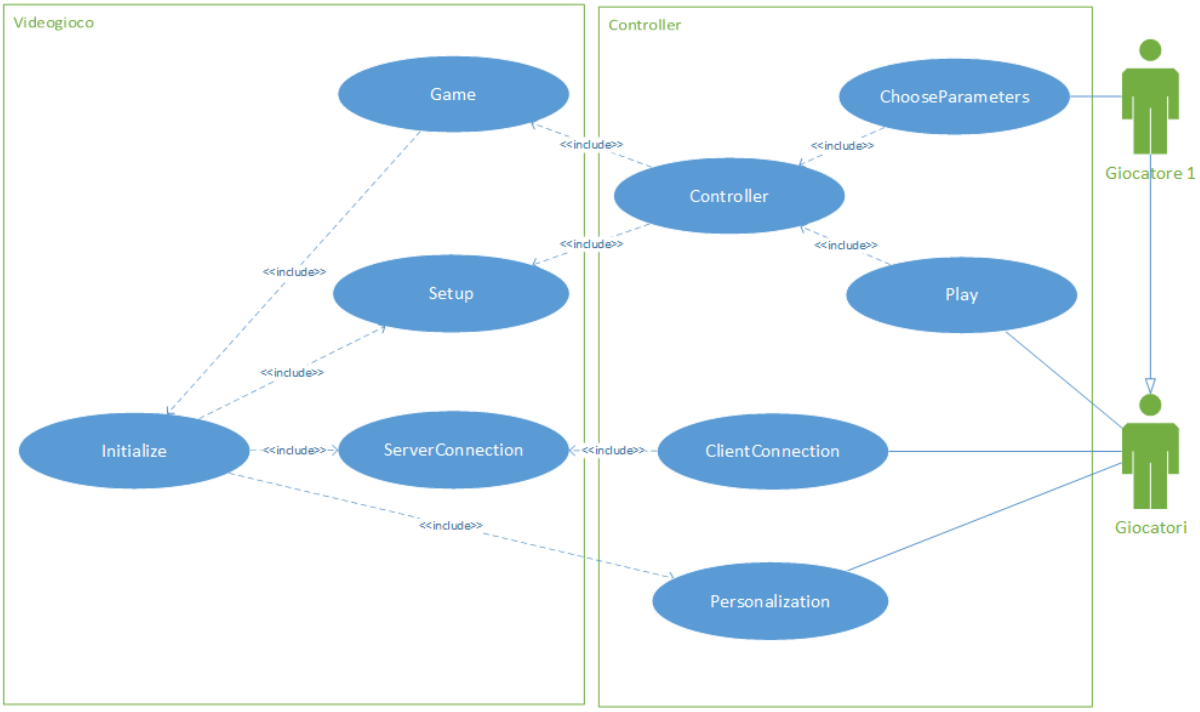
Voce	**Definizione**	**Sinonimi**
Nave Spaziale	Entità principale del videogioco che verrà controllata dal giocatore	Navicella, Ship
Controller	Oggetto che consente di interagire con il videogioco sul PC	Smartphone, Telefono

Giocatore	Persona che usa il controller per giocare al videogioco	Utente, Player
Giocatore_1	Primo giocatore a connettersi al videogioco. Sarà l'unico giocatore in grado di scegliere le impostazioni generali di gioco.	
Partita	Istanza del videogioco in corso di esecuzione	Game
Impostazioni	Insieme di parametri che il Giocatore_1 potrà scegliere per personalizzare la partita da giocare	Setup
Power Up	Oggetti che una volta acquisiti dalla navicella le danno dei bonus alle statistiche o nuove abilità	
Mappa di Gioco	Area in cui si svolge la partita	
Connessione	Collegamento tramite WiFi instaurato tra i Controller ed il Computer	
Statistiche	Attributi statici associati ad una particolare navicella	Stats
Comandi	Insieme di input provenienti dal Controller per gestire le azioni eseguite dalla propria navicella	Commands
Attacco	Azione con cui una Navetta spara per	Attack

	infliggere Danno ad un avversario	
Vita	Quantità di Danno che una Navicella può prendere prima di essere distrutta	HP, Salute
Danno	Quantità di Vita tolta ad un avversario quando viene colpito da un Attacco	Damage
Avversario	Navicella controllata da un altro giocatore	Enemy
De-spawn	Tempo che passa tra l'apparizione e la sparizione di un oggetto sulla mappa di gioco.	
Eliminazione	Una navicella si definisce eliminata nel momento in cui i suoi HP arrivano a zero. In questo caso dovrà attendere un certo tempo prima di poter ri-apparire sulla mappa e riprendere il gioco.	Morte, Death

2.3 Analisi dei Requisiti

2.3.1 Casi d'Uso



2.3.2 Scenari

Scenari videogioco

Titolo	Initialize
Descrizione	Permette di far connettere i giocatori, personalizzare le loro navicelle e decidere le impostazioni della partita
Attori	Giocatore_1, Giocatori
Relazioni	ServerConnection, Setup, Personalization Game
Precondizioni	Non deve già essere aperta una sessione del videogioco
Postcondizioni	Viene preclusa la possibilità ad altri giocatori che non siano quelli individuati durante l'apertura della sessione di gioco la possibilità di intromettersi nella partita correntemente in esecuzione o creare un'altra partita per conto proprio
Scenario principale	1. Avvio applicazione da PC e Smartphone, ad entrambe verrà mostrata la maschera di menu principale 2. ServerConnection 3. Il primo giocatore a connettersi sarà il Giocatore_1 4. Impostazioni 5. Personalizzazione 6. Avvio partita
Scenari alternativi	
Requisiti non funzionali	
Punti aperti	

Titolo	ServerConnection
Descrizione	I giocatori si connettono, il primo a connettersi sarà nominato Giocatore_1, il quale poi potrà decidere i parametri della partita.
Attori	Giocatori, Giocatore_1
Relazioni	Initialize, ClientConnection
Precondizioni	Non devono essere connessi altri giocatori
Postcondizioni	Deve essere scelto un Giocatore_1 per poter poi gestire le impostazioni
Scenario principale	1. Da PC si accederà alla sezione connessione, dove verrà mostrata la maschera per la Connessione dei Giocatori da PC 2. Si attenderanno le richieste di connessione dei giocatori. 3. Si verificheranno e si accetteranno le richieste di connessioni 4. Il primo a connettersi sarà automaticamente nominato Giocatore_1 5. Una volta che un giocatore è connesso gli si comunicherà che la procedura ha avuto successo ed eventualmente di essere il Giocatore_1
Scenari alternativi	Scenario A: -Uno o più giocatori non riescono a connettersi. Si dovrà ritentare la connessione Scenario B: -Uno o più giocatori si disconnettono dalla partita. Potranno ritentare la connessione a fine partita
Requisiti non funzionali	
Punti aperti	

Titolo	Setup
Descrizione	Il Giocatore_1 deciderà i parametri del gioco: 1. Numero di eliminazioni necessarie per vincere la partita 2. Frequenza dell'apparizione dei power up 3. Attiva/disattiva abilità speciali 4. Attiva/disattiva power-up 5. Volume musica 6. Volume effetti
Attori	Giocatore_1
Relazioni	Initialize, Controller
Precondizioni	I giocatori devono aver effettuato con successo la connessione
Postcondizioni	L'istanza del videogioco dovrà tenere conto di tutte le impostazioni definite dal giocatore designato
Scenario principale	1. Il Giocatore_1 potrà modificare le impostazioni di default della partita navigando nella maschera delle impostazioni tramite il controller, il quale sarà dotato di un joypad e tasti per selezionare l'opzione voluta per ciascun parametro
Scenari alternativi	
Requisiti non funzionali	
Punti aperti	

Titolo	Game
Descrizione	I giocatori iniziano la partita e una volta terminata potranno decidere se iniziarne un'altra o terminare l'applicazione
Attori	Giocatori, Giocatore_1
Relazioni	Initialize, Controller
Precondizioni	Sono state definite le impostazioni per la partita
Postcondizioni	I giocatori potranno iniziare una nuova partita o terminare l'applicazione.
Scenario principale	1. Inizia la partita 2. Al termine della partita verrà mostrata una maschera per la schermata punteggi 3. Il Giocatore_1 potrà decidere se iniziare un'altra partita: a. Se risponde di volerlo fare, per prima cosa il gioco chiederà se si vogliono cambiare i giocatori. i. se il Giocatore_1 risponde positivamente si tornerà alla maschera di Connessione ii. Se il Giocatore_1 non vuole cambiare i giocatori, gli verrà chiesto se vuole cambiare le impostazioni. 1. Se risponde di sì tornerà alla maschera di Setup. 2. Se risponde di no, tornerà alla maschera di gioco. b. Se i giocatori non vogliono giocare di nuovo l'applicazione tornerà alla maschera del menu principale dove sarà possibile terminare l'applicazione
Scenari alternativi	1. Durante la partita potranno interrompere il gioco e gli verrà mostrata una maschera di timeout dove potranno scegliere di tornare al menù principale oppure potranno far terminare la partita 2. Se faranno terminare la partita in quest'ultimo modo il gioco mostrerà la maschera di fine partita. 3. Il Giocatore_1 potrà decidere se iniziare un'altra partita: a. Se risponde di volerlo fare, per prima cosa il gioco chiederà se si vogliono cambiare i giocatori. i. se il Giocatore_1 risponde positivamente si tornerà alla maschera di Connessione ii. Se il Giocatore_1 non vuole cambiare i giocatori, gli verrà chiesto se vuole cambiare le impostazioni. 1. Se risponde di sì tornerà alla maschera di Setup. 2. Se risponde di no, tornerà alla maschera di gioco. b. Se i giocatori non vogliono giocare di nuovo l'applicazione tornerà alla maschera del menu principale dove sarà possibile terminare l'applicazione
Requisiti non funzionali	
Punti aperti	

Scenari Controller

Titolo	Personalization
Descrizione	Una volta che il Giocatore_1 avrà scelto le impostazioni, tutti i giocatori potranno personalizzare le loro navicelle: 1. Scelta di navicella tra i tipi: veloce, forte e resistente
Attori	Giocatori
Relazioni	Setup
Precondizioni	I giocatori devono aver effettuato con successo la connessione
Postcondizioni	Ogni controller deve avere la grafica personalizzata in base alla navicella e al colore scelto
Scenario principale	1. Il sistema mostrerà la maschera della Personalization delle navi su ciascun controller 2. Ciascun giocatore potrà scegliere la propria navicella e il colore che preferisce 3. I colori delle navicella verranno assegnati in base all'ordine di connessione dei giocatori (1-4) 4. Per procedere tutti i giocatori dovranno confermare le loro scelte
Scenari alternativi	
Requisiti non funzionali	
Punti aperti	

Titolo	ClientConnection
Descrizione	I giocatori si connettono, il primo a connettersi sarà nominato Giocatore_1, il quale poi potrà decidere i parametri della partita.
Attori	Giocatori
Relazioni	ServerConnection
Precondizioni	Non devono essere connessi altri giocatori
Postcondizioni	● Il giocatore dovrà essersi riuscito a connettere ● Il primo giocatore ad essersi connesso dovrà essere nominato
Scenario principale	1. Da smartphone si aprirà l'applicazione e verrà mostrata la maschera del menu principale. 2. Si accederà alla sezione connessione, dove verrà mostrata la maschera per la Connessione dei Giocatori 3. Si manderà la richiesta di connessione al videogioco al computer 4. Connessione Giocatore 5. Si attenderà la conferma che la procedura è andata a buon fine e l'eventuale notifica di essere il Giocatore_1
Scenari alternativi	Scenario A: ● Uno o più giocatori non riescono a connettersi. Si dovrà ritentare la connessione.
Requisiti non funzionali	● Disponibilità connessione WiFi
Punti aperti	

Titolo	Controller
Descrizione	Conterrà le freccette direzionali e vari tasti. Permetterà al giocatore 1 di scegliere le impostazioni e poi a tutti i giocatori di giocare una volta iniziata la partita
Attori	Giocatori, Giocatore_1
Relazioni	Game, Setup, Personalization
Precondizioni	I giocatori devono aver effettuato con successo la connessione e scelte le loro navicelle
Postcondizioni	
Scenario principale	1. Si accede alla sezione di Controller e verrà mostrata la maschera dei comandi su smartphone 2. Impostazioni 3. Gioca
Scenari alternativi	
Requisiti non funzionali	
Punti aperti	

Titolo	Play
Descrizione	I giocatori iniziano la partita e giocano
Attori	Giocatori, Giocatore_1
Relazioni	Controller
Precondizioni	Sono state definite le impostazioni per la partita, sono state decise le impostazioni e deve essere stata stabilita una connessione
Postcondizioni	I giocatori potranno iniziare una nuova partita o terminare l'applicazione.
Scenario principale	1. Si accede alla sezione gioco in automatico 2. Partita
Scenari alternativi	
Requisiti non funzionali	
Punti aperti	

Titolo	ChooseParameters
Descrizione	I giocatori iniziano la partita e giocano
Attori	Giocatore_1
Relazioni	Controller
Precondizioni	Il Giocatore_1 userà il controller per scegliere le impostazioni della partita
Postcondizioni	I giocatori potranno iniziare una nuova partita o terminare l'applicazione.
Scenario principale	1. Usando il Controller il giocatore_1 potrà selezionare varie opzioni in Setup
Scenari alternativi	
Requisiti non funzionali	
Punti aperti	

2.4 Analisi del Rischio

Nel nostro caso di risorse fisiche di cui curarci non ce ne sono, essendo il server implementato all'interno del videogioco e non essendoci risorse fisiche di altra natura.
Ciò che potrebbe esporre a rischi l'applicazione sono le risorse logiche, che rischiano l'intercettazione delle comunicazioni tra smartphone e computer per l'invio dei comandi, la connessione forzata di utenti non previsti e DoS.

2.4.1 Tabella di valutazione dei beni

Bene	Valore	Esposizione
Connessione Controller-Videogame	Alto. Se questa viene a mancare è impossibile portare avanti la Sessione di Gioco.	Alta. Perdita d'immagine e abbandono del gioco.

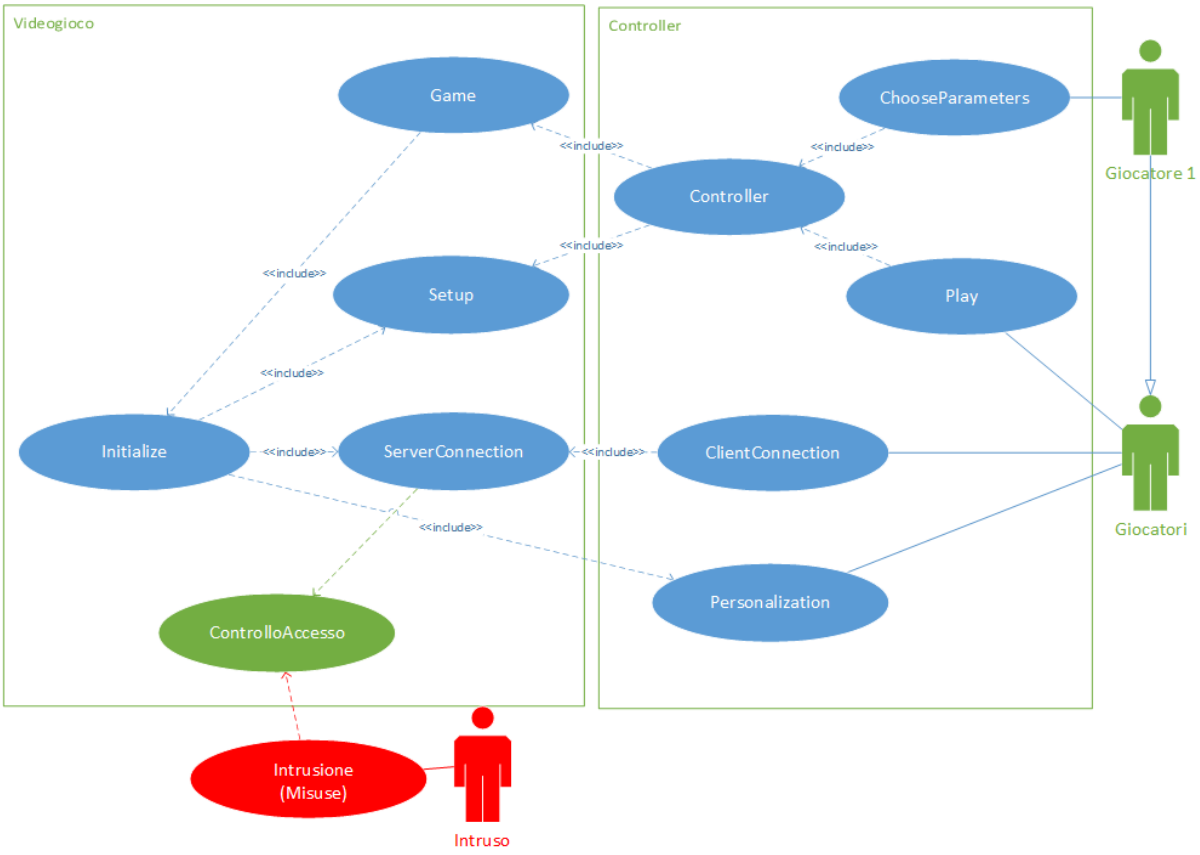
2.4.2 Tabella delle Minacce e dei Controlli

Minaccia	Probabilità	Controlli	Fattibilità
Verifica connessione giocatori	Bassa	Log delle comunicazioni	Basso costo

2.4.3 Analisi Tecnologica della Sicurezza

Tecnologia	Vulnerabilità
Architettura Client/Server	• DoS • Men in the Middle • Sniffing delle comunicazioni

2.4.4 Security Use Case & Misuse Case



2.4.5 Scenari Security Use Case e Misuse Case

Titolo	AccessControl	
Descrizione	Gli accessi al sistema devono essere controllati	
Misuse case	Verifica connessione giocatori	
Relazioni		
Precondizioni		
Postcondizioni	Il sistema blocca momentaneamente l'accesso all'utente e notifica il tentativo di accesso illecito	
Scenario principale	**Sistema**	**Attaccante**
		Un utente indesiderato tenta di connettersi alla partita
	Il sistema mostra sullo schermo del computer un codice di conferma da inserire nel controller. Il sistema poi controllerà se il codice è corretto e dopo un certo numero di tentativi bloccherà l'accesso.	
Scenario di attacco avvenuto con successo	**Sistema**	**Attaccante**
		L'utente tenta la connessione e inserisce il codice di conferma corretto
	Il sistema, dopo aver verificato il codice di conferma, permette al giocatore di unirsi alla sessione di gioco	
	I giocatori potranno accorgersi di un giocatore indesiderato nella loro sessione di gioco e potranno abbandonarla e crearne una nuova	

2.4.6 Requisiti di Protezione dei Dati

Dall'analisi del rischio si evince che non ci sono dati sensibili da proteggere ma per evitare l'intrusione di utenti esterni nella sessione di gioco dei giocatori il sistema dovrà avere una politica di controllo delle connessioni.

2.5 Descrizione delle interfacce grafiche

2.5.1 Interfaccia Videogioco

L'interfaccia del videogioco dovrà prevedere la schermata iniziale, la schermata di connessione, la schermata di impostazioni e quella di gioco. Dovrà essere possibile scegliere, al termine della partita, se ricominciare a giocare con gli stessi giocatori, passando comunque dall'interfaccia di impostazioni, se ricominciare "velocemente" una partita, ovvero con le impostazioni precedenti, se far connettere/disconnettere altri giocatori e quindi tornare alla schermata di connessione oppure se uscire dal gioco. Dovrà anche essere possibile chiamare da Controller, durante la partita, una schermata che consente di uscire eventualmente dalla partita al momento avviata, uscire definitivamente dal gioco oppure semplicemente essere usata come "timeout".

2.5.2 Interfaccia Controller

L'interfaccia dell'applicazione dovrà avere una schermata iniziale, una schermata di avvio connessione con il PC, la schermata di scelta della navicella ed infine la schermata Controller, contenente l'interfaccia del controller. L'interfaccia del controller dovrà avere un joystick per il movimento, un pulsante per sparare ed uno per usare l'abilità speciale e infine un pulsante "start" che consente di chiamare la schermata di "timeout" del gioco ed uno per tornare al menu connessione dell'applicazione.

3.0 ANALISI DEL PROBLEMA

3.1 Analisi delle Funzionalità

3.1.1 Tabella delle funzionalità

Videogioco

Funzionalità	Tipo	Complessità
Initialize	gestione delle fasi di preparazione	semplice
ServerConnection	creazione e gestione connessione	semplice
Setup	gestione dati	semplice
Game	elaborazione dati	complessa

Inizializzazione: Tabella Informazioni/flusso

Informazioni	Tipo	Livello riservatezza / privacy	Input/output	Vincoli
ID Giocatore 1	semplice	protezione media	input	max 1 carattere (numero tra 1 e 4)
ID altri giocatori	semplice	protezione media	input	max 1 carattere (numero tra 1 e 4)
Parametro:frequenza powerup	semplice	protezione media	input	un valore tra 5 e 60 secondi
Parametro: numero di uccisioni per vincere	semplice	protezione media	input	un valore tra 1 e 100
Parametro: esistenza dei power up	semplice	protezione media	input	sì o no

Parametro: volume musica	semplice	protezione bassa	input	un valore tra 1 e 100
Parametro: volume effetti	semplice	protezione bassa	input	un valore tra 1 e 100

ConnectionServer: Tabella Informazioni/flusso

Informazioni	**Tipo**	**Livello riservatezza / privacy**	**Input/output**	**Vincoli**
Indirizzo IP computer	semplice	protezione media	output	dev'essere un indirizzo ip valido
Indirizzo IP giocatori	semplice	protezione media	input	dev'essere un indirizzo ip valido
IP giocatore 1	semplice	protezione media	input	dev'essere un indirizzo ip valido
Comandi controller	composto	protezione media	input/output	

Impostazioni: Tabella Informazioni/flusso

Informazioni	**Tipo**	**Livello riservatezza / privacy**	**Input/output**	**Vincoli**
IP giocatore 1	semplice	protezione media	input/output	max 1 carattere (numero tra 1 e 4)
Parametro:frequenza powerup	semplice	protezione media	input/output	un valore tra 5 e 60 secondi
Parametro: numero di uccisioni per vincere	semplice	protezione media	input/output	un valore tra 1 e 10
Parametro: esistenza dei power up	semplice	protezione media	input/output	sì o no
Parametro: volume musica	semplice	protezione bassa	input/output	un valore tra 1 e 100
Parametro: volume effetti	semplice	protezione bassa	input/output	un valore tra 1 e 100

Partita: Tabella Informazioni/flusso

Informazioni	**Tipo**	**Livello riservatezza / privacy**	**Input/output**	**Vincoli**
Comandi controller	composto	protezione media	input	

Controller

Funzionalità	Tipo	Complessità
Controller	input dati	semplice
ClientConnection	invio dati	semplice
Personalization	input dati	semplice

Personalizationi: Tabella Informazioni/flusso

Informazioni	Tipo	Livello riservatezza / privacy	Input/output	Vincoli
ID giocatore	semplice	protezione media	input/output	max 1 carattere (numero tra 1 e 4)
ID chosenShip	semplice	protezione media	input/output	max 1 carattere (numero tra 1 e 4)

ClientConnection: Tabella Informazioni/flusso

Informazioni	Tipo	Livello riservatezza / privacy	Input/output	Vincoli
Indirizzo IP giocatore	semplice	protezione alta	output	dev'essere un indirizzo ip valido
Indirizzo IP computer	semplice	protezione alta	input	dev'essere un indirizzo ip valido
Comandi controller	composto	protezione media	output	

Gioca: Tabella Informazioni/flusso

Informazioni	**Tipo**	**Livello riservatezza / privacy**	**Input/output**	**Vincoli**
Comandi controller	Composto	protezione media	input/output	

3.2 Analisi dei vincoli

tabella dei vincoli

Requisito	**Categorie**	**Tipo di impatto**	**Funzionalità coinvolte**
Tempi di risposta Veloci	Performance	Necessari per una buona esperienza di gioco	Partita, Gioca

3.3 Analisi delle interazioni

3.2.1 Tabelle Delle maschere

Tabella delle Maschere - Videogioco

Maschera	Informazioni	Funzionalità
MainMenu	Schermata iniziale e avvio o chiusura applicazione	Setup
ConnectionInterface	Schermata di connessione che visualizza quanti e quali controller sono connessi	Setup
SetupInterface	Schermata in cui il giocatore proprietario del controller 1 può modificare le impostazioni ed ogni giocatore può scegliersi la propria navicella e personalizzarla	Setup
Game	Schermata di gioco	Partita
PauseMenu	Schermata di interruzione di gioco con possibilità di tornare alla schermata di impostazioni, riavviare la partita o uscire dal gioco	Partita
ScoreScreen	Schermata che viene visualizzata alla fine della partita e che riassume i punteggi dei giocatori e la classifica	Partita

Tabella delle Maschere - Controller

Maschera	Informazioni	Funzionalità
MainMenu	Schermata iniziale e avvio o chiusura applicazione	Setup
ConnectionInterface	Schermata di connessione per consentire e assicurare al controller la connessione con il videogioco	Setup
ControllerInterface	Schermata che consente, attraverso i pulsanti, di interagire con le schermate del videogioco	Setup, Partita
Personalization	Schermata che consente di scegliere la navicella con cui giocare	Setup

3.4 Analisi dei Ruoli e Responsabilità

Tabella Ruoli

Ruolo	Responsabilità	Maschere	Riservatezza	Numerosità
Giocatore_1	Deve scegliere le impostazioni della partita, per il resto è un normale giocatore	SetupInterface, PauseMenu	Non è richiesta una riservatezza particolare	1
Giocatore	Deve scegliere la propria nave e il proprio colore e poi giocare	Videogioco: MainMenu, ConnectionInterface, Game, ScoreScreen Controller: ConnectionInterface, MainMenu, InterfacciaController, InterfacciaPersonalizzazione	Non è richiesta una riservatezza particolare	1-4

Giocatore_1: Tabella Ruolo-Informazioni

Informazione	Tipo Accesso
Parametro: frequenza powerup	Lettura/Scrittura
Parametro: numero di uccisioni per vincere	Lettura/Scrittura
Parametro: esistenza dei power up	Lettura/Scrittura
Parametro: esistenza abilità speciali	Lettura/Scrittura
Parametro: volume musica	Lettura/Scrittura
Parametro: volume effetti	Lettura/Scrittura
ID nave	Lettura/Scrittura
Controlli	Lettura/Scrittura

Giocatore: Tabella Ruolo-Informazioni

Informazione	**Tipo Accesso**
ID nave	Lettura/Scrittura
Controlli	Lettura/Scrittura

3.5 Scomposizione del Problema

Tabella Scomposizione Funzionalità

Funzionalità	Scomposizione
Partita	GameManager, ShipManager, PowerupManager, ScoreManager, IOManager

Partita: Tabella Sotto-funzionalità

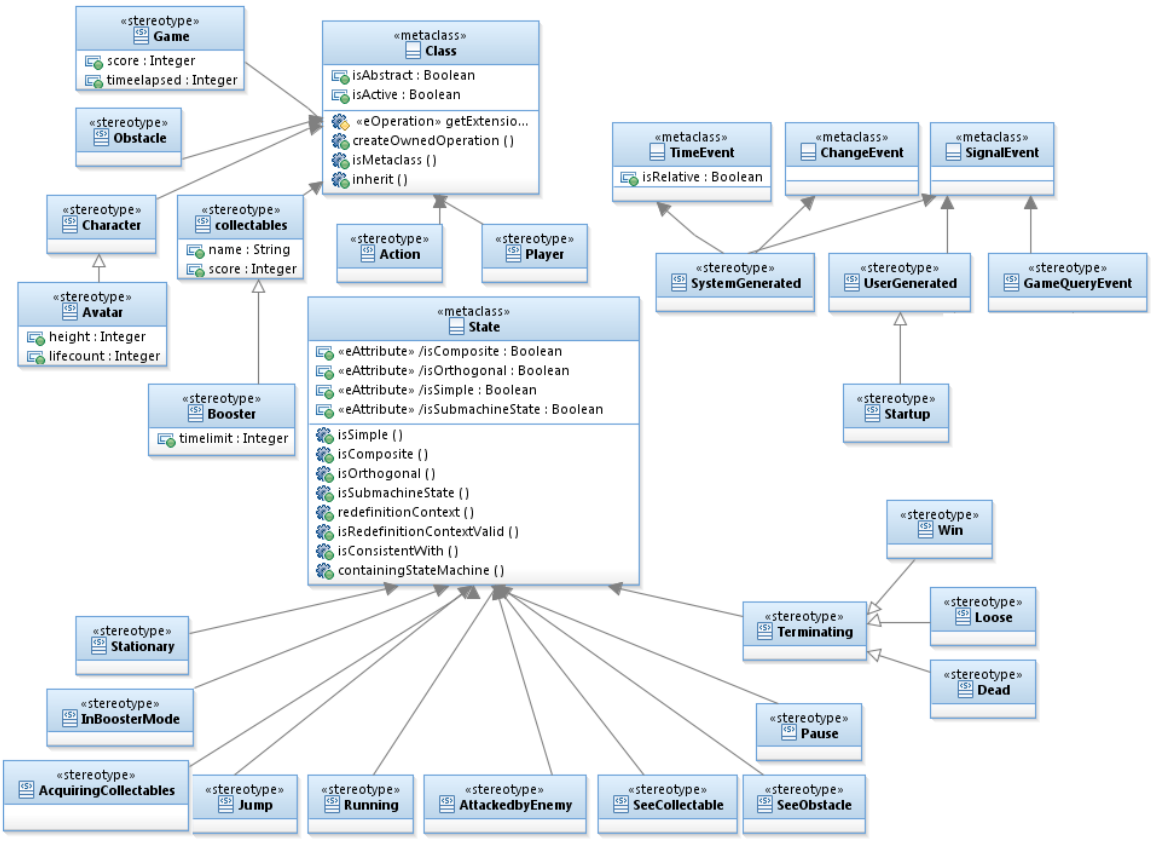
Sotto-funzionalità	Sotto-funzionalità	Legame	Informazioni
GameManager	ShipManager	Il GameManager deve passare al ShipManager le informazioni necessarie per farle comparire correttamente sulla mappa di gioco e legarle ai rispettivi giocatori in remoto	esistenza abilità speciali, ID navicelle, ID colori
GameManager	PowerupManager	Il Setup deve passare al PowerupManager le informazioni necessarie per farli comparire (o meno) sulla mappa di gioco nei modi specificati dal Gocatore_1	durata power up prima del de-spawn, esistenza dei power up
GameManager	ScoreManager	Il Setup deve passare al ScoreManager le informazioni necessarie per tenere traccia delle statistiche di gioco e per poter gestire le condizioni di vittoria	numero di uccisioni per vincere
IOManager	ShipManager	Il IOManager deve prendere l'input dei comandi dai Controller e passarli al ShipManager per riprodurre azioni e movimento	direzione orizzontale, direzione verticale, tasto spara, tasto usa abilità

4.0 Architettura Logica Videogioco

Per analizzare efficacemente il dominio e l'architettura logica del videogioco che abbiamo intenzione di creare ci rifacciamo, come indicazioni per la stesura del modello del dominio e dell'architettura logica, al profilo UML proposto nell'articolo *An Automated Model Based Testing Approach for Platform Games* presentato in occasione della conferenza *18th International Conference on Model Driven Engineering Languages and Systems · MODELS 2015, Canada*[1], di cui riportiamo qui sotto la struttura.

Unico cambiamento fatto agli stereotipi è stato quello di ridefinire alcuni stati in modo da renderli più attinenti alla nostra situazione:

- *Running -> Moving*;
- *Jumping -> Shooting.*

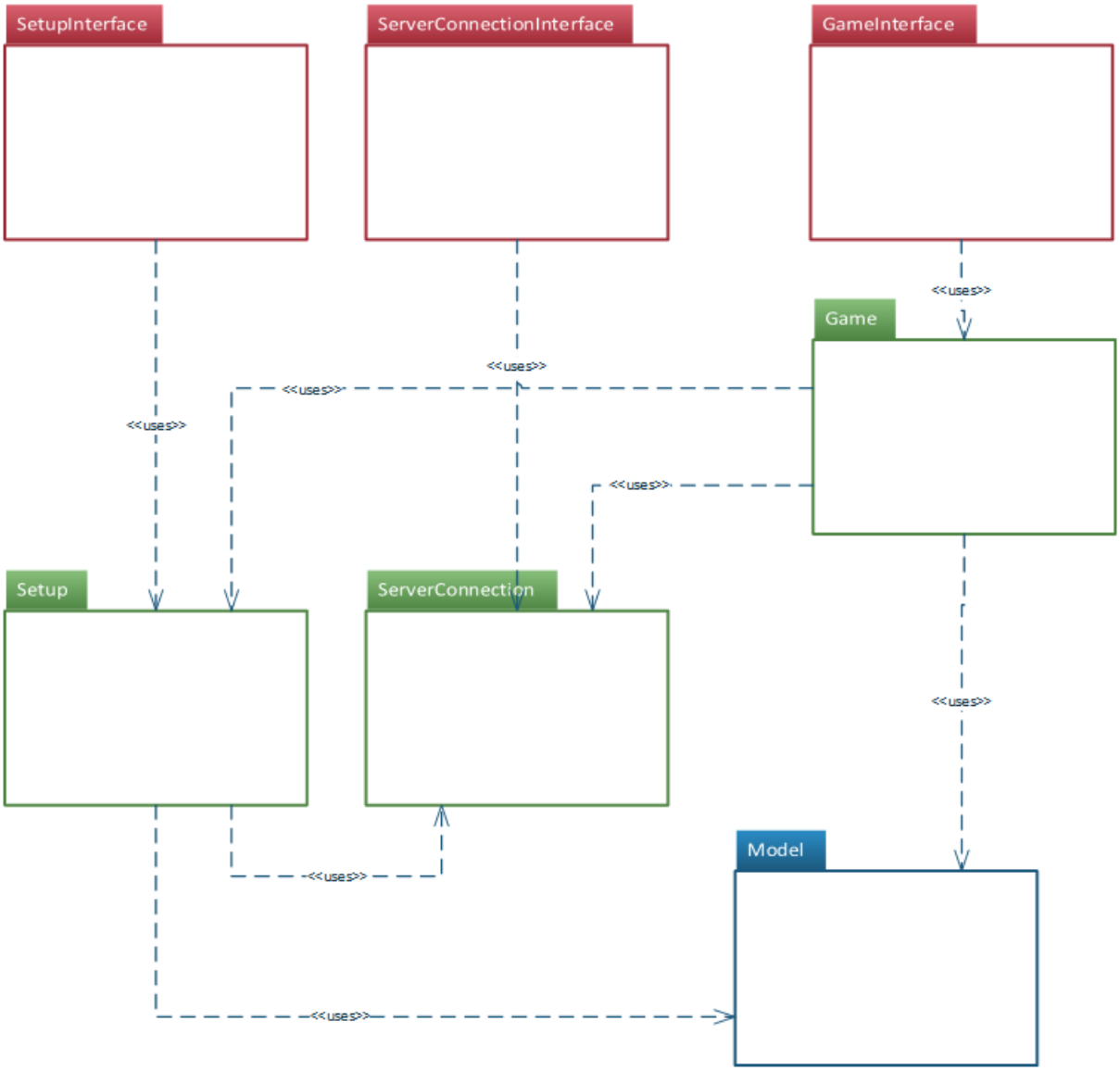


[1] Iftikhar, Sidra & Iqbal, Muhammad Zohaib & Khan, Muhammad Uzair & Mahmood, Wardah. (2015). An Automated Model Based Testing Approach for Platform Games. 10.1109/MODELS.2015.733228274.

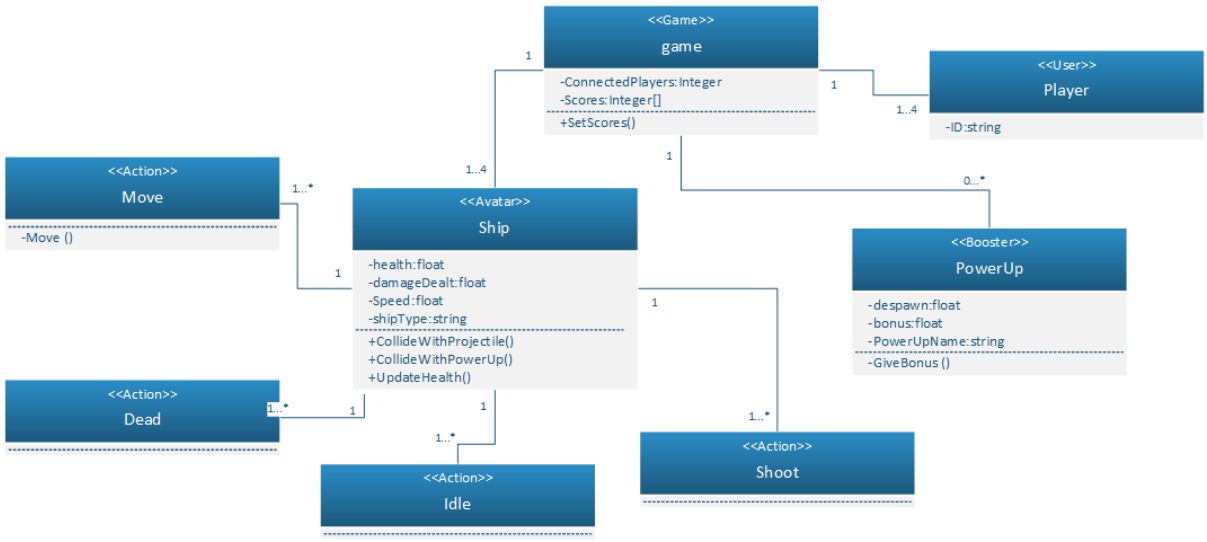
4.1 Architettura Logica Videogioco: Struttura

4.1.1 Diagramma dei Package

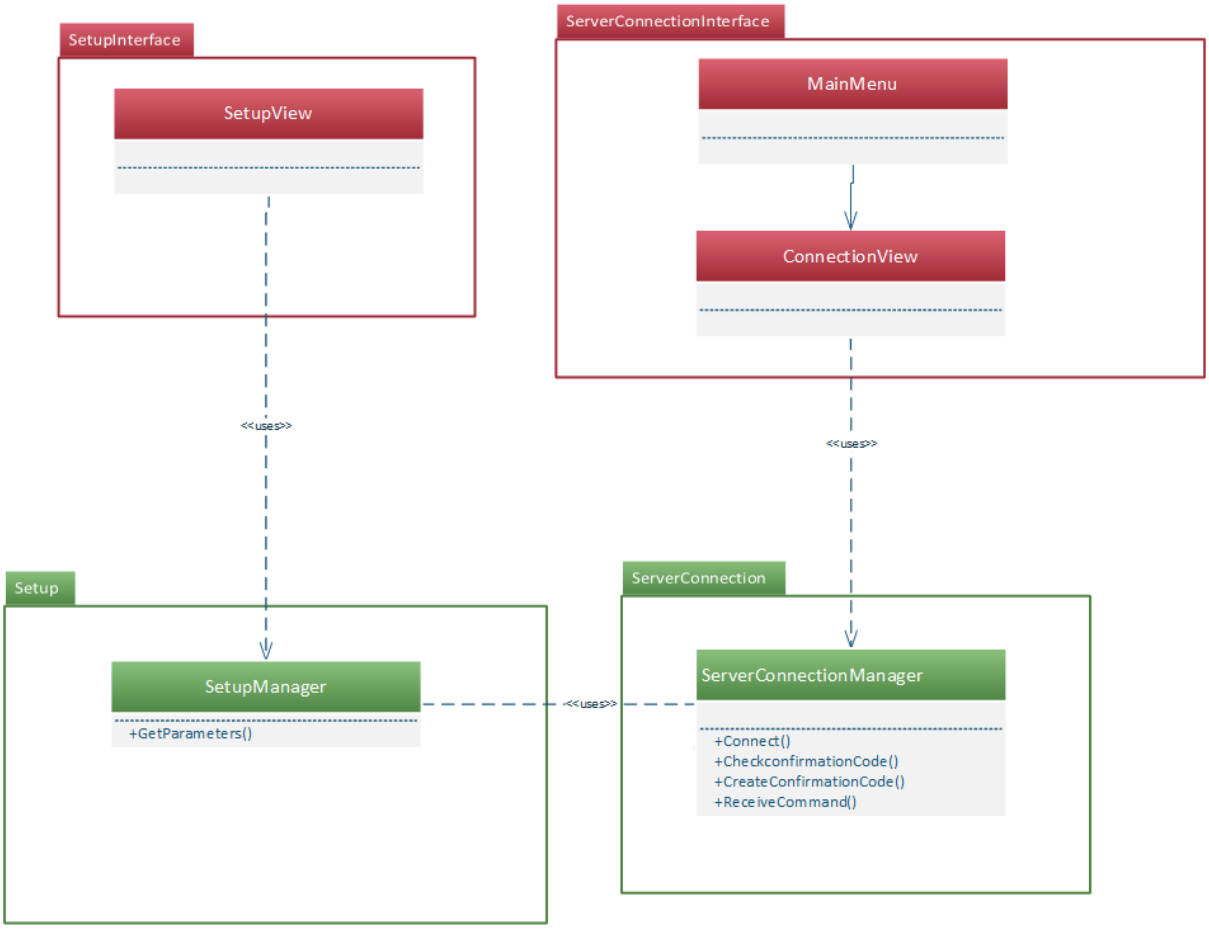
Essendo Initialize una semplice raccolta di informazioni da diversi package per conto della Partita, è stato deciso di non implementare l'inizializzazione come package, ma come classe GameManager all'interno di partita.



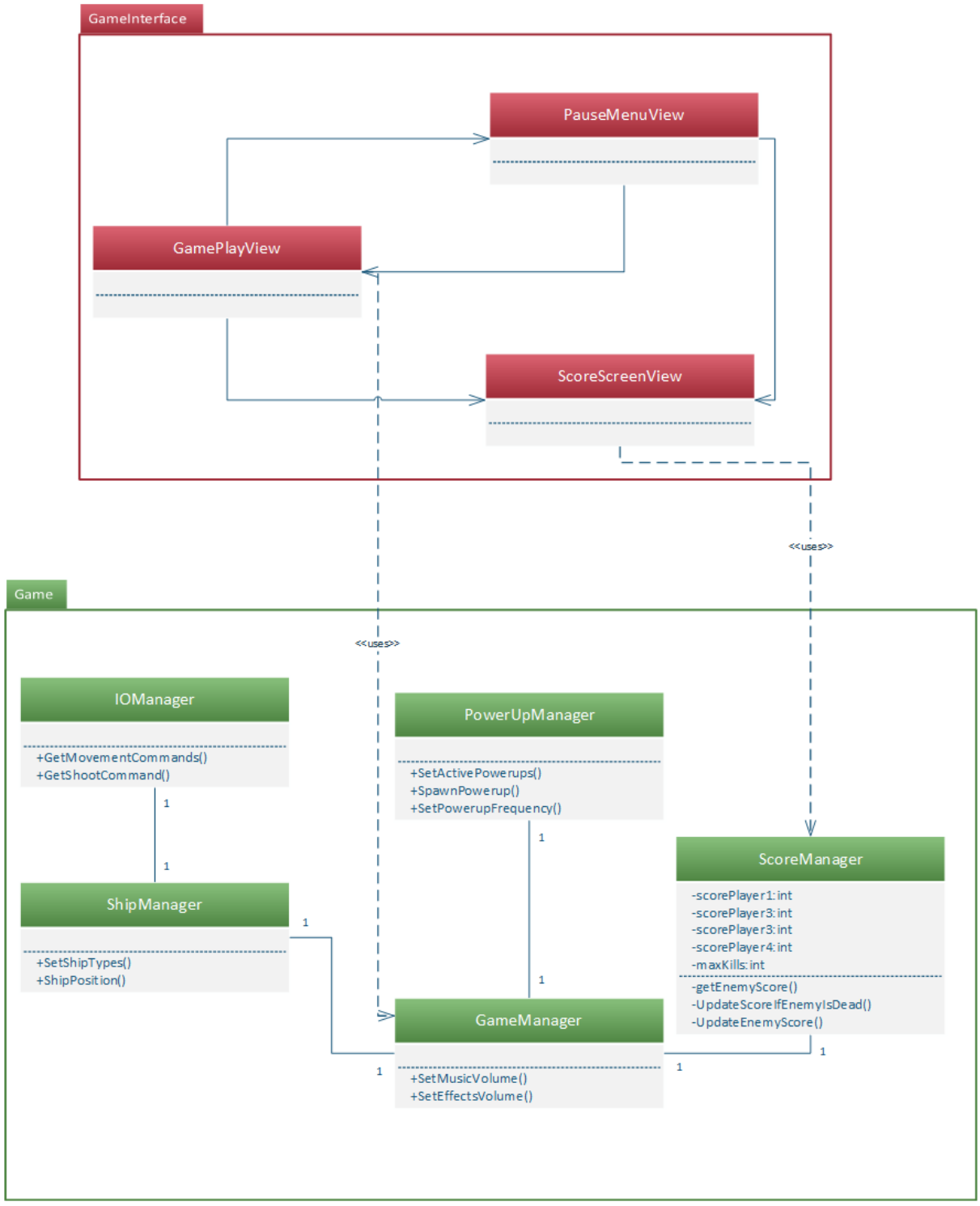
4.1.2 Diagramma delle classi: Dominio



4.1.3 Diagramma delle classi: ConnectionServer, Setup e ConnectionInterface, SetupInterface

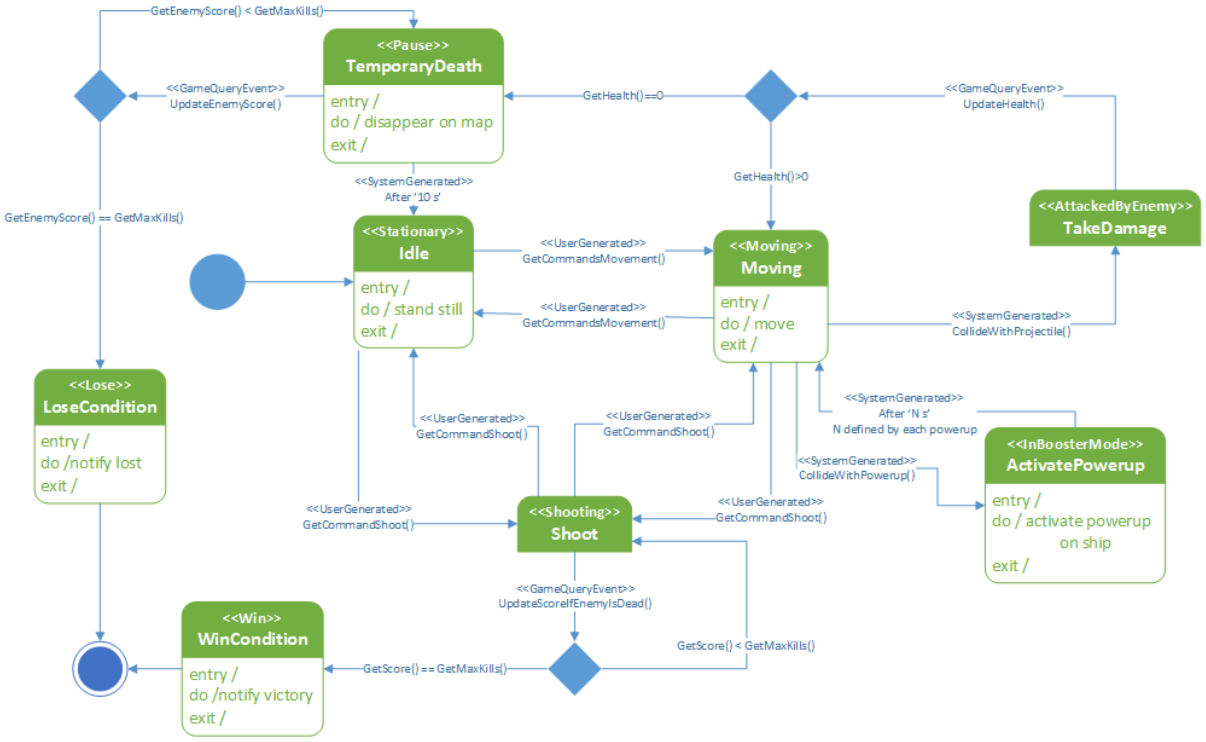


4.1.4 Diagramma delle classi: GameInterface e Game



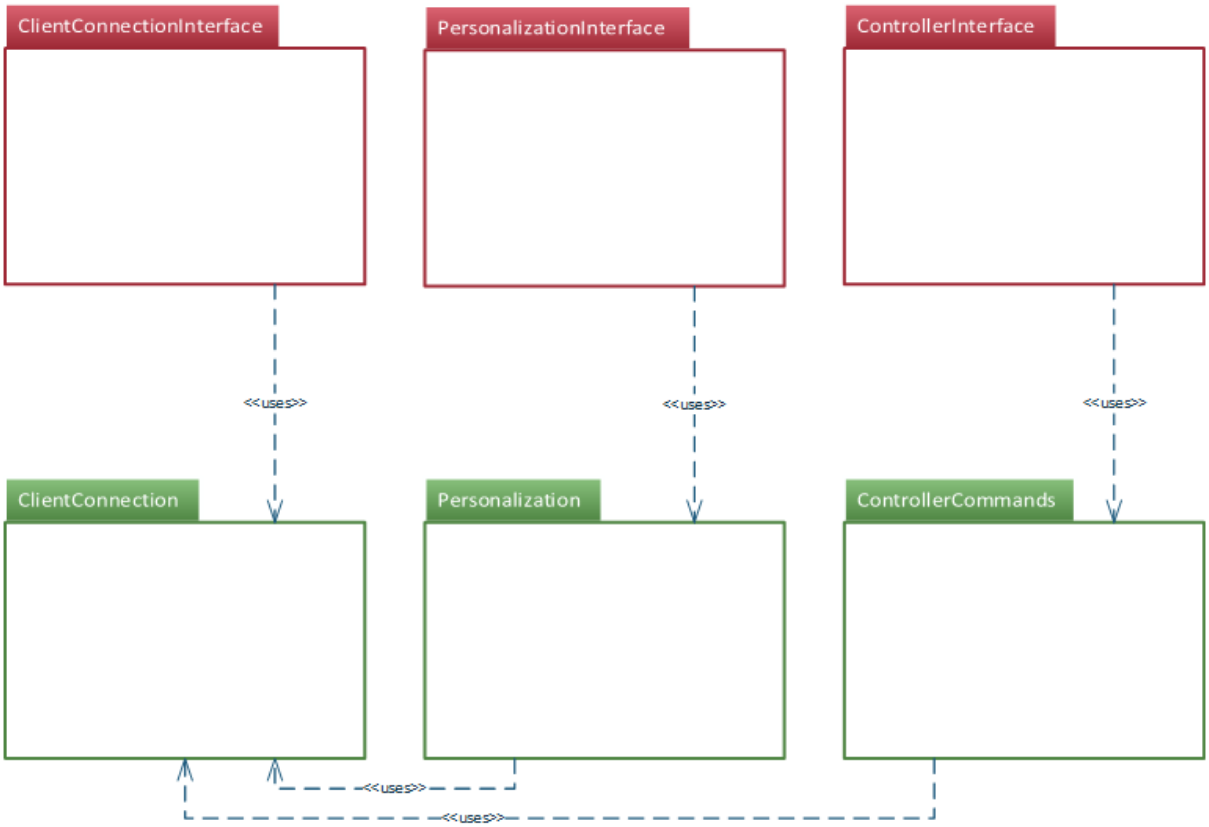
4.2 Architettura Logica Videogioco: Comportamento

Diagramma di stato: Ship

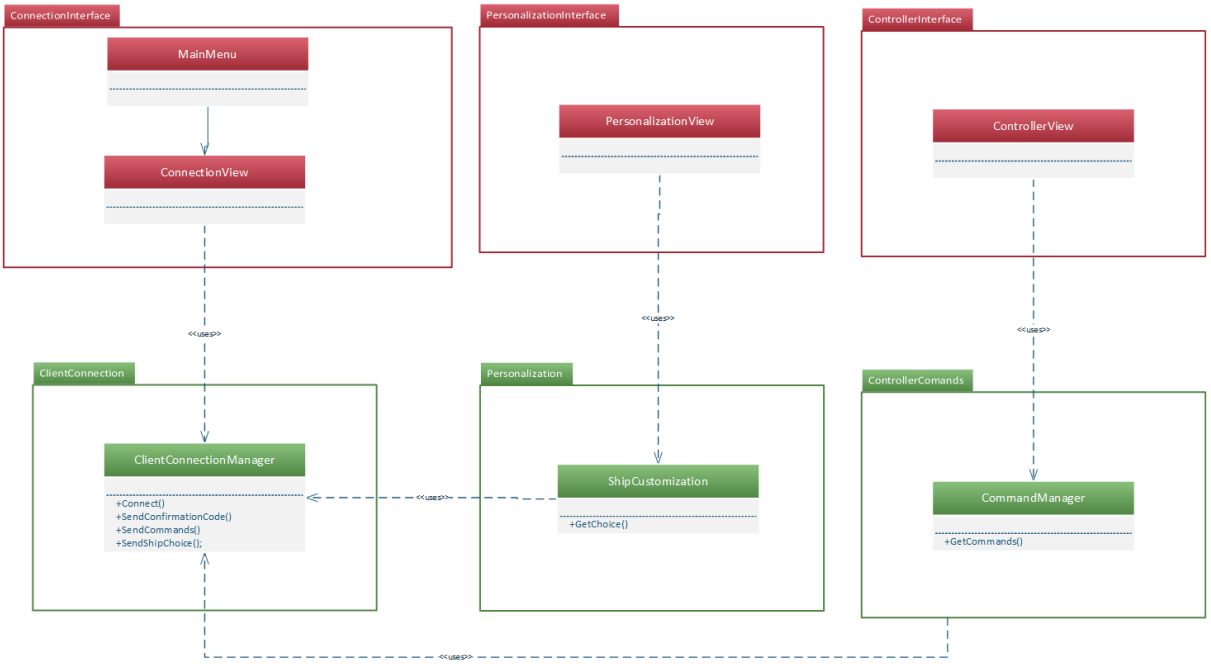


4.3 Architettura Logica Controller: struttura

4.3.1 Diagramma dei Package



4.3.2 Diagramma delle classi



4.4 Architettura logica Controller: Comportamento

Il controller non presenta particolari situazioni da dettagliare con diagrammi di stato o di attività.

4.5 Architettura Logica: Interazione

Si è deciso di rappresentare i diagrammi di interazione senza distinzione tra Videogioco e Controller.

4.5.1 Diagramma di sequenza:

Diagramma di sequenza: Connessione eseguita con successo

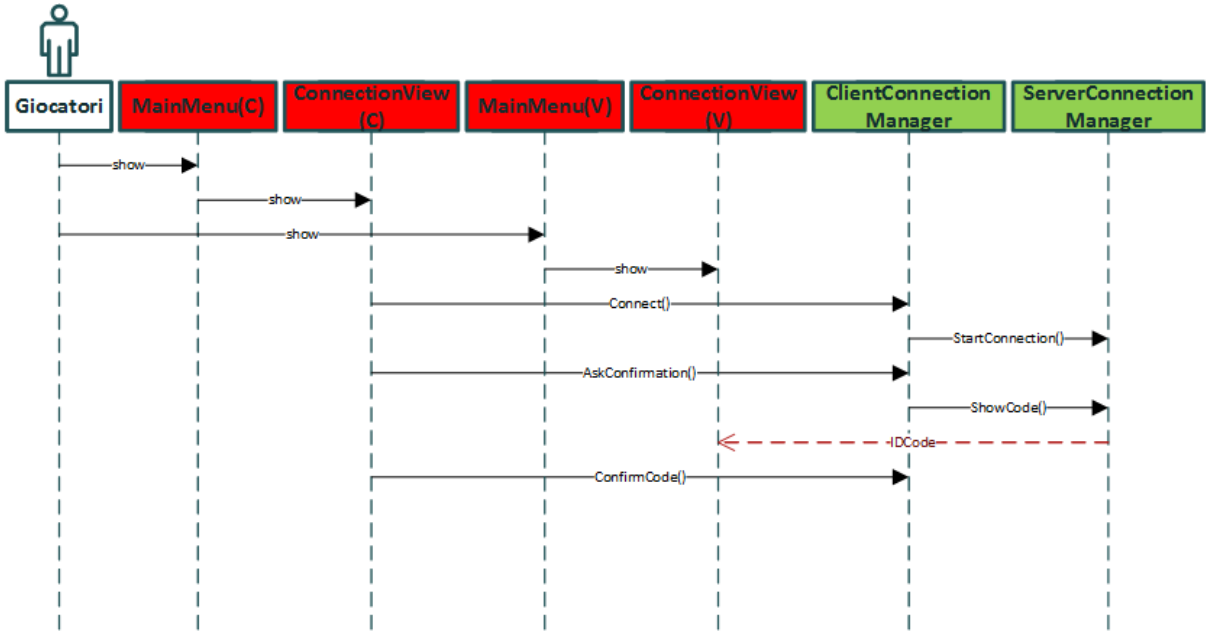


Diagramma di sequenza: Scelta navicelle eseguita con successo

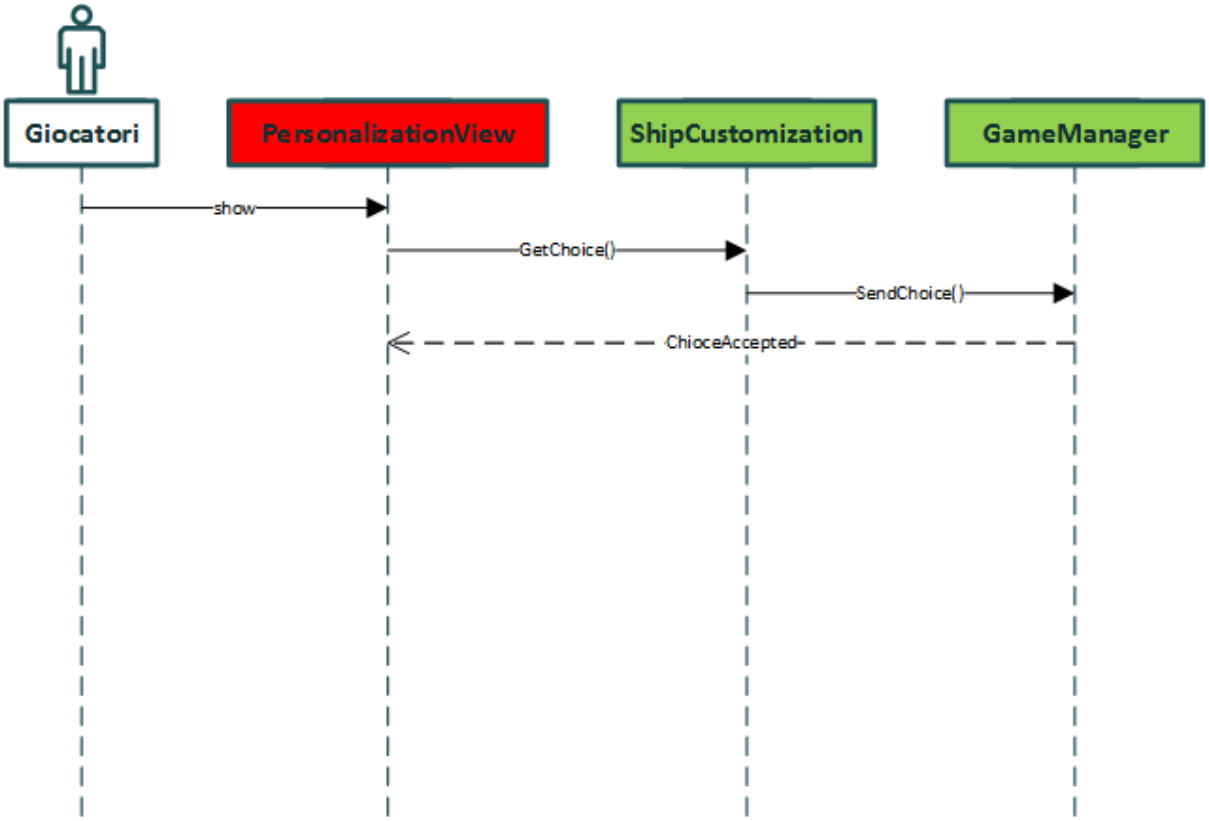
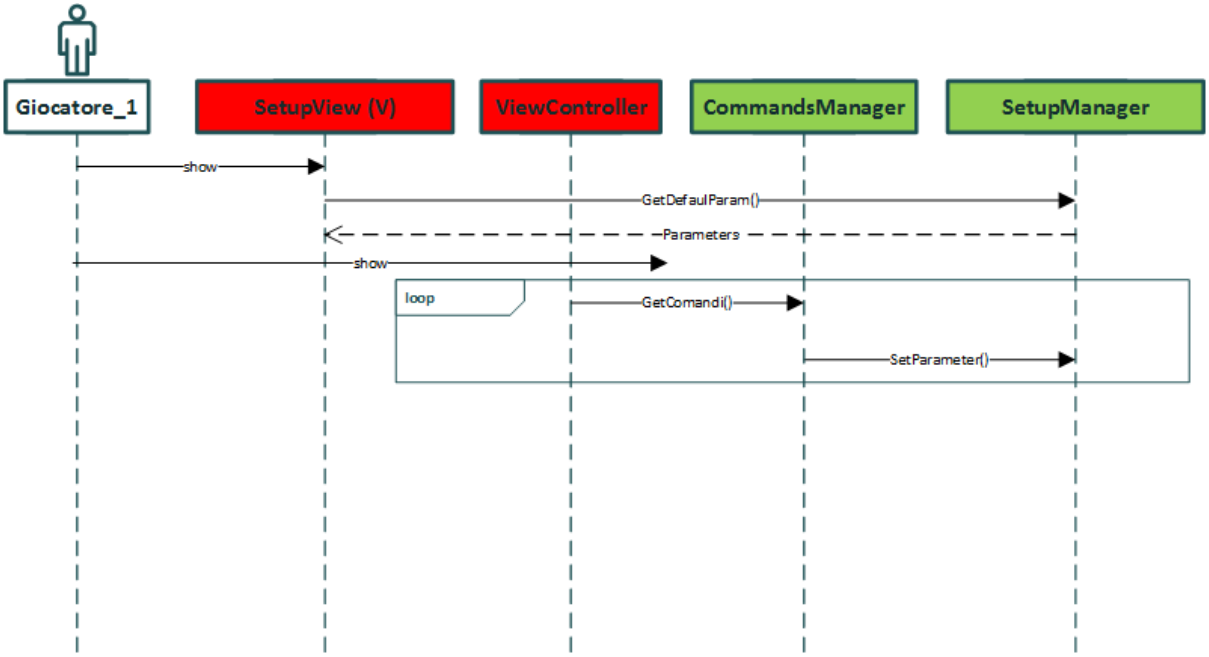


Diagramma di sequenza: Scelta impostazioni eseguita con successo



4.6 Piano di Lavoro

Il progetto e lo sviluppo del sistema sono assegnati a diverse persone come indicato nella tabella sottostante

Videogioco

Package	**Progetto**	**Sviluppo**
Dominio	Nicholas Carroll + Laura Mazzuca	Nicholas Carroll + Laura Mazzuca
Setup	Nicholas Carroll + Laura Mazzuca	Nicholas Carroll + Laura Mazzuca
Connection	Nicholas Carroll	Federico Mustich
Game	Nicholas Carroll + Laura Mazzuca	Nicholas Carroll + Laura Mazzuca
SetupInterface	Nicholas Carroll + Laura Mazzuca	Nicholas Carroll + Laura Mazzuca
ConnectionInterface	Nicholas Carroll + Laura Mazzuca	Nicholas Carroll + Laura Mazzuca
GameInterface	Nicholas Carroll + Laura Mazzuca	Nicholas Carroll + Laura Mazzuca

Controller

Package	**Progetto**	**Sviluppo**
Personalization	Federico Mustich	Federico Mustich
ComandiController	Federico Mustich	Federico Mustich
Connection	Federico Mustich	Federico Mustich
PersonalizationInterface	Federico Mustich	Nicholas Carroll + Laura Mazzuca
ControllerInterface	Federico Mustich	Nicholas Carroll + Laura Mazzuca
ConnectionInterface	Federico Mustich	Nicholas Carroll + Laura Mazzuca

4.7 Piano di Collaudo

Si sono individuati vari punti critici per l'esecuzione del programma:

Connessione: Al fine di assicurarsi che tutti e soli i giocatori accedano alla partita, chiunque voglia connettersi per giocare dovrà inserire un codice generato dal server su PC ed essere autenticato dal server che verificherà la correttezza del codice inserito.

Ci si aspetta quindi che il server possa generare codici alfanumerici di pochi caratteri automaticamente non appena si avvia la schermata di connessione e che riceva degli input dai controller contenenti il codice inserito dagli utenti e solo dopo aver verificato che il codice ricevuto da ciascun giocatore sia uguale a quello generato ammetterà ogni giocatore alla partita.
Il server inoltre dovrà assegnare al primo giocatore che riesce ad autenticarsi con successo lo status di "Giocatore_1" ed informare i giocatori di ciò, assicurandosi che ciascun giocatore riconosca il "numero" assegnatogli.

```
[TestConnessione]
public void TestConnessione{
        ClientConnectionManager connectionManager = new ClientConnectionManager();
        connectionManager.sendConfirmationCode();
        Assert.True( ClientConnectionManager.isConfirmationSucceded());
}
```

Setup: Il server dovrà memorizzare le impostazioni ricevute dai giocatori riguardo le impostazioni di gioco e le personalizzazioni delle navette, e fornire questi dati al GameManager che li utilizzerà per avviare la partita.

Gioco: Il server dovrà ricevere tutti gli input ricevuti dai controller e tradurli in *real time* in azioni sulla schermata di gioco. Dovrà gestire il punteggio e tenere in memoria e costantemente aggiornare lo stato di ogni navetta in termini di punti salute e bonus.
Dovrà inoltre saper gestire eventuali interruzioni dovute alla disconnessione di uno o più giocatori aprendo automaticamente la schermata di "timeout".
Al raggiungimento del punteggio massimo deciso precedentemente da parte di un giocatore qualsiasi dovrà concludere la partita, salvare i punteggi, e mostrare la schermata finale contenente la classifica della partita conclusa e le opzioni per rigiocare od uscire, opzioni che dovrà saper gestire.

```
[TestGameplay]
public void GameplayTest(){

        int[] scores = scoreManager.getScores();
        while(testShip.getHealth() != 0){
                testShip.collideWithProjectile();
        }
        Assert.AreNotEqual(score, scoreManager.getScores());

        int maxScore = scoreManager.getMaxKills();
        for(int i = 0; i<maxScore; i++){
                scoreManager.setScorePlayer1();
        }
        Assert.That(Application.Quit());

}
```

5.0 PROGETTAZIONE

5.1 Requisiti non funzionali

L'unico requisito non funzionale emerso nell'analisi del problema è quello della performance, essendo il videogioco un sistema in tempo reale. La connessione è comunque prevista in rete locale, quindi non si prevedono problemi di ritardi. Inoltre, non ci sono algoritmi che richiedono particolare potenza di calcolo, quindi questo non inciderà negativamente sulle prestazioni.

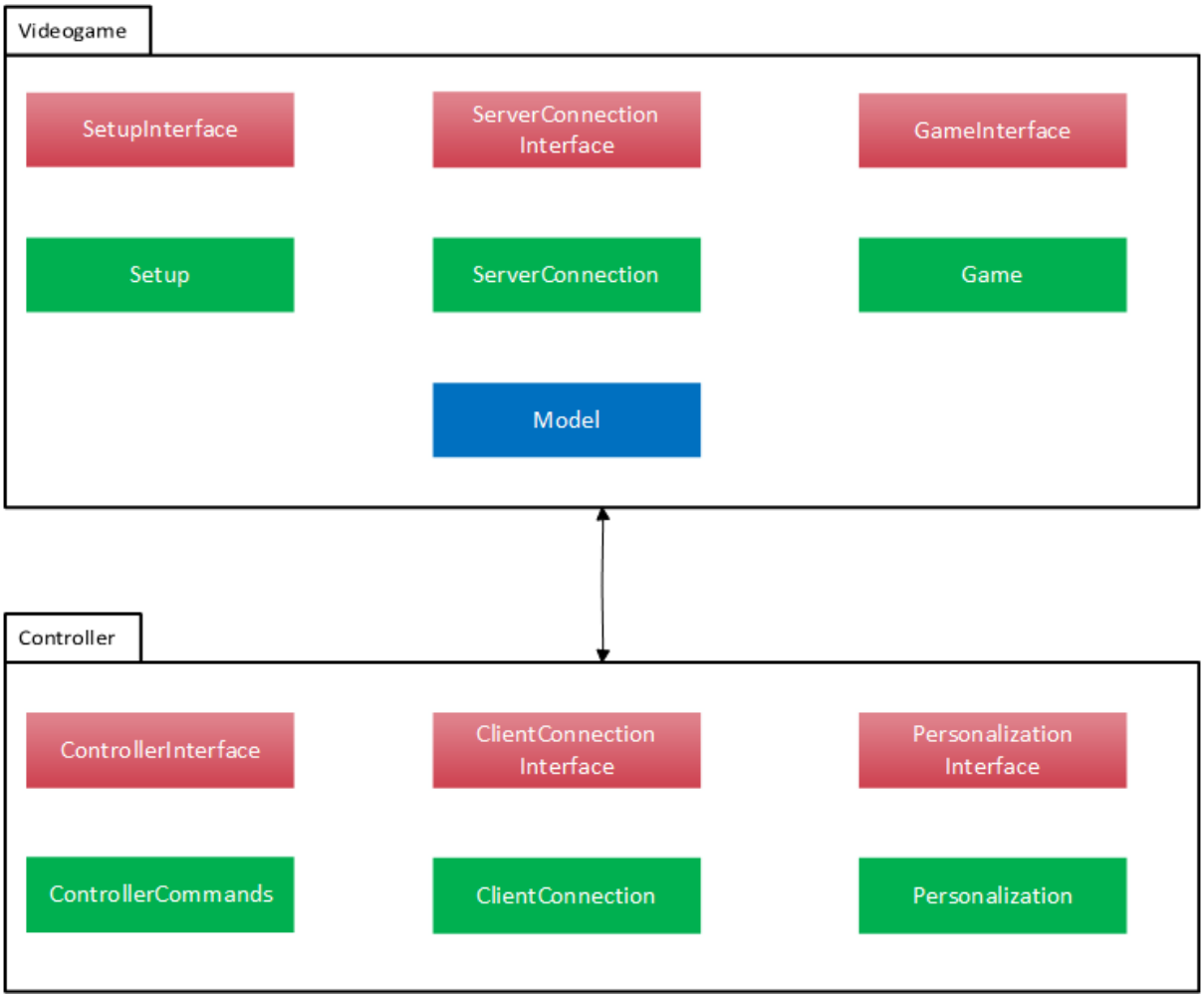
Per avere una connessione più veloce e per semplificare la gestione della connessione è stata scelta la tecnologia WiFi piuttosto che quella Bluetooth.

5.2 Scelta dell'architettura

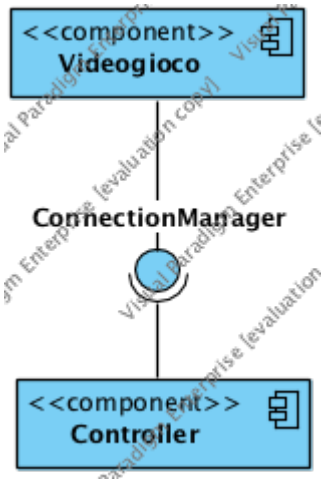
Per semplificare l'applicazione Smartphone ed essendo il videogioco ad elaborare i comandi è stato scelto il pattern thin Client/Server, dove il Server è il Videogioco, sviluppato secondo il design architetturale MVC, e il thin Client il Controller.

Come ambiente di sviluppo è stato scelto Unity3D, che consente di attenersi al principio di Design for Change, essendo possibile sviluppare applicazioni per sistemi operativi diversi in contemporanea, e di avere semplificato il rendering grafico.

Nella figura sottostante è raffigurata l'architettura del sistema tramite un diagramma dei package.



Nella figura sottostante è rappresentata l’architettura del sistema raffigurata tramite un diagramma dei package.



5.3 Progettazione di Dettaglio

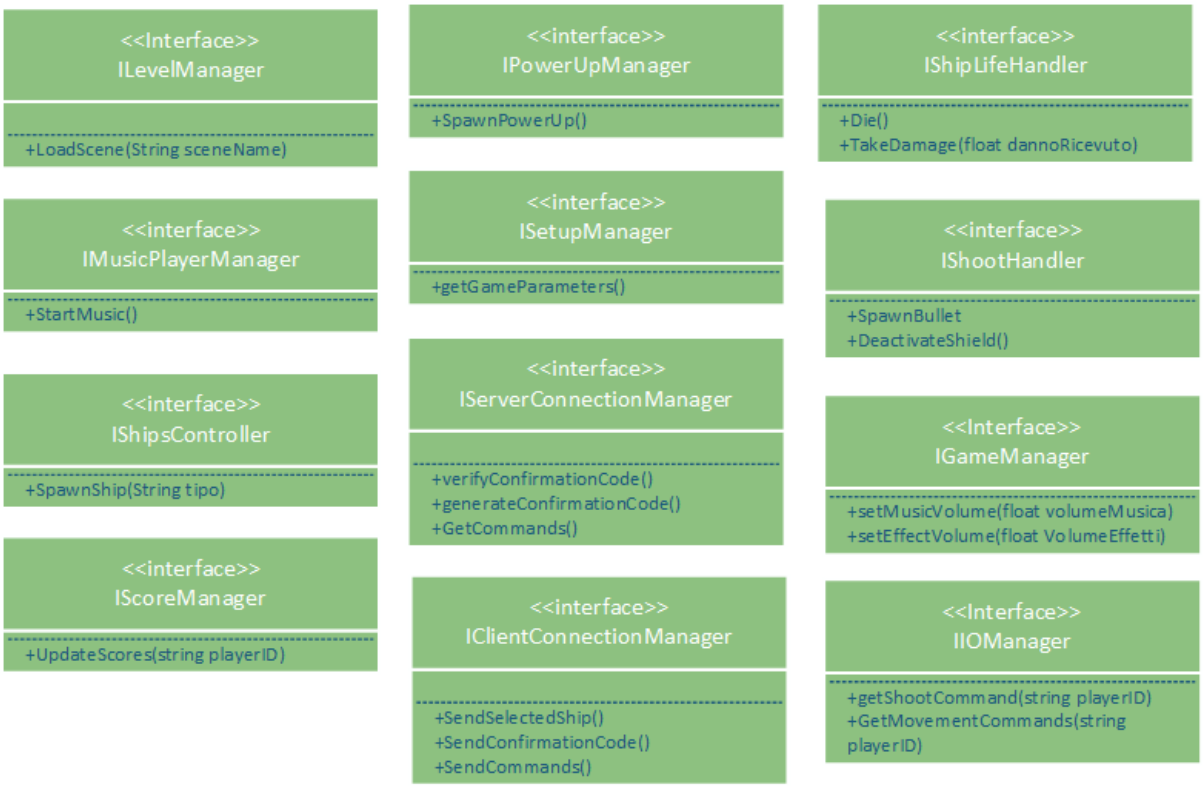
Essendo Unity3D la piattaforma di sviluppo scelta, i diagrammi delle classi rispecchieranno la struttura che devono avere in questo ambiente.

Si rende quindi necessaria una breve descrizione del funzionamento dei componenti più importanti propri di Unity3D, direttamente citando il manuale:

Componente	**Descrizione**
Scene	**Scenes** contain the environments and menus of your game. Think of each unique Scene file as a unique level. In each Scene, you place your environments, obstacles, and decorations, essentially designing and building your game in pieces.
GameObject	**GameObjects** are the fundamental objects in Unity that represent characters, props and scenery. They do not accomplish much in themselves but they act as containers for **Components**, which implement the real functionality.
Component	**Components** are the nuts & bolts of objects and behaviors in a game. They are the functional pieces of every **GameObject**.
Transform Component	The **Transform** component determines the **Position**, **Rotation**, and **Scale** of each object in the scene. Every GameObject has a Transform.
Collider 2D Component	**Collider 2D** components define the shape of a 2D GameObject for the purposes of physical collisions.
Rigidbody 2D Component	A **Rigidbody 2D** component places an object under the control of the physics engine.
Canvas	The **Canvas** component represents the abstract space in which the UI is laid out and rendered. All UI elements must be children of a GameObject that has a Canvas component attached.
Text	The **Text** control displays a non-interactive piece of text to the user.
Image	The **Image** control displays a non-interactive image to the user. This can be used for decoration, icons, etc
Event manager	This subsystem is responsible for controlling all the other elements that make up eventing. It coordinates which Input Module is currently active, which GameObject is currently considered ‘selected’, and a host of other high level Event

	System concepts. Used for user interfaces.
Button	The **Button** control responds to a click from the user and is used to initiate or confirm an action.
Toggle	The **Toggle** is a checkbox that allows users to turn an option on or off.
Slider	The **Slider** control allows the user to select a numeric value from a predetermined range by dragging the mouse.
MonoBehavior	**MonoBehaviour** is the base class from which every Unity script derives.When you use C#, you must explicitly derive from MonoBehaviour. The most important methods inherited by this class are: ● *Awake()*, called when the script instance is being loaded; ● *Start()*, called on the frame when a script is enabled just before any of the Update methods are called the first time and right after the *Awake()* function; ● *Update()*, called every frame, if the MonoBehaviour is enabled.
Network Manager	This class configures the network interface and all the network parameters. You use it to set up a server or connect to one and have a row of helper functions to help you with those tasks.
NetworkDiscovery	The **NetworkDiscovery** component allows Unity games to find each other on a local network. It can broadcast presence and listen for broadcasts, and optionally join matching games using the NetworkManager.

5.3.1 Diagramma di dettaglio: Interfacce del videogioco



Queste interfacce permettono di applicare The Dependency Inversion Principle.

5.3.2 Diagrammi di dettaglio videogioco

Diagramma di dettaglio Dominio-Videogioco
Nell'ottica di rispettare i principi di dependancy inversion e di single responsability, vengono scomposte le varie classi che si occupano di effettuare azioni particolari, come sparare. Nell'Analisi del Problema erano state divise le Azioni possibili eseguite da una singola *Ship*, per cui ci rifaremo proprio a quelle per generare le classi che si occuperanno di singoli compiti che riguardano le *Ship*.
Quelle che erano state definite come "<<*Action*>> Idle" e "<<*Action*>> Move" vengono accorpate in *Ship* e quella che era stata definita come "<<*Action*>> Dead" verrà invece presa in carico dalla classe *ShipLifeHandler*, che sarà quella con cui interagiranno anche le classi *PowerupHandler* e Bullet.

La collisione con il proiettile viene invece ridefinita, facendo in modo che sia *Bullet* a gestire la collisione con una Ship. Essendo previste Ship di diverso tipo con diverse statistiche, si è scelto di inserire nella classe generica che le riguarda un *damageDealtModifier*, valore compreso tra 0.0 e 1.0, che andrà a modificare il *damageDealt* del *Bullet* quando questo viene istanziato dalla classe ShootHandler.

Game viene invece eliminato dal diagramma delle classi, in quanto ad occuparsi delle sue funzioni sono i vari Manager del Videogioco (*GameManager, ScoreManager, ShipManager, PowerupManager* e *IOManager*).

Diagramma di Dettaglio: Dominio-Powerup

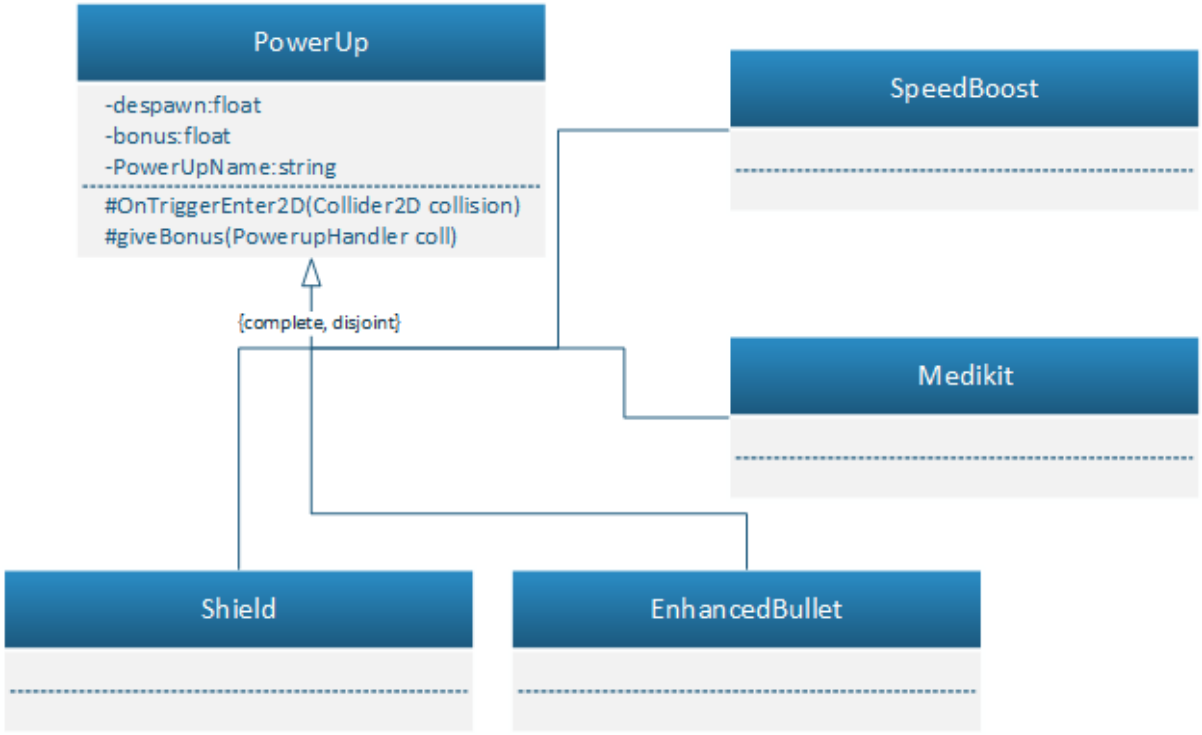


Diagramma di Dettaglio: Dominio-Ship

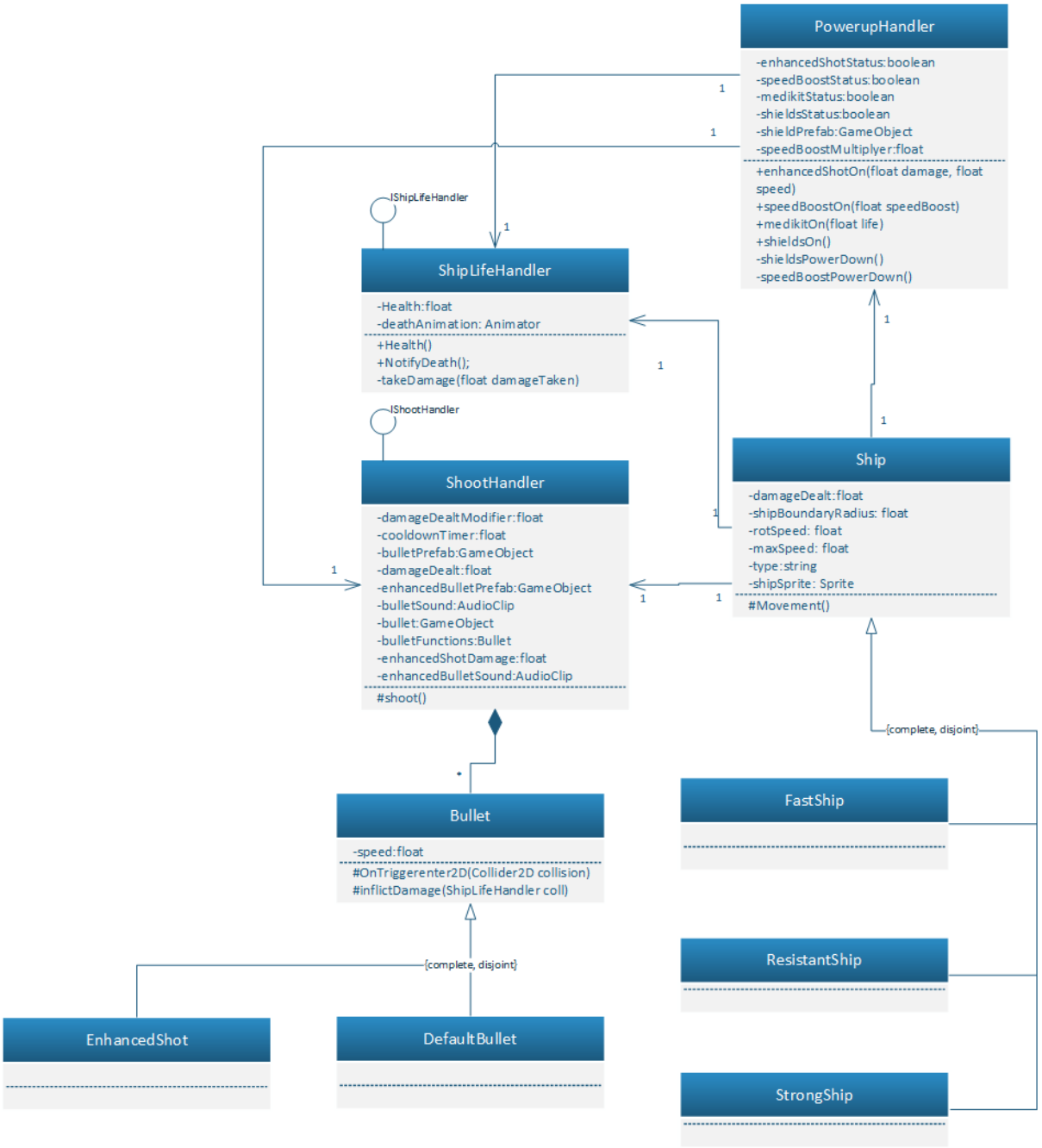
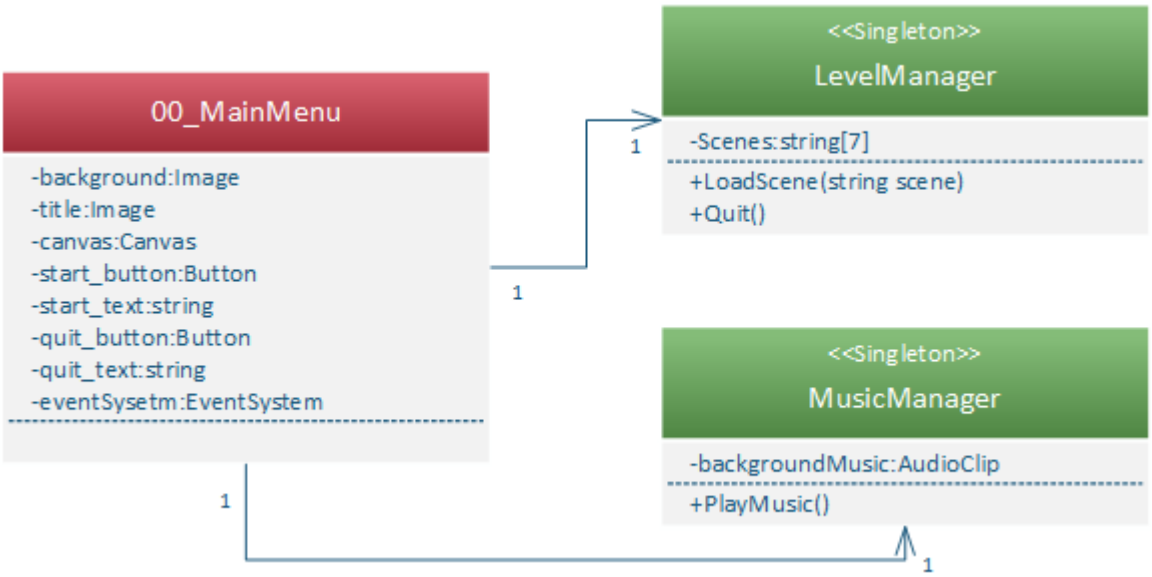


Diagramma di Dettaglio: Interfaccia di MainMenu



I due Manager che gestiscono il cambio di livello e la musica è stato deciso di definirli rispettando il Design Pattern Singleton, dato che questi gestori devono essere tramandati di classe in classe e di cui ne deve esistere una ed una sola copia.

Qui sotto si riporta la schermata di Interfaccia del MainMenu.

Diagramma di Dettaglio: Interfacce e Manager Connessione

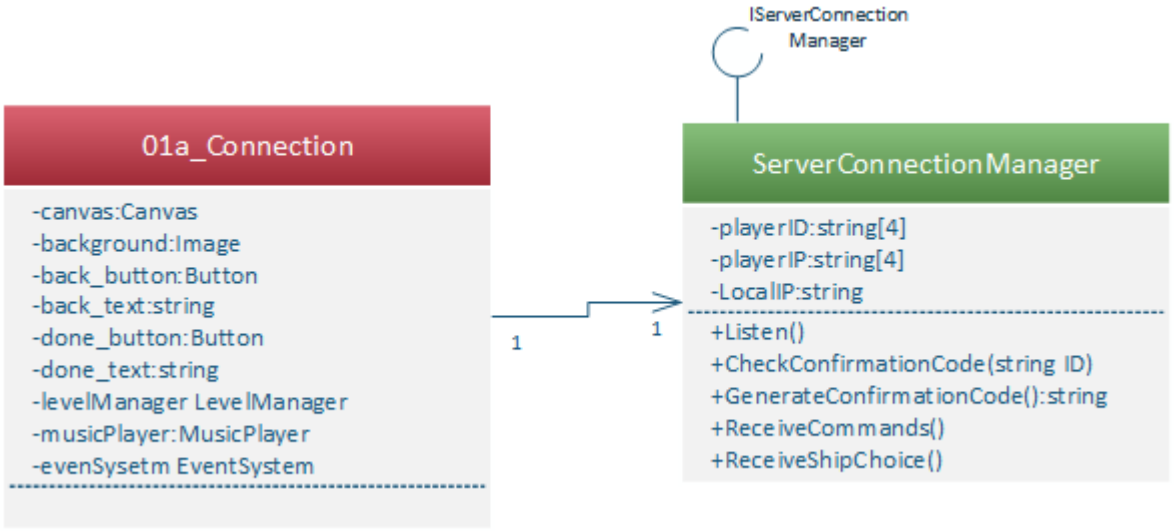
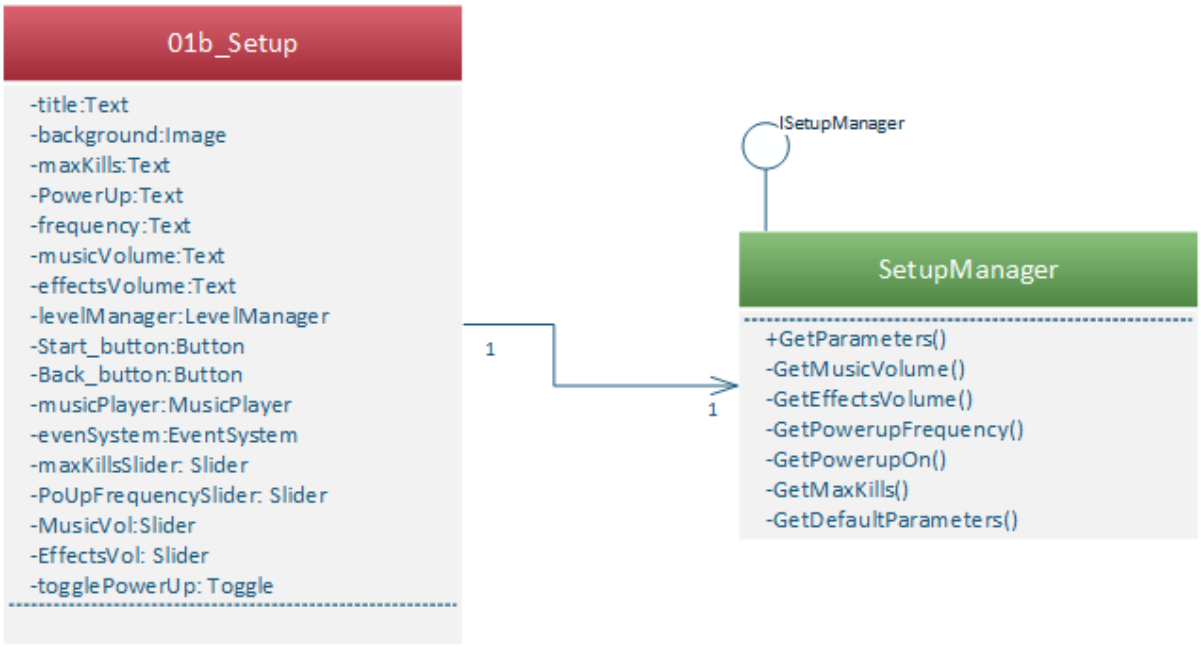
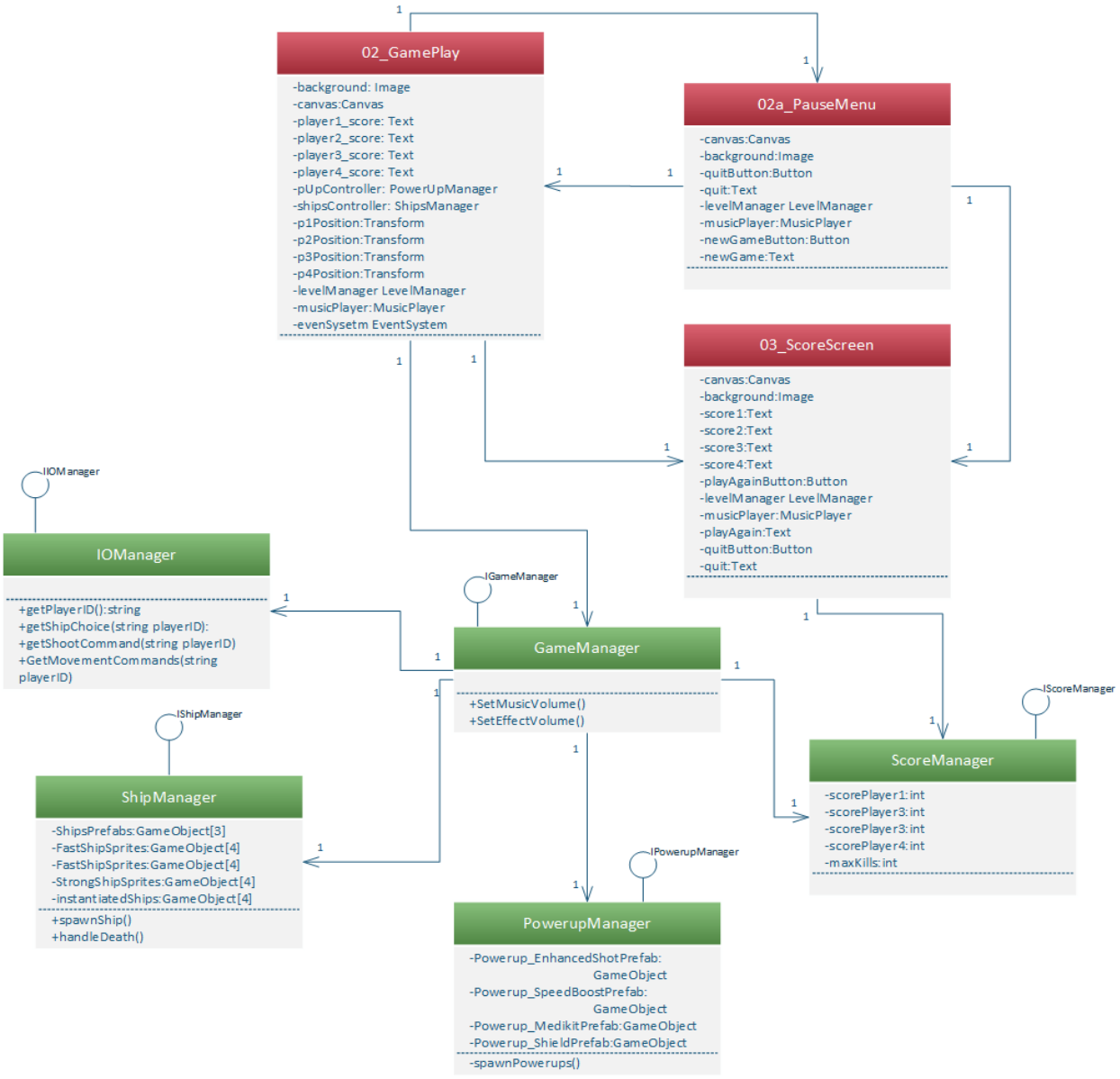


Diagramma di Dettaglio: Interfacce e Manager Setup



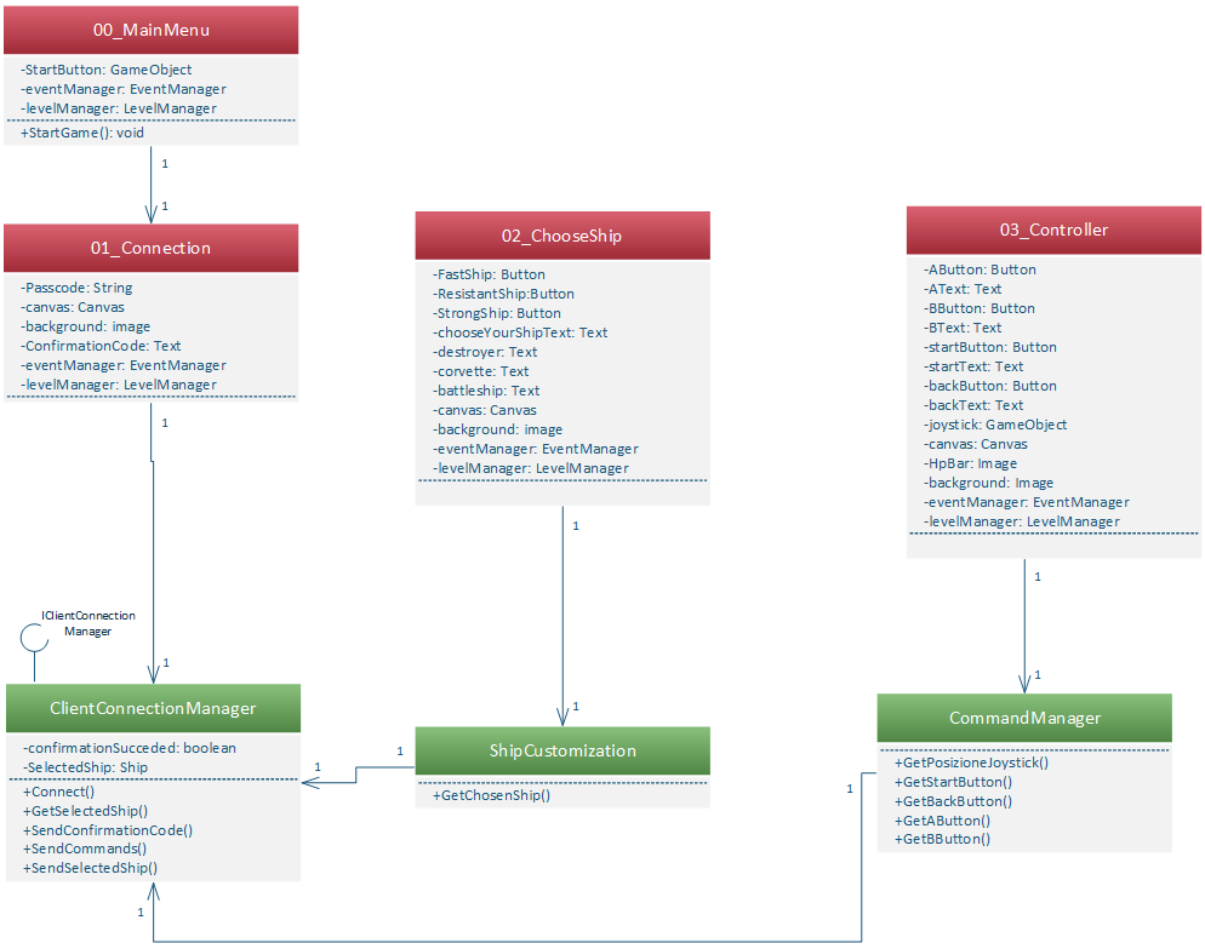
Qui sotto si riporta la schermata di Interfaccia delle Impostazioni.

Diagramma di Dettaglio: Interfacce e Manager in package GameInterface e Game

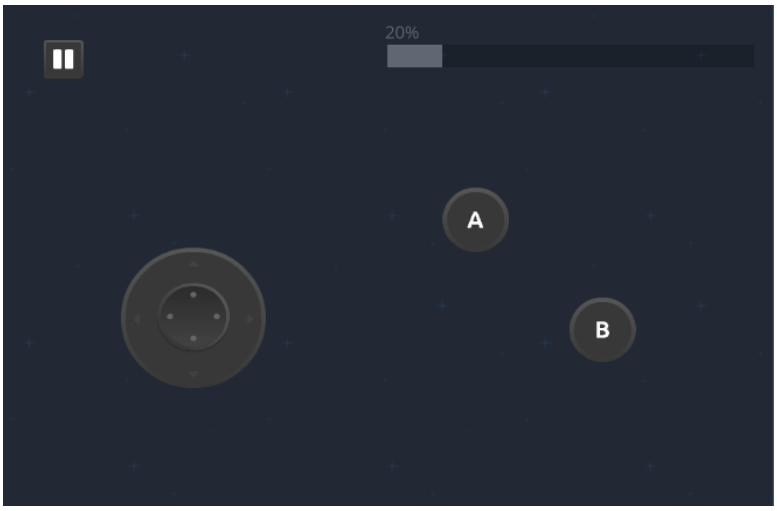


Qui sotto si riporta la schermata di Interfaccia del GamePlay ad inizio partita.

5.3.3 Diagramma di dettaglio Controller



Prototipo di interfaccia del Controller



5.4 Interazioni

5.4.1 Diagramma di sequenza

Diagramma di sequenza: Connessione eseguita con successo

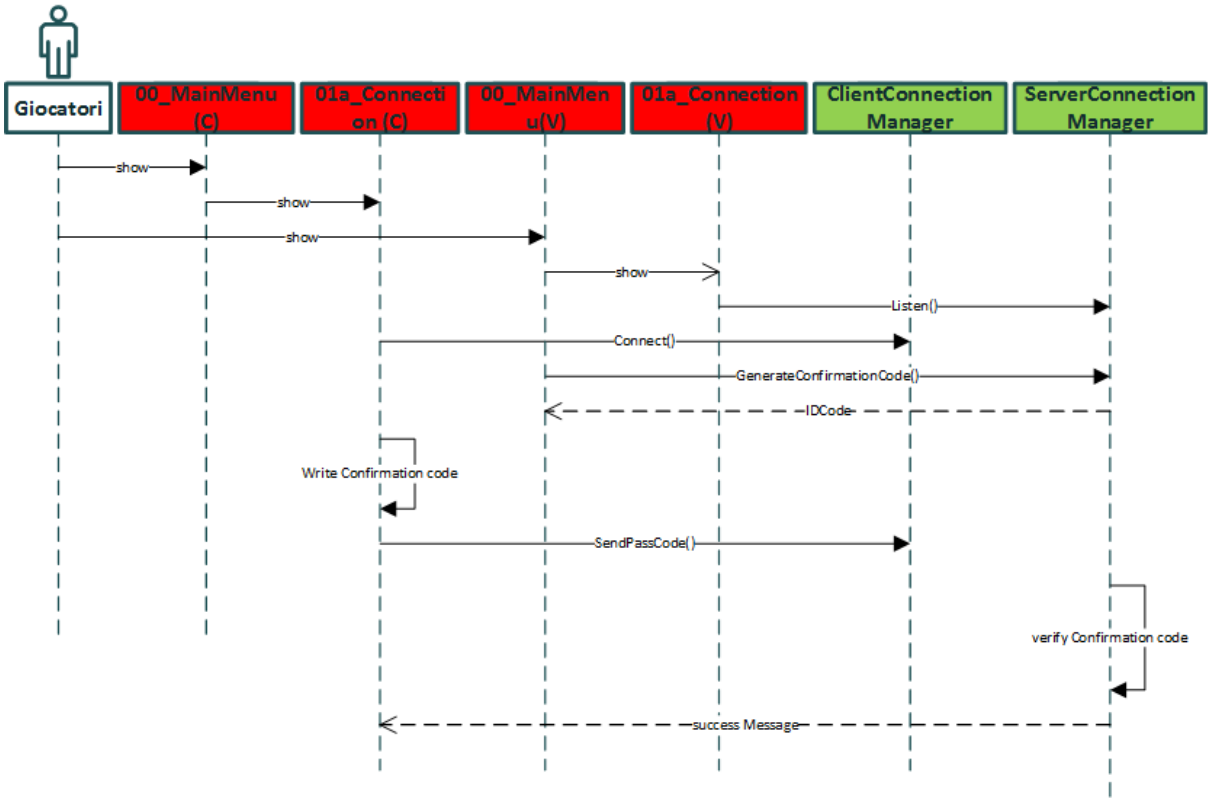


Diagramma di sequenza: Scelta navicelle eseguita con successo

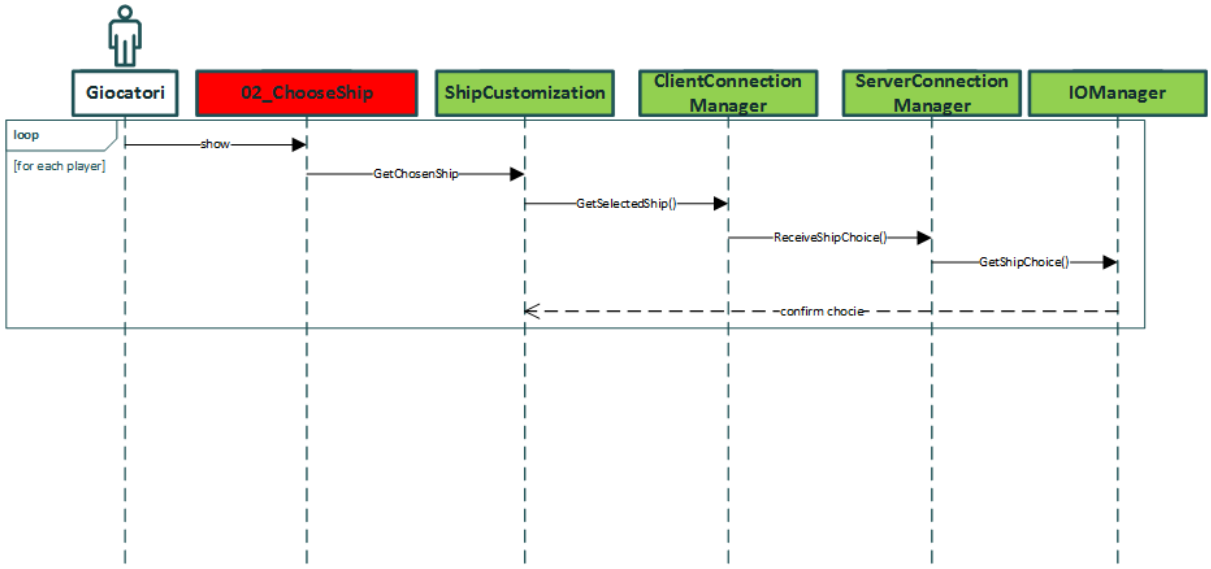
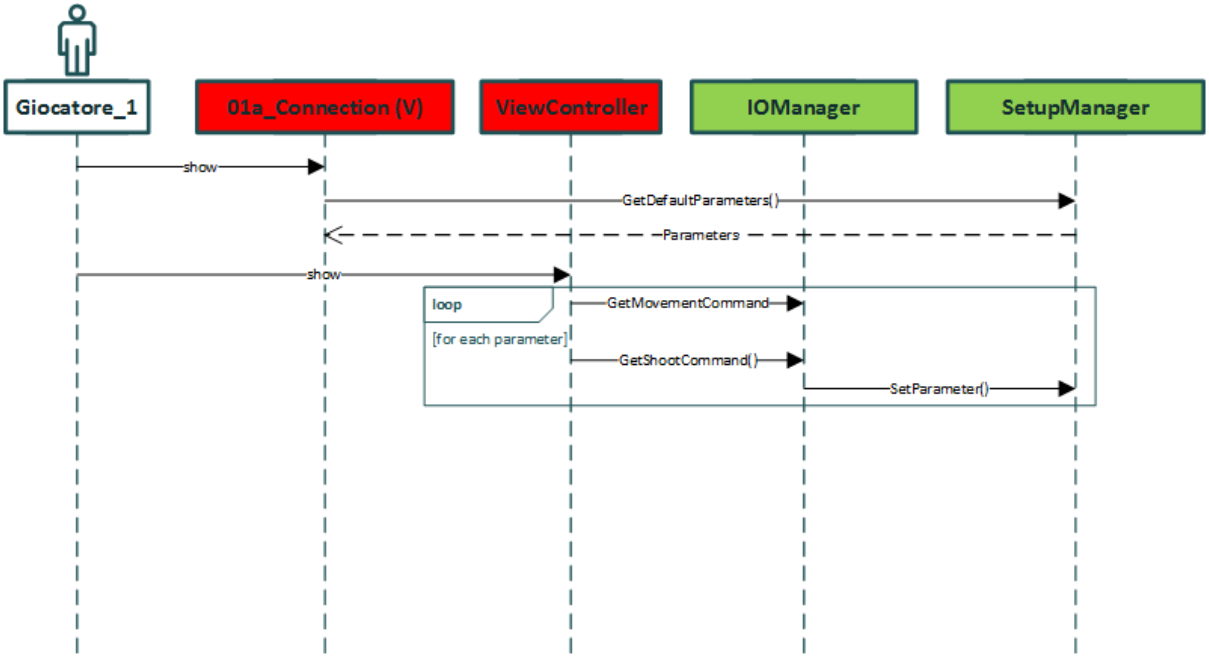


Diagramma di sequenza: Scelta impostazioni eseguita con successo

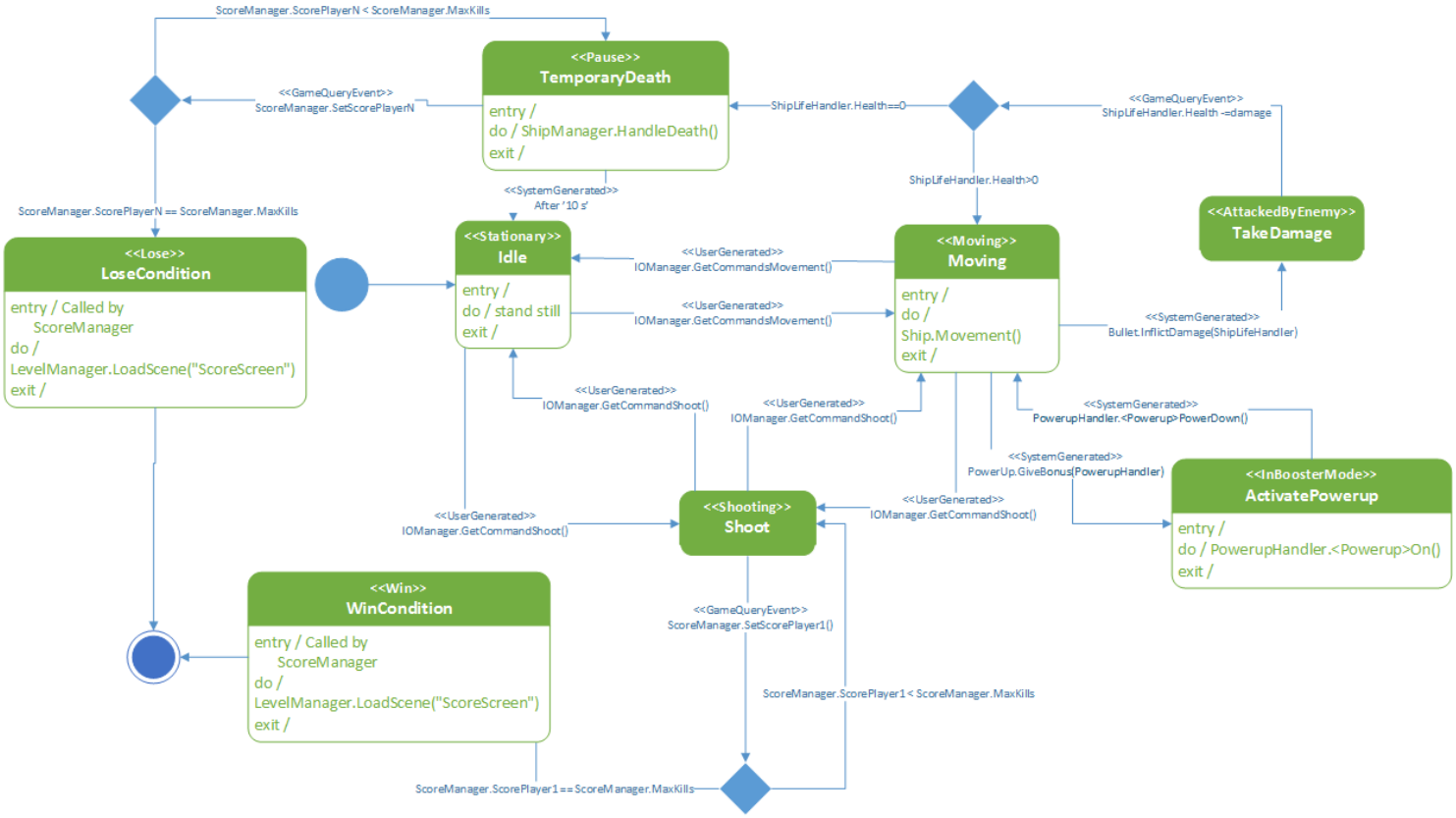


GetMovementCommand permetterà di muoversi sulla schermata del videogioco del computer, mentre getShootCommand (chiamato tasto A sull'interfaccia grafica del controller), permetterà di selezionare l'opzione voluta.

5.5 Comportamento

È stato aggiornato il diagramma dei componenti dell'Analisi del Problema con le nuove funzioni e la divisione dei compiti tra Handler e Manager in modo da render più chiara l'implementazione degli algoritmi.

Si è presa in esame una generica navicella *Ship* e il giocatore *Player1*. Il generico giocatore che prende punti dalla morte della Ship in esame è descritto come *PlayerN*.



5.6 TEST DI PROGETTAZIONE

```
[TestConnessione]
public void TestConnessione{
	ConnectionManager clientManager = new ClientConnectionManager();
	clientManager.sendPasscode();
	Assert.IsTrue( connectionManager.isConfirmationSucceded());
}
```

```
[TestGameplay]
public void GameplayTest(){
	[SerializeField]
	GameObject ShipPrefab;
	Ship testShip = Instatiate(ShipPrefab, gameObject.transform);
	ScoreManager scoreManager = new IScoreManager();
	ShipManager shipManager = new ShipManager();

//1) Navetta si muove

	Vector3 originalPosition = testShip.transform.position;
	testShip.Update();
	Assert.AreNotEqual(originalPosition, testShip.transform.position);

//2) Navetta prende danno

	float hp = testShip.getHealth();
	testShip.collideWithProjectile();
	Assert.AreNotEqual(hp, testShip.getHealth());

//3) Navetta muore

	int[] scores = scoreManager.getScores();
	while(testShip.getHealth() != 0){
		testShip.collideWithProjectile();
	}
	Assert.IsTrue(shipManager.getComponentInChildren.Ship() == null);
	Assert.AreNotEqual(score, scoreManager.getScores());

//4) Fine partita

	int maxScore = scoreManager.getMaxKills();
	for(int i = 0; i<maxScore; i++){
		scoreManager.setScorePlayer1();
	}
	Assert.AreEqual(SceneManager.getActiveScene(), "03_ScoreScreen");

}
```

5.7 Diagramma di Deployment