

DON'T PANIC

3DMob: Grafica 3D su device mobili



Specifica Tecnica

Informazioni sul documento

Versione	3.2.0
Redazione	Busato Luca Cesarato Fabio Lain Daniele Pezzutti Marco Rampazzo Federico
Verifica	Basaglia Mattia
Responsabile	Sciarrone Riccardo
Uso	Esterno
Lista di distribuzione	Don't Panic Prof. Vardanega Tullio Prof. Cardin Riccardo

Descrizione

Specifica tecnica e architettura dell'applicazione 3DMob



Diario delle modifiche

Descrizione modifica	Autore	Ruolo	Data	Versione
Approvazione documento	Sciarrone Riccardo	Responsabile	2013-01-30	3.2.0
Verifica del documento	Basaglia Mattia	Verificatore	2013-01-26	3.1.0
Stesura ed importazione tracciamento	Pezzutti Marco	Progettista	2013-01-25	3.0.7
Stesura stime di fattibilità e di bisogno di risorse	Lain Daniele	Responsabile	2013-01-23	3.0.6
Stesura descrizione design pattern _G	Rampazzo Federico	Progettista	2013-01-22	3.0.5
Stesura descrizione dei singoli componenti	Busato Luca	Progettista	2013-01-21	3.0.4
Stesura della architettura generale	Cesarato Fabio	Progettista	2013-01-18	3.0.3
Stesura del metodo e formalismo di specifica	Rampazzo Federico	Progettista	2013-01-18	3.0.2
Stesura tecnologie utilizzate	Lain Daniele	Progettista	2013-01-17	3.0.1
Creazione scheletro del documento e stesura introduzione	Pezzutti Marco	Analista	2013-01-16	3.0.0



Indice

1	Introduzione	1
1.1	Scopo del documento	1
1.2	Scopo del Prodotto	1
1.3	Glossario	1
1.4	Riferimenti	1
1.4.1	Normativi	1
1.4.2	Informativi	1
2	Tecnologie Utilizzate	2
2.1	C++	2
2.2	Qt	2
2.2.1	Segnali e Slot	2
2.2.2	Qt Plugins e Qt3D	3
2.2.3	Qt3D	3
2.2.4	QMake	4
3	Descrizione architettura	5
3.1	Metodo e formalismo di specifica	5
3.2	Architettura generale	5
3.2.1	Model	8
3.2.2	View	9
3.2.3	Controller	10
4	Componenti	11
4.1	DDDMob	11
4.2	DDDMob::Model	12
4.3	DDDMob::Model::CHelpModel	12
4.4	DDDMob::Model::CBasicEditor	13
4.5	DDDMob::Model::CBasicEditor::CConverter	14
4.6	DDDMob::Model::CBasicEditor::CConverter::CSettingsModel	15
4.7	DDDMob::Model::CBasicEditor::CConverter::CExportModel	16
4.8	DDDMob::Model::CBasicEditor::CConverter::CLoaderModel	17
4.9	DDDMob::Controller	18
4.10	DDDMob::Controller::CSceneController	19
4.11	DDDMob::View	20
4.12	DDDMob::View::CDockWidgets	21
4.13	DDDMob::View::CMainWindow	22
4.14	DDDMob::View::CHelpWindow	23
4.15	DDDMob::View::CSettingsWindow	24
4.16	DDDMob::View::CErrorWindow	25
4.17	DDDMob::View::CFileWindow	25
4.18	DDDMob::View::CWidgetElement	26
4.19	DDDMob::C3DObject	27



5	Classi	29
5.1	Model	29
5.1.1	Help	29
5.1.2	Setting	29
5.1.3	DeviceLimit	30
5.1.4	NumericPrecisionType	30
5.1.5	SceneExporter	30
5.1.6	ExporterJSON	31
5.1.7	ExporterUBJSON	31
5.1.8	ExporterXML	32
5.1.9	ExporterMinifiedJSON	32
5.1.10	Scene	32
5.1.11	SceneAdapter	33
5.2	Controller	33
5.2.1	Controller	33
5.2.2	CommandTransform	34
5.2.3	CommandAddLight	34
5.2.4	CommandSelectObject	35
5.2.5	CommandEditMesh	35
5.2.6	CommandEditLight	35
5.2.7	CommandScene	36
5.2.8	CommandAddElement	36
5.2.9	CommandPosition	37
5.2.10	CommandScale	37
5.2.11	CommandRotation	37
5.2.12	CommandLightType	37
5.2.13	CommandShiness	38
5.2.14	CommandEmission	38
5.2.15	CommandColor	38
5.2.16	CommandColorEmission	39
5.2.17	CommandColorSpecular	39
5.2.18	CommandColorDiffuse	39
5.3	View	39
5.3.1	FacadeView	39
5.3.2	About	40
5.3.3	Material	40
5.3.4	Selector	41
5.3.5	Light	41
5.3.6	Action	42
5.3.7	Transformation	42
5.3.8	WindowWithMenu	43
5.3.9	WindowWithDock	43
5.3.10	WindowWithStatus	43
5.3.11	Decorator	44
5.3.12	Window	44
5.3.13	View3D	44
5.3.14	HelpWindow	45
5.3.15	SettingsWindow	45
5.3.16	ErrorMessageBox	46



5.3.17	FileOpener	46
5.3.18	FileExporter	46
5.3.19	AxisSlider	47
5.3.20	ColorPicker	47
5.4	C3DObject	48
5.4.1	SceneObject	48
5.4.2	Light	48
5.4.3	Mesh	48
5.4.4	LightType	49
6	Diagrammi di attività	50
6.0.5	Attività principali	51
6.0.6	Caricare File	52
6.0.7	Modificare Scena	53
6.0.8	Modificare Camera	54
6.0.9	Salvare File	55
6.0.10	Impostare Preferenze	56
6.0.11	Aprire Aiuto contestuale	57
7	Design Pattern	58
7.1	Design Pattern architetturali	58
7.1.1	MVC	58
7.2	Design Pattern creazionali	60
7.2.1	Singleton	60
7.3	Design Pattern strutturali	60
7.3.1	Adapter	60
7.3.2	Facade	61
7.3.3	Decorator	62
7.4	Design Pattern comportamentali	62
7.4.1	Command	62
7.4.2	Strategy	64
8	Stime di fattibilità e di bisogno di risorse	65
9	Tracciamento	66
9.1	Tracciamento componenti - requisiti	66
9.2	Tracciamento requisiti - componenti	69
A	Descrizione Design Pattern	75
A.1	Design Pattern architetturali	75
A.1.1	MVC	75
A.2	Design Pattern creazionali	76
A.2.1	Singleton	76
A.3	Design Pattern strutturali	76
A.3.1	Adapter	76
A.3.2	Facade	77
A.3.3	Decorator	78
A.4	Design Pattern comportamentali	79
A.4.1	Command	79
A.4.2	Strategy	79



B	Mockup interfaccia grafica	81
B.1	Finestra principale	81
B.2	Finestra di configurazione	82
B.3	Finestra di selezione file da importare	82
B.4	Finestra di esportazione file	83
B.5	Finestra di aiuto	84
B.6	Finestra di informazioni sul sistema	84
B.7	Finestra dei messaggi di sistema	84
B.8	Finestra degli errori di sistema	85



Elenco delle tabelle

2	Tabella componenti/requisiti	68
3	Tabella requisiti / componenti	74



Elenco delle figure

1	Architettura generale dell'applicazione - vista package	6
2	Architettura generale dell'applicazione	7
3	Diagramma delle classi del Model	8
4	Diagramma delle classi del View	9
5	Diagramma delle classi del Controller	10
6	Componente DDDMob	11
7	Componente DDDMob::Model	12
8	Componente DDDMob::Model::CHelpModel	12
9	Componente DDDMob::Model::CBasicEditor	13
10	Componente DDDMob::Model::CBasicEditor::CConverter	14
11	Componente DDDMob::Model::CBasicEditor::CConverter::CSettingsModel	15
12	Componente DDDMob::Model::CBasicEditor::CConverter::CExportModel	16
13	Componente DDDMob::Model::CBasicEditor::CConverter::CLoaderModel	17
14	Componente DDDMob::Controller	18
15	Componente DDDMob::Controller::CSceneController	19
16	Componente DDDMob::View	20
17	Componente DDDMob::View::CDockWidgets	21
18	Componente DDDMob::View::CMainWindow	22
19	Componente DDDMob::View::CHelpWindow	23
20	Componente DDDMob::View::CSettingsWindow	24
21	Componente DDDMob::View::CErrorWindow	25
22	Componente DDDMob::View::CFileWindow	25
23	Componente DDDMob::View::CWidgetElement	26
24	Componente DDDMob::C3DObject	27
25	Diagramma attività - Attività principali dell'applicativo 3DMob	51
26	Diagramma attività - Caricamento di un file	52
27	Diagramma attività - Modifica della scena	53
28	Diagramma attività - Modifica della camera	54
29	Diagramma attività - Salvataggio della scena	55
30	Diagramma attività - Impostazione delle preferenze del sistema	56
31	Diagramma attività - Apertura della finestra di aiuto contestuale	57
32	Diagramma di sequenza - Modifica della scena	58
33	Diagramma di sequenza - Modifica della scena con errore	59
34	Diagramma di sequenza - Selezione ed apertura della vista	59
35	Diagramma del Design Pattern Singleton	60
36	Applicazione di Adapter in 3DMob	60
37	Applicazione di Facade in 3DMob	61
38	Applicazione di Decorator in 3DMob	62
39	Applicazione di Command in 3DMob	62
40	Diagramma di sequenza - Gestore di comandi della scena	63
41	Applicazione di Strategy in 3DMob	64
42	Diagramma del Design Pattern MVC	75
43	Diagramma del Design Pattern Singleton	76
44	Diagramma del Design Pattern Adapter	76
45	Diagramma del Design Pattern Facade	77
46	Diagramma del Design Pattern Decorator	78
47	Diagramma del Design Pattern Command	79



48	Diagramma del Design Pattern Strategy	79
49	Mockup della finestra principale dell'applicazione 3DMob	81
50	Mockup del menu file	82
51	Mockup della finestra per la configurazione delle impostazioni	82
52	Mockup della finestra usata per selezionare il file in fase di importazione	83
53	Mockup della finestra usata per selezionare il file in fase di esportazione	83
54	Mockup della finestra di aiuto	84
55	Mockup della finestra di informazioni di sistema	84
56	Mockup della finestra di messaggio di sistema	85
57	Mockup della finestra di errore di sistema	85



1 Introduzione

1.1 Scopo del documento

Il presente documento ha lo scopo di definire la progettazione ad alto livello del progetto 3DMob.

Verrà presentata l'architettura generale secondo la quale saranno organizzate le varie componenti software e saranno descritti i Design Pattern_G utilizzati.

1.2 Scopo del Prodotto

Lo scopo del progetto è la realizzazione di un'applicazione in grado di convertire file prodotti da programmi di grafica 3D in file in formato JSON_G in grado di essere visualizzati su dispositivi mobile senza perdita di informazione. L'obiettivo è quello di semplificare il workflow attuale necessario a rendere compatibili i file.

1.3 Glossario

Al fine di evitare ogni ambiguità di linguaggio e massimizzare la comprensione dei documenti, i termini tecnici, di dominio, gli acronimi e le parole che necessitano di essere chiarite, sono riportate nel documento *Glossario v3.2.0*.

Ogni occorrenza di vocaboli presenti nel *Glossario* è marcata da una "G" maiuscola in pedice.

1.4 Riferimenti

1.4.1 Normativi

- **Analisi dei Requisiti:** *Analisi dei Requisiti v3.2.0*;
- **Norme di Progetto:** *Norme di Progetto v3.2.0*.

1.4.2 Informativi

- Documentazione Qt_G per Segnali e Slot
<http://qt-project.org/doc/qt-5.0/qtcore/signalsandslots.html>;
- Documentazione per i plugin di Qt_G
<http://doc.qt.digia.com/qt/plugins-howto.html>;
- Documentazione per i plugin di Qt3D
<http://doc.qt.digia.com/qt-quick3d-snapshot/sceneformats-obj.html>;
- Descrizione dei Design Pattern_G
http://sourcemaking.com/design_patterns;
- Design Patterns: Elementi per il riuso di software a oggetti - E. Gamma, R. Helm, R. Johnson, J. Vlissides - 1^a Edizione (2002)



2 Tecnologie Utilizzate

In questa sezione vengono descritte le tecnologie su cui si basa lo sviluppo del progetto e la motivazione del loro utilizzo.

2.1 C++

Si è deciso di utilizzare il C++ poiché offre i seguenti vantaggi:

- **Librerie:** il C++ mette a disposizione numerose librerie, descritte successivamente, che permettono di semplificare la realizzazione di alcune funzionalità richieste come il parsing dei file e la realizzazione di interfacce 3D;
- **Conoscenza:** il gruppo ha già avuto modo di utilizzare il linguaggio C++, sia per progetti di altri corsi universitari sia per esperienze personali, permettendo di raggiungere un buon grado di conoscenza;
- **Strumenti di analisi:** la maggior parte dei componenti del gruppo ha utilizzato gli strumenti di analisi che permettono di verificare dinamicamente il codice prodotto.

2.2 Qt

Si è deciso di scegliere il framework_G Qt_G perché offre i seguenti vantaggi:

- **Funzionalità 3D:** tra le numerose librerie disponibili sono presenti alcune funzionalità che semplificheranno lo sviluppo del software;
- **Qt_G Creator:** IDE_G ottimizzato per lo sviluppo con Qt_G;
- **Qt_G Designer:** editor grafico per disegnare l'interfaccia grafica;
- **Qt_G Linguist:** tool in grado di semplificare il supporto e la traduzione per numerose lingue;
- **Crossplatform:** Qt_G è un framework_G crossplatform e tra le piattaforme supportate ufficialmente sono presenti Windows_G, OS X e molte distribuzioni GNU/Linux_G;
- **Esperienza del team:** tutti i membri del team hanno già lavorato con Qt_G durante l'insegnamento di Programmazione ad Oggetti;
- **Licenza GNU LGPL_G v2.1:** Qt_G è disponibile con licenza GNU LGPL_G v2.1 e può essere quindi utilizzato nel nostro progetto senza che sia necessario rilasciare il codice prodotto.

2.2.1 Segnali e Slot

Il framework_G Qt_G offre un sistema di comunicazione tra oggetti tramite l'uso di Signal_G. Qt_G permette di connettere un Signal_G ad un puntatore di funzione o ad un oggetto di funzione. Quando un oggetto emette un Signal_G tutti i funtori_G connessi ad esso vengono eseguiti nell'ordine in cui questi sono stati connessi al Signal_G.



2.2.1.1 Descrizione

Un Signal_G è un metodo pubblico la cui implementazione viene generata dal meta-object compiler $_G$ di Qt $_G$.

Essendo parte del sistema di meta-informazioni è possibile definire Signal_G solo in classi che derivano da `QObject` che contengono la macro `Q_OBJECT` e che sono stati preprocessati dal meta-object compiler $_G$.

Storicamente era possibile connettere ai Signal_G solo degli Slot, anch'essi parte delle meta-informazioni. Con Qt $_G$ 5 questa limitazione è stata rimossa ed è possibile connettere ad un Signal_G funzioni globali, metodi e funtori $_G$ generici.

Quando un Signal_G viene emesso, la coda di funtori $_G$ connessi è eseguita in ordine passando eventuali argomenti forniti in fase di emissione.

2.2.1.2 Vantaggi

I Signal_G permettono di far comunicare un oggetto con un numero illimitato di ascoltatori senza che questo aumenti le dipendenze per la classe istanziata da questo oggetto. Il meccanismo di Signal_G è parte integrante del framework $_G$ Qt $_G$ ed è usato dalla maggior parte delle classi fornite. Il suo utilizzo è quindi indispensabile per integrare Qt $_G$ all'applicativo che lo usa.

2.2.2 Qt Plugins e Qt3D

Qt $_G$ fornisce un sistema di plugin che consente di estendere Qt $_G$. Tra le possibili estensioni sono presenti formati d'immagini, driver $_G$ per i database o stili per le interfacce grafiche. Il sistema ha due API:

- **API a basso livello:** utilizzata per consentire a librerie esterne di aggiungere tipologie estensibili;
- **API ad alto livello:** utilizzata per aggiungere plugin.

2.2.2.1 API a basso livello

L'applicazione utilizza `QPluginLoader` per caricare i plugin e fornisce una classe astratta che verrà utilizzata dall'API ad alto livello.

2.2.2.2 API ad alto livello

Creare un nuovo plugin consiste nel creare una classe derivata dalla classe astratta del plugin e reimplementare i metodi virtuali richiesti.

2.2.3 Qt3D

Qt3D è una libreria basata su Qt $_G$ che permette di avere una semplice interfaccia a funzionalità di visualizzazione e manipolazione di scene 3D.

Qt3D incapsula operazioni OpenGL $_G$ permettendo un approccio ad alto livello alle operazioni sulla scena $_G$ 3D.

Qt3D implementa le API a basso livello del sistema di plugin di Qt $_G$ per permettere di estendere facilmente il supporto ai vari formati di file. Sono presenti nativamente plugin per aprire file nei formati 3D più diffusi.

Nell'applicazione 3DMob le API ad alto livello del sistema di plugin di Qt $_G$ verranno



utilizzate per aggiungere il supporto ai formati di importazioni richiesti non forniti dalla libreria.

Si prevede che verrà aggiunto il supporto a:

- **JSON_G**: File in formato JSON_G esportato dall'applicazione;
- **XML_G**: File in formato XML_G esportato dall'applicazione;
- **UBJSON**: File in formato UBJSON esportato dall'applicazione.

2.2.4 QMake

Qmake è il sistema di build_G di Qt_G. Automatizza la compilazione generando un Makefile in base alle informazioni presenti sul file di progetto datogli in input. Il makefile generato da qmake si occupa dei seguenti passi:

- **User Interface Compiler**: genera codice a partire dai file del designer di interfacce grafiche;
- **Resource Compiler**: genera codice a partire dai file di risorse, per poter includere file di dati nell'eseguibile;
- **Meta-object Compiler_G**: genera il codice necessario per utilizzare le estensioni fornite da QObject;
- **Compilatore C++** genera codice oggetto a partire dai file sorgente e da quelli generati ai passi precedenti;
- **Linker** unisce tutti i file oggetto e inserisce informazioni riguardanti le librerie utilizzate producendo un file eseguibile.

2.2.4.1 Vantaggi

- il Makefile viene generato in base al sistema richiesto;
- automatizza tutti i processi necessari ad usare le estensioni di Qt_G;
- è integrato in Qt_G Creator, permettendo di avere un ambiente di sviluppo rapido.

2.2.4.2 Svantaggi

- alcuni dei target del Makefile generato non sono ottimali;
- è difficile da integrare con altri sistemi di build_G.



3 Descrizione architettura

3.1 Metodo e formalismo di specifica

Nell'esposizione dell'architettura dell'applicazione si procederà con un approccio top-down_G, descrivendo l'architettura iniziando dal generale ed andando al particolare.

Si procederà quindi alla descrizione dei package_G e dei componenti, per poi descrivere nel dettaglio le singole classi, specificando per ognuna il tipo, l'obiettivo, la funzione e le relazioni in ingresso ed in uscita.

Successivamente si illustreranno degli esempi di uso dei Design Pattern_G nell'architettura del sistema, rimandando la spiegazione generale degli stessi all'appendice A.1.

Per i diagrammi di package_G, di classe e di attività si utilizza il formalismo UML_G 2.x. Nel riportare i diagrammi di package_G e di classe si farà uso, dove appropriato, dei colori, per aiutare la distinzione tra componenti diversi. Si noti in particolare che le classi di colore verde appartengono a Qt_G e sono quindi da considerarsi fuori dai package_G e componenti definiti nell'architettura, anche se riportate all'interno in alcuni diagrammi per maggior chiarezza.

Nel trattare i componenti, si chiarisce che sono da intendersi come package_G e i due termini verranno quindi usati come sinonimi.

Le classi astratte potrebbero non contenere metodi astratti in quanto questi verranno specificati durante la **Progettazione di Dettaglio**.

3.2 Architettura generale

L'architettura del software segue il Design Pattern MVC ed è quindi suddivisa in: Model, View e Controller.

Il diagramma seguente presenta l'architettura ad alto livello dell'applicazione indicandone i package_G e le relazioni tra questi. Un visione ancor più ad alto livello si trova nel componente .

Nel diagramma 1 sono presentate le relazioni tra i package_G Model, View e Controller evidenziano il Design Pattern_G MVC.

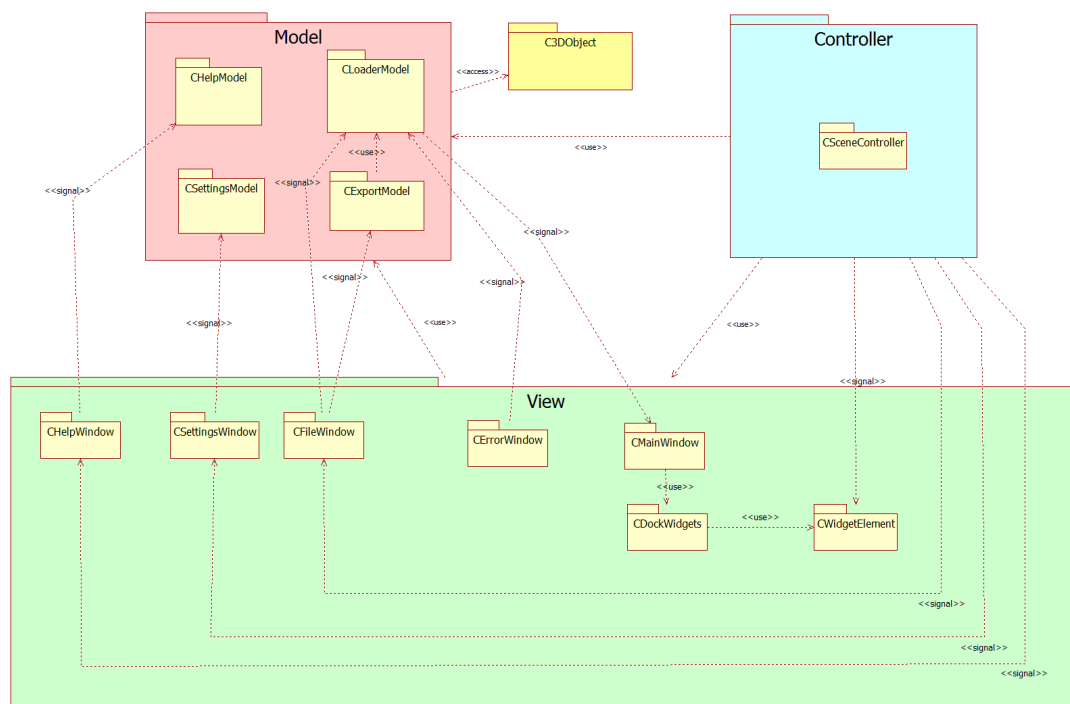


Figura 1: Architettura generale dell'applicazione - vista package

Nel diagramma 2 presenta l'architettura ad alto livello dell'applicazione e vengono indicate le classi fondamentali per rappresentare le relazioni del pattern MVC.

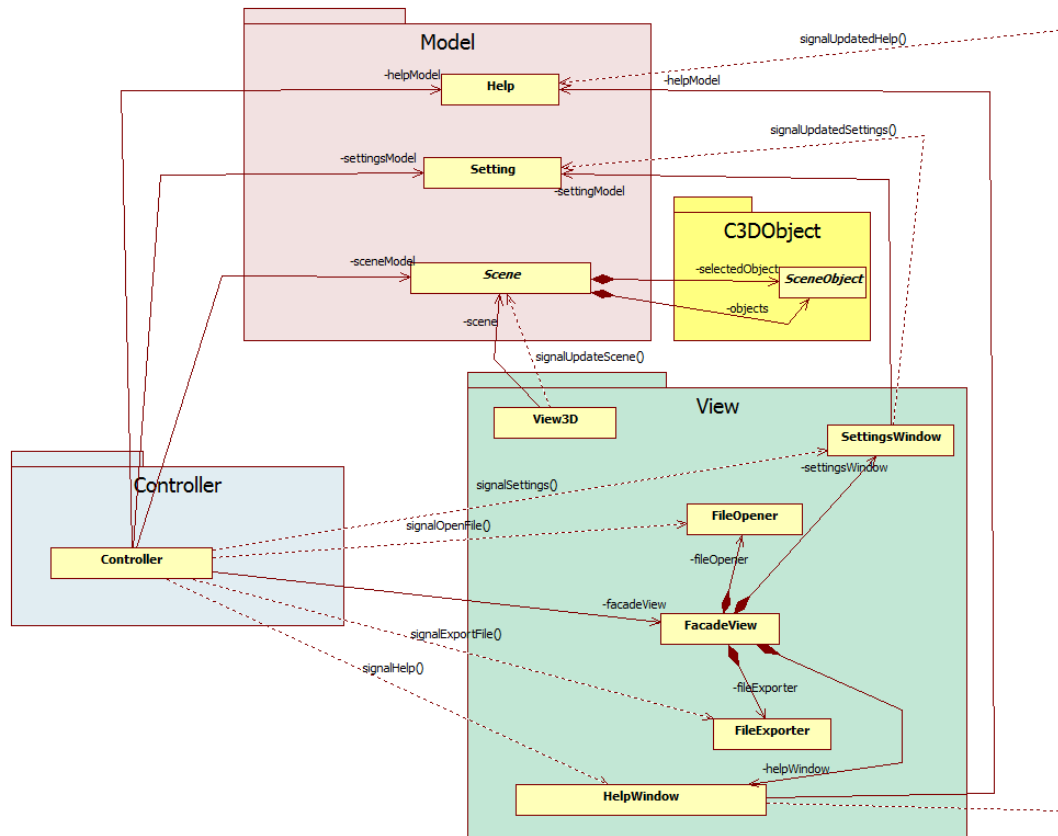


Figura 2: Architettura generale dell'applicazione



3.2.1 Model

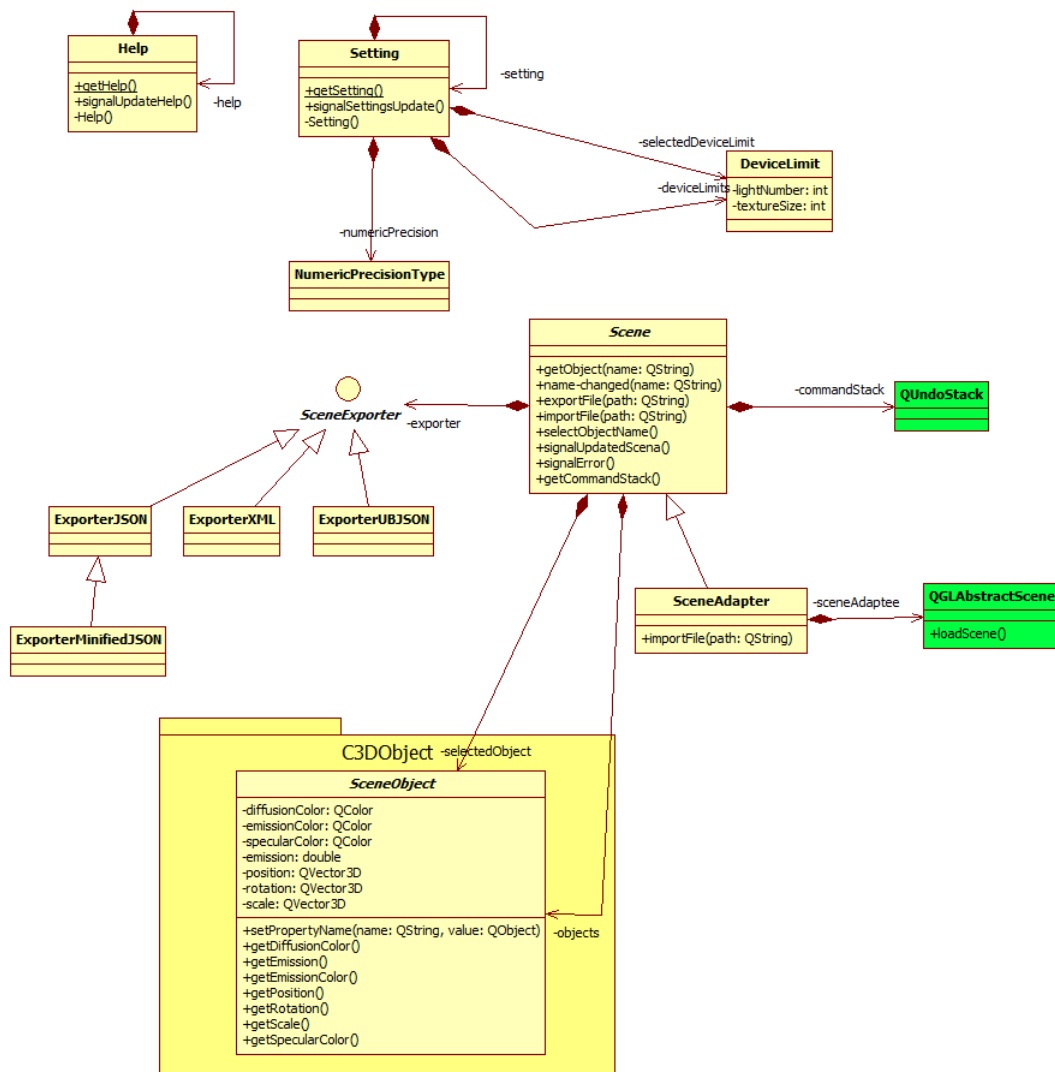


Figura 3: Diagramma delle classi del Model

Nel Model sono presenti gli oggetti che rappresentano:

- La scena_G 3D caricata in memoria;
- La lista dei comandi eseguiti sulla scena_G;
- I dati sulle impostazioni del programma;
- I contenuti del sistema di aiuto.

Mette inoltre a disposizione tutti i metodi per modificare gli oggetti in esso contenuti. Manda segnali alla View quando vi è un aggiornamento dei dati in esso contenuti. In caso si verifichino errori, questi vengono gestiti dal Model e segnalati alla View.

3.2.2 View

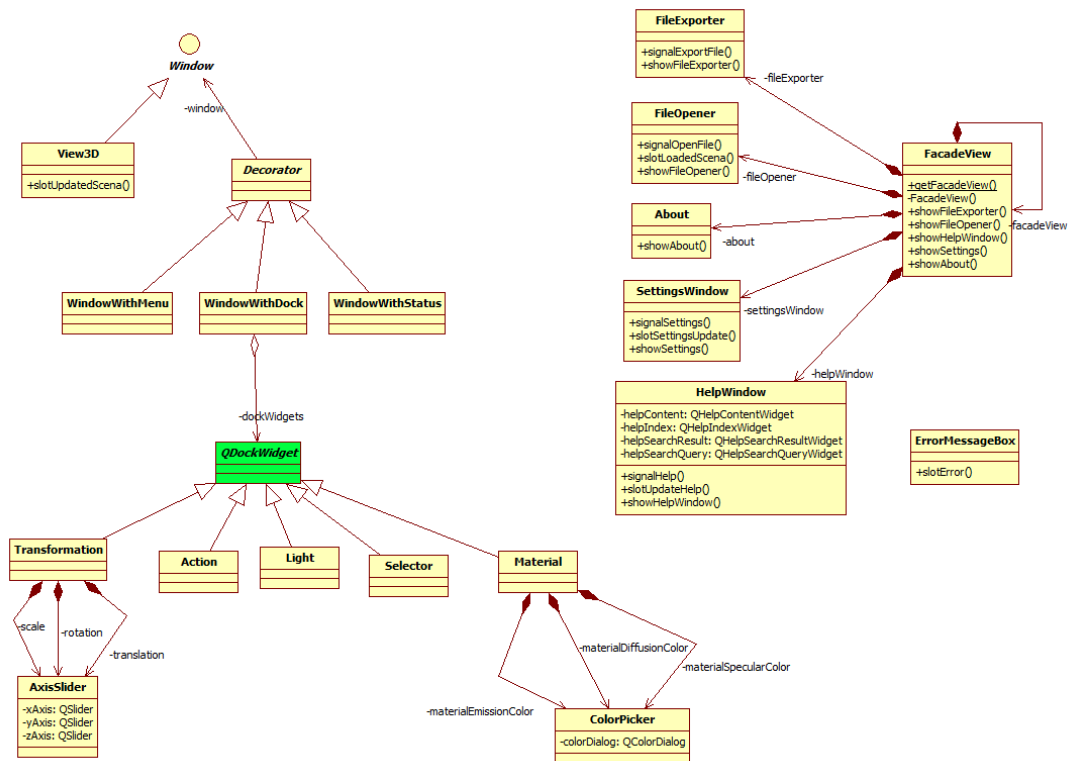


Figura 4: Diagramma delle classi del View

Nella View sono presenti le finestre che permettono all'utente di interagire con il programma.

Tutti i componenti il cui scopo è modificare dati nel Model mandano segnali al Controller. Sarà questo ad invocare i metodi del Model per effettuare l'aggiornamento o la richiesta dei dati.

I componenti delle View che devono essere aggiornati dopo la modifica ricevono un segnale dal Model e da questo vanno a prendere i dati aggiornati.



3.2.3 Controller

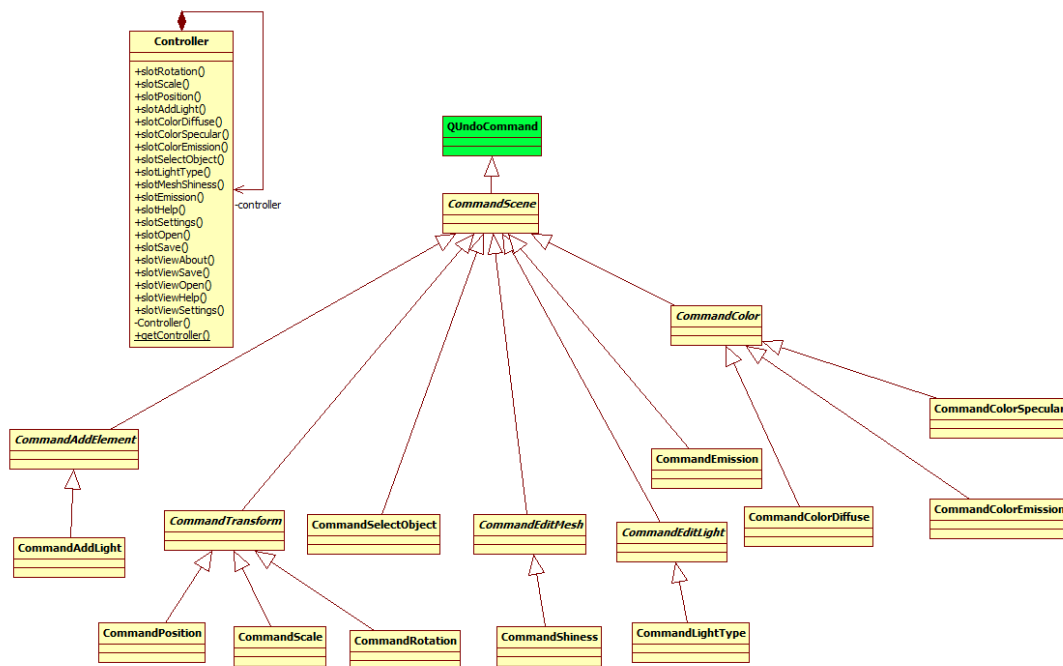


Figura 5: Diagramma delle classi del Controller

Nel Controller è presente la classe che riceve tutti i Signal_C emessi dalla View e che decide che azioni intraprendere:

- **View:** il Controller può agire sulla View e aprire nuove finestre;
- **Model:** il Controller può creare un nuovo comando e inviarlo al Model.



4 Componenti

4.1 DDDMob

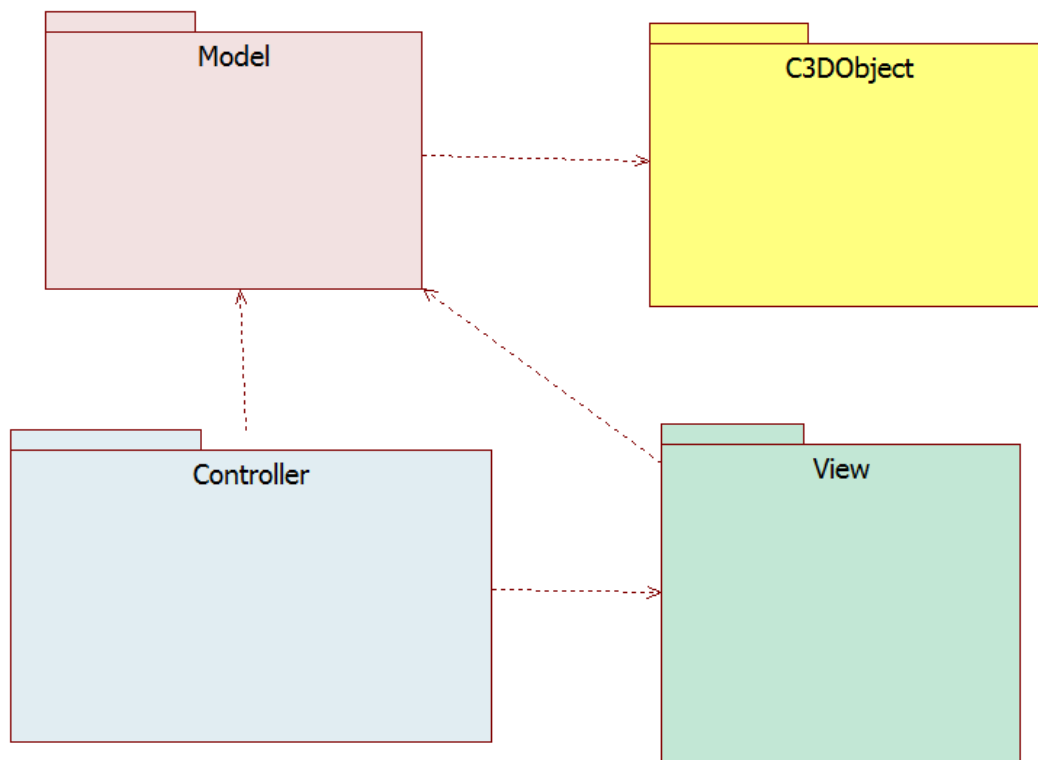


Figura 6: Componente DDDMob

4.1.0.1 Descrizione

Namespace_G globale per il progetto. Le relazioni tra i package_G Model, View e Controller identificano il Design Pattern_G MVC.

4.1.0.2 Figli

- DDDMob::Model;
- DDDMob::Controller;
- DDDMob::View;
- DDDMob::C3DObject.



4.2 DDDMob::Model

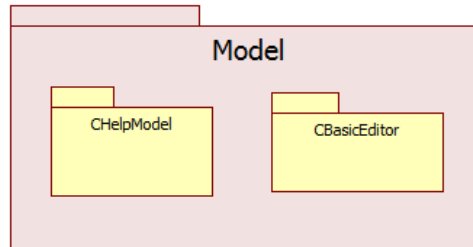


Figura 7: Componente DDDMob::Model

4.2.0.3 Descrizione

Package_G per il componente Model dell'architettura MVC.

4.2.0.4 Figli

- DDDMob::Model::CHelpModel;
- DDDMob::Model::CBasicEditor.

4.3 DDDMob::Model::CHelpModel

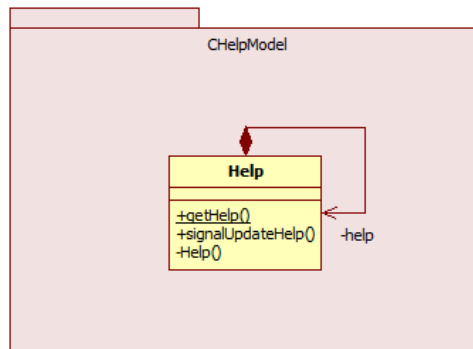


Figura 8: Componente DDDMob::Model::CHelpModel

4.3.0.5 Descrizione

Componente parte del Model per le funzionalità del sistema di aiuto.

4.3.0.6 Classi

- DDDMob::Model::CHelpModel::Help.



4.3.0.7 Interazioni con altri componenti

- DDDMob::Controller;
- DDDMob::View::CHelpWindow;
- Qt_G.

4.4 DDDMob::Model::CBasicEditor

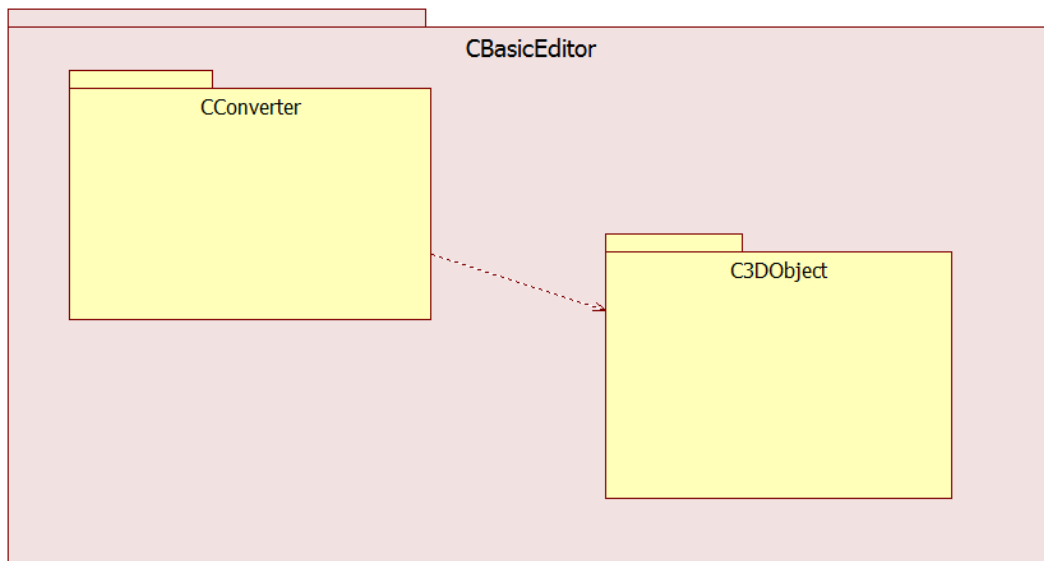


Figura 9: Componente DDDMob::Model::CBasicEditor

4.4.0.8 Descrizione

Macro componente virtuale per l'integrazione delle funzionalità di conversione e modifica dell'oggetto a formare l'editor di base. È definito solamente come parte della strategia di integrazione incrementale, per definire uno stato progressivo del sistema in cui sono offerte le funzionalità di conversione e modifica.

4.4.0.9 Figli

- DDDMob::Model::CBasicEditor::CConverter.



4.5 DDDMob::Model::CBasicEditor::CConverter

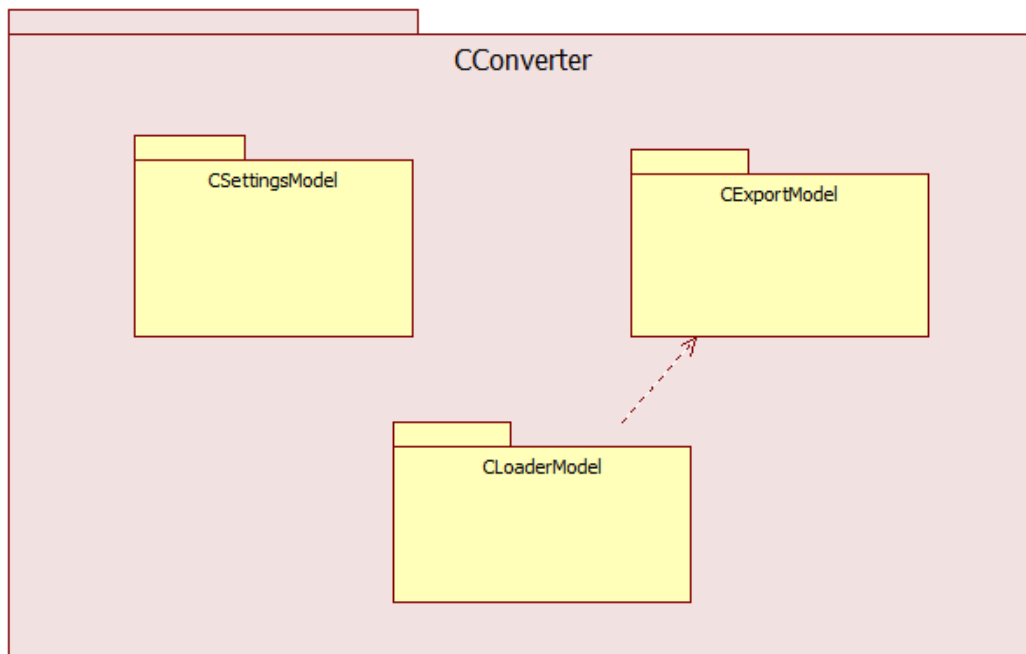


Figura 10: Componente DDDMob::Model::CBasicEditor::CConverter

4.5.0.10 Descrizione

Macro componente per l'integrazione delle funzionalità di importazione, applicazione dei limiti ed esportazione a formare il convertitore di base.

4.5.0.11 Figli

- DDDMob::Model::CBasicEditor::CConverter::CSettingsModel;
- DDDMob::Model::CBasicEditor::CConverter::CExportModel;
- DDDMob::Model::CBasicEditor::CConverter::CLoaderModel.



4.6 DDDMob::Model::CBasicEditor::CConverter::CSettingsModel

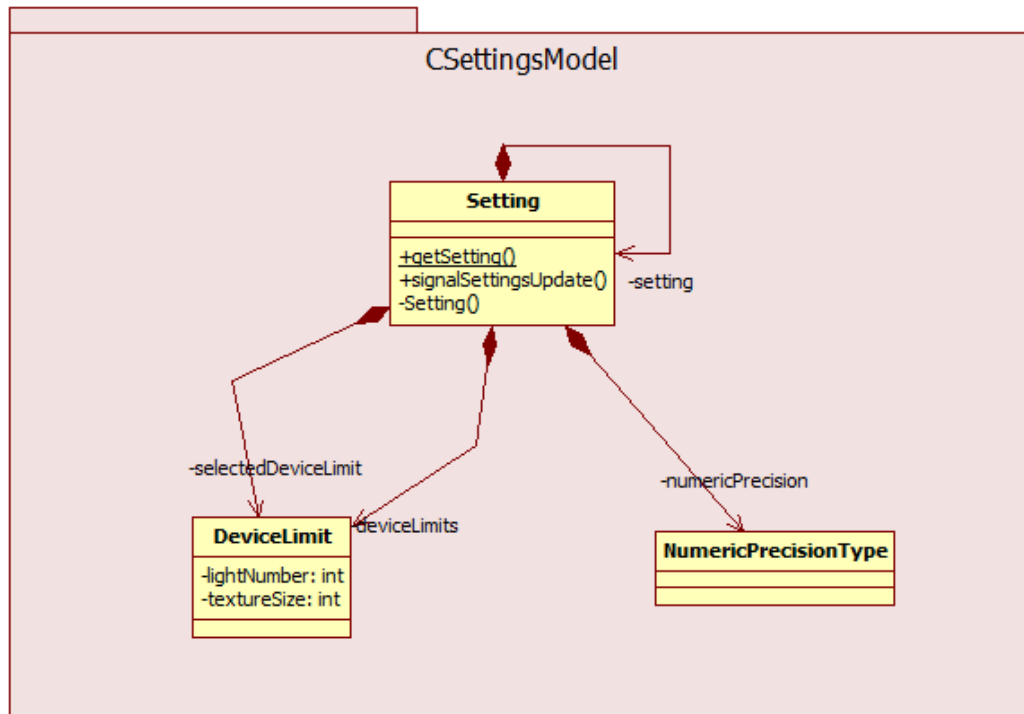


Figura 11: Componente DDDMob::Model::CBasicEditor::CConverter::CSettingsModel

4.6.0.12 Descrizione

Componente parte del Model per le funzionalità di impostazioni e configurazione.

4.6.0.13 Classi

- DDDMob::Model::CBasicEditor::CConverter::CSettingsModel::Setting;
- DDDMob::Model::CBasicEditor::CConverter::CSettingsModel::DeviceLimit;
- DDDMob::Model::CBasicEditor::CConverter::CSettingsModel::NumericPrecisionType.

4.6.0.14 Interazioni con altri componenti

- DDDMob::Controller;
- DDDMob::View::CSettingsWindow.



4.7 DDDMob::Model::CBasicEditor::CConverter::CExportModel

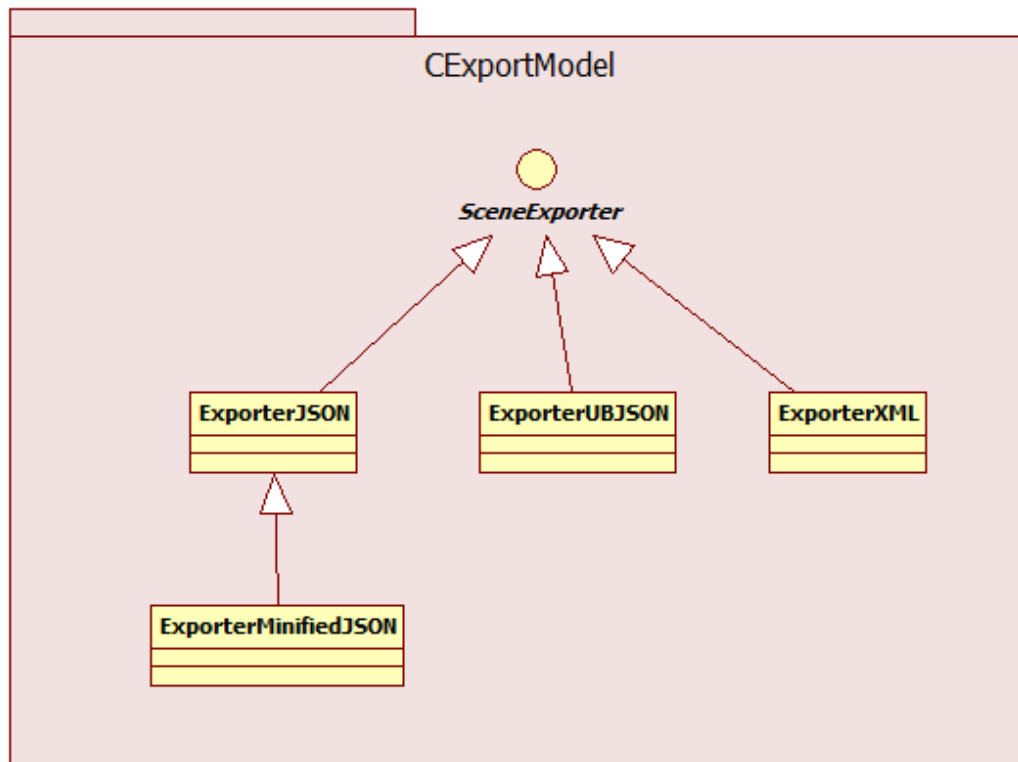


Figura 12: Componente DDDMob::Model::CBasicEditor::CConverter::CExportModel

4.7.0.15 Descrizione

Componente parte del Model per le funzionalità di esportazione.

4.7.0.16 Classi

- DDDMob::Model::CBasicEditor::CConverter::CExportModel::SceneExporter;
- DDDMob::Model::CBasicEditor::CConverter::CExportModel::Exporter.JSON;
- DDDMob::Model::CBasicEditor::CConverter::CExportModel::ExporterUBJSON;
- DDDMob::Model::CBasicEditor::CConverter::CExportModel::ExporterXML;
- DDDMob::Model::CBasicEditor::CConverter::CExportModel::ExporterMinifiedJSON.

4.7.0.17 Interazioni con altri componenti

- DDDMob::Model::CBasicEditor::CConverter::CLoaderModel.



4.8 DDDMob::Model::CBasicEditor::CConverter::CLoaderModel

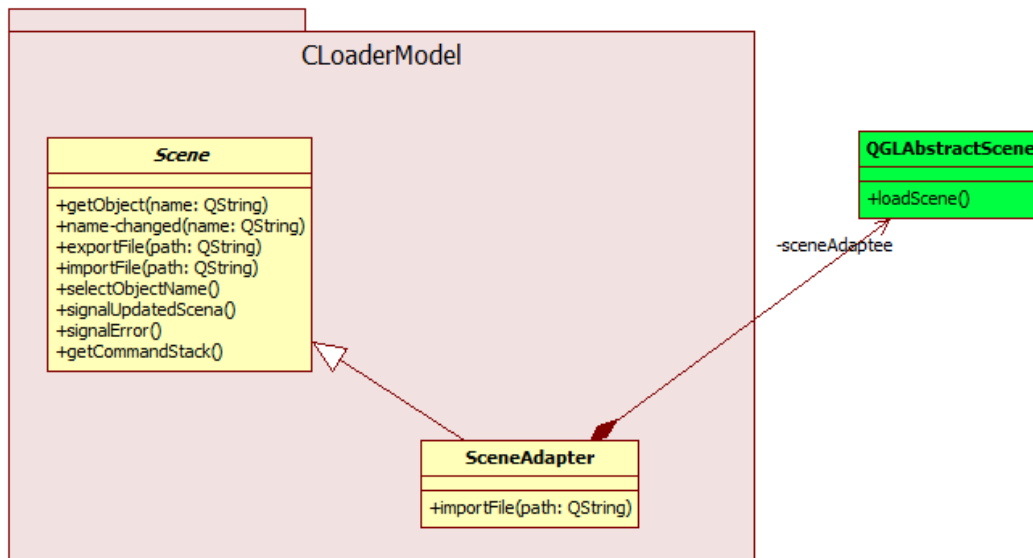


Figura 13: Componente DDDMob::Model::CBasicEditor::CConverter::CLoaderModel

4.8.0.18 Descrizione

Componente parte del Model per le funzionalità di caricamento del file nella scena_G.

4.8.0.19 Classi

- DDDMob::Model::CBasicEditor::CConverter::CLoaderModel::Scene;
- DDDMob::Model::CBasicEditor::CConverter::CLoaderModel::SceneAdapter.

4.8.0.20 Interazioni con altri componenti

- DDDMob::Controller;
- DDDMob::Model::CBasicEditor::CConverter::CExportModel;
- DDDMob::C3DObject;
- Qt_G;
- DDDMob::Controller::CSceneController;
- DDDMob::View::CMainWindow.



4.9 DDDMob::Controller

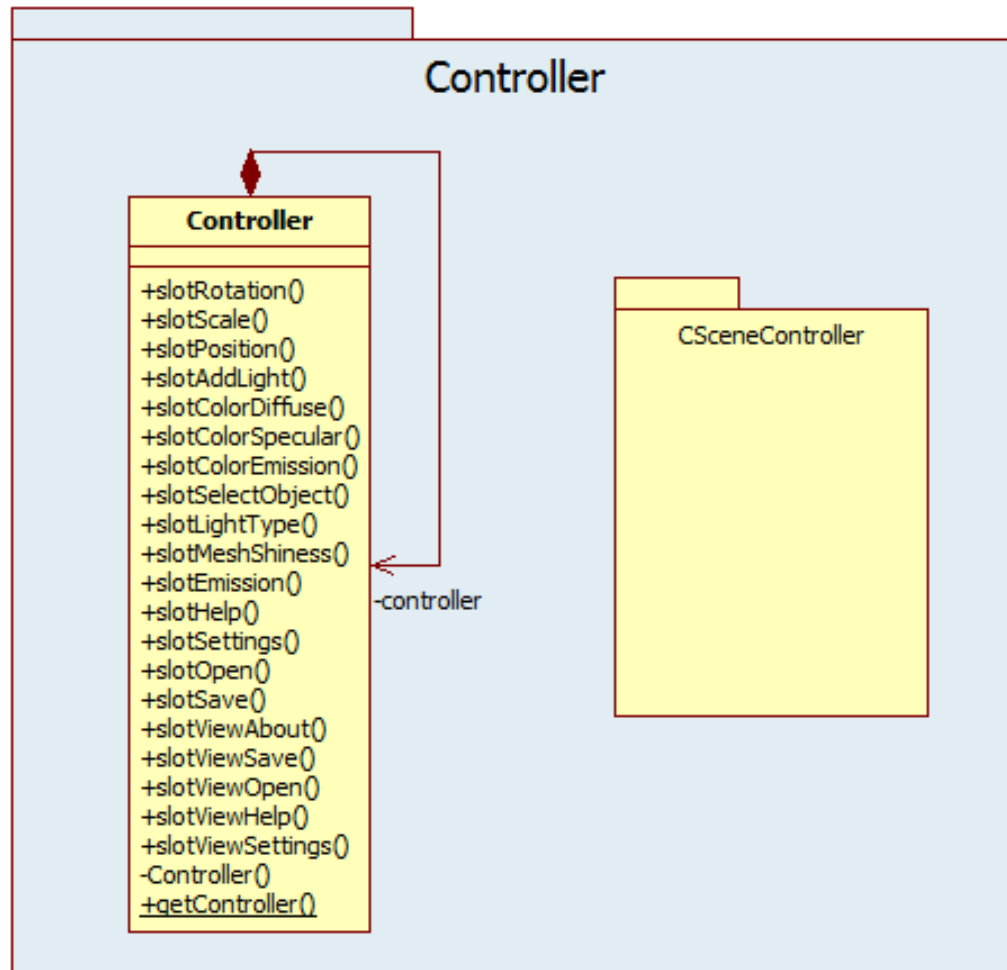


Figura 14: Componente DDDMob::Controller

4.9.0.21 Descrizione

Package_G per il componente Controller dell'architettura MVC.

4.9.0.22 Classi

- DDDMob::Controller::Controller.

4.9.0.23 Figli

- DDDMob::Controller::CSceneController.

4.9.0.24 Interazioni con altri componenti

- DDDMob::View;



- DDDMob::Model::CBasicEditor::CConverter::CLoaderModel;
- DDDMob::Model::CBasicEditor::CConverter::CSettingsModel;
- DDDMob::Model::CHelpModel.

4.10 DDDMob::Controller::CSceneController

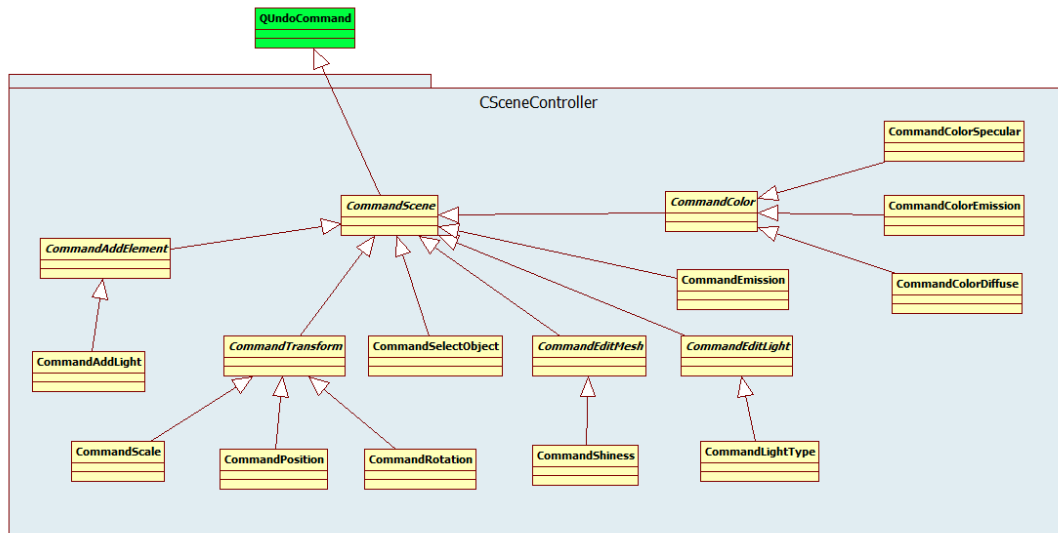


Figura 15: Componente DDDMob::Controller::CSceneController

4.10.0.25 Descrizione

Componente parte del Controller per la gestione dei comandi.

4.10.0.26 Classi

- DDDMob::Controller::CSceneController::CommandTransform;
- DDDMob::Controller::CSceneController::CommandAddLight;
- DDDMob::Controller::CSceneController::CommandSelectObject;
- DDDMob::Controller::CSceneController::CommandEditMesh;
- DDDMob::Controller::CSceneController::CommandEditLight;
- DDDMob::Controller::CSceneController::CommandScene;
- DDDMob::Controller::CSceneController::CommandAddElement;
- DDDMob::Controller::CSceneController::CommandPosition;
- DDDMob::Controller::CSceneController::CommandScale;
- DDDMob::Controller::CSceneController::CommandRotation;



- DDDMob::Controller::CSceneController::CommandLightType;
- DDDMob::Controller::CSceneController::CommandShiness;
- DDDMob::Controller::CSceneController::CommandEmission;
- DDDMob::Controller::CSceneController::CommandColor;
- DDDMob::Controller::CSceneController::CommandColorEmission;
- DDDMob::Controller::CSceneController::CommandColorSpecular;
- DDDMob::Controller::CSceneController::CommandColorDiffuse.

4.10.0.27 Interazioni con altri componenti

- DDDMob::Model::CBasicEditor::CConverter::CLoaderModel;
- Qt_G .

4.11 DDDMob::View

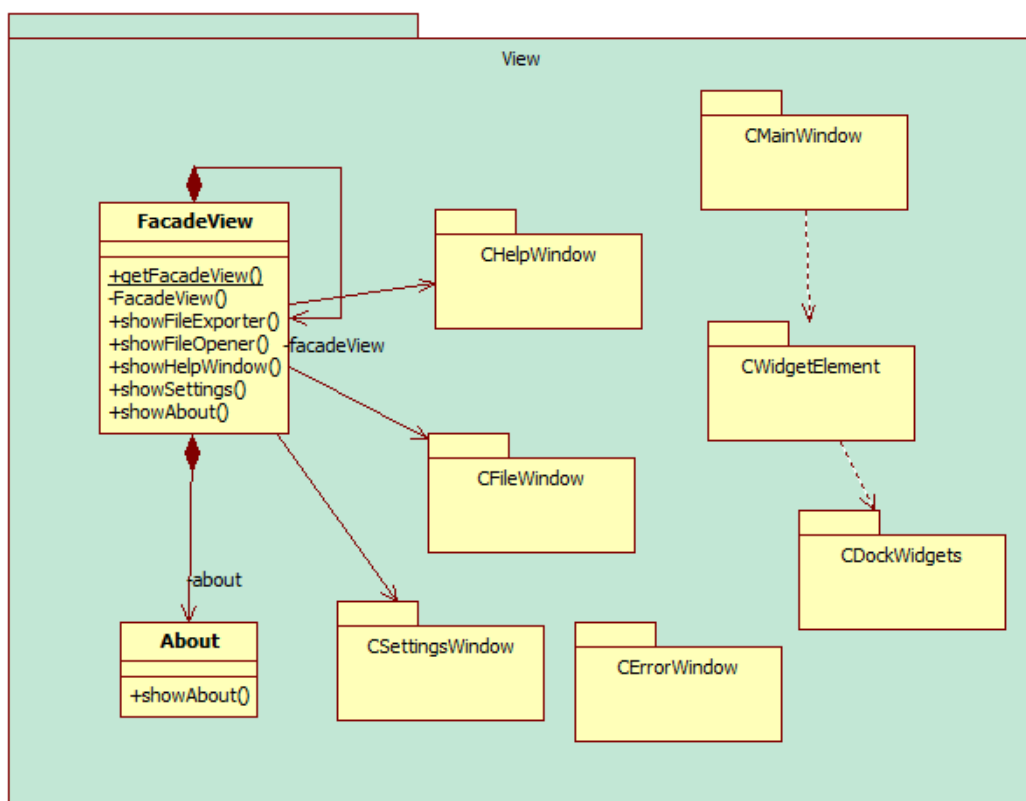


Figura 16: Componente DDDMob::View

4.11.0.28 Descrizione

Package_G per il componente View dell'architettura MVC.



4.11.0.29 Classi

- DDDMob::View::FacadeView;
- DDDMob::View::About.

4.11.0.30 Figli

- DDDMob::View::CDockWidgets;
- DDDMob::View::CMainWindow;
- DDDMob::View::CHelpWindow;
- DDDMob::View::CSettingsWindow;
- DDDMob::View::CErrorWindow;
- DDDMob::View::CFileWindow;
- DDDMob::View::CWidgetElement.

4.11.0.31 Interazioni con altri componenti

- DDDMob::Controller;
- Qt_G.

4.12 DDDMob::View::CDockWidgets

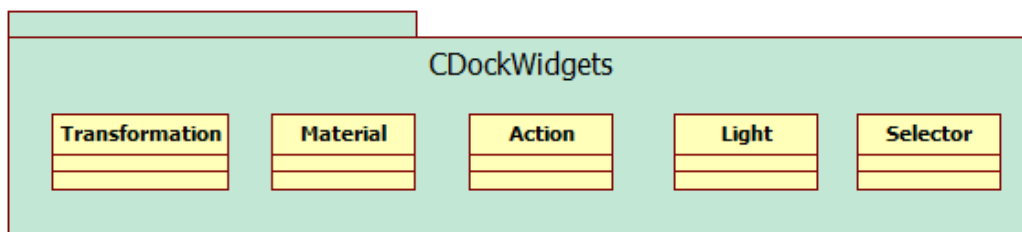


Figura 17: Componente DDDMob::View::CDockWidgets

4.12.0.32 Descrizione

Componente che contiene tutti i tipi di dockwidget disponibili nel sistema.

4.12.0.33 Classi

- DDDMob::View::CDockWidgets::Material;
- DDDMob::View::CDockWidgets::Selector;
- DDDMob::View::CDockWidgets::Light;
- DDDMob::View::CDockWidgets::Action;
- DDDMob::View::CDockWidgets::Transformation.



4.12.0.34 Interazioni con altri componenti

- Qt_G ;
- `DDDMob::View::CWidgetElement`.

4.13 DDDMob::View::CMainWindow

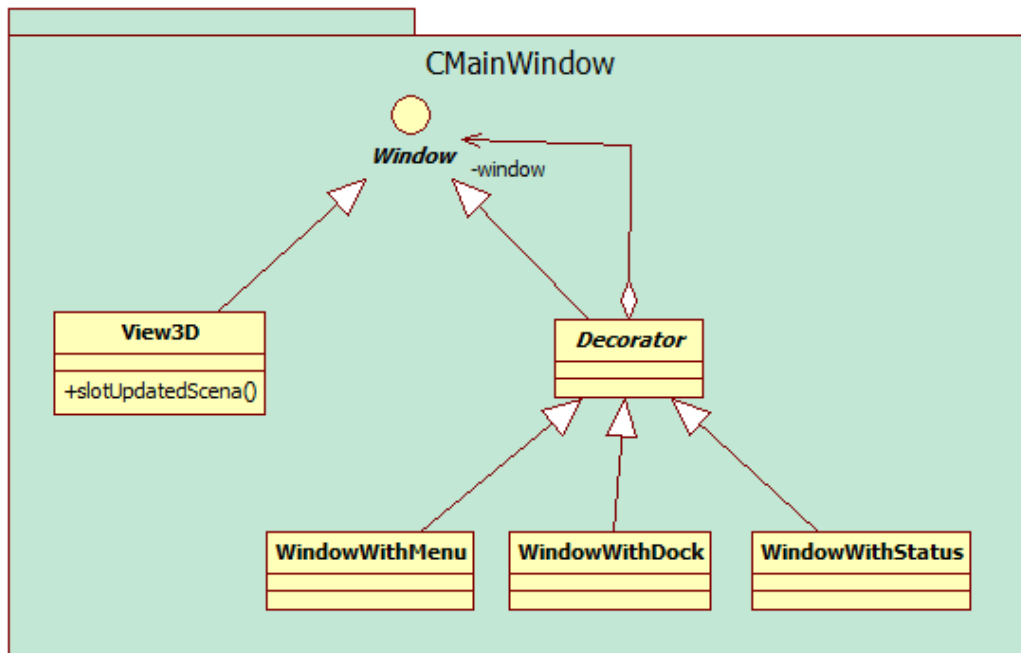


Figura 18: Componente DDDMob::View::CMainWindow

4.13.0.35 Descrizione

Componente parte di View per la finestra principale dell'applicazione.

4.13.0.36 Classi

- `DDDMob::View::CMainWindow::WindowWithMenu`;
- `DDDMob::View::CMainWindow::WindowWithDock`;
- `DDDMob::View::CMainWindow::WindowWithStatus`;
- `DDDMob::View::CMainWindow::Decorator`;
- `DDDMob::View::CMainWindow::Window`;
- `DDDMob::View::CMainWindow::View3D`.



4.13.0.37 Interazioni con altri componenti

- Qt_G ;
- DDDMob::Model::CBasicEditor::CConverter::CLoaderModel.

4.14 DDDMob::View::CHelpWindow

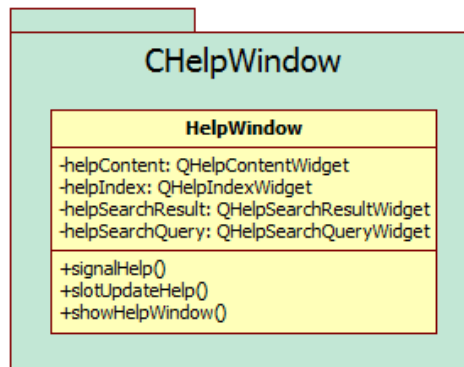


Figura 19: Componente DDDMob::View::CHelpWindow

4.14.0.38 Descrizione

Componente parte di View per la finestra di aiuto per l'utente.

4.14.0.39 Classi

- DDDMob::View::CHelpWindow::HelpWindow.

4.14.0.40 Interazioni con altri componenti

- DDDMob::View;
- DDDMob::Model::CHelpModel;
- Qt_G .



4.15 DDDMob::View::CSettingsWindow

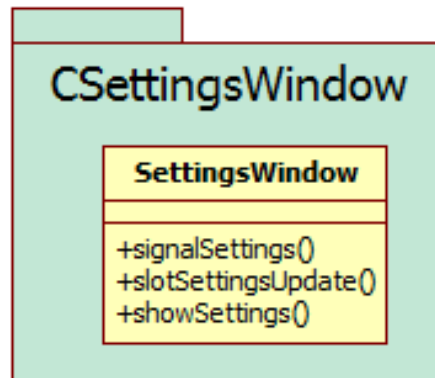


Figura 20: Componente DDDMob::View::CSettingsWindow

4.15.0.41 Descrizione

Componente parte di View per la finestra di configurazione delle impostazioni del programma.

4.15.0.42 Classi

- DDDMob::View::CSettingsWindow::SettingsWindow.

4.15.0.43 Interazioni con altri componenti

- DDDMob::View;
- DDDMob::Model::CBasicEditor::CConverter::CSettingsModel;
- Qt_G .



4.16 DDDMob::View::CErrorWindow

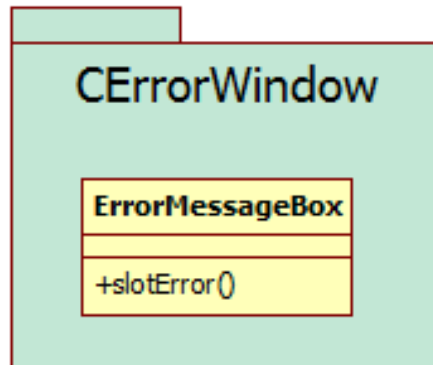


Figura 21: Componente DDDMob::View::CErrorWindow

4.16.0.44 Descrizione

Componente parte di View per la finestra di segnalazione degli errori.

4.16.0.45 Classi

- DDDMob::View::CErrorWindow::ErrorMessageBox.

4.17 DDDMob::View::CFileWindow

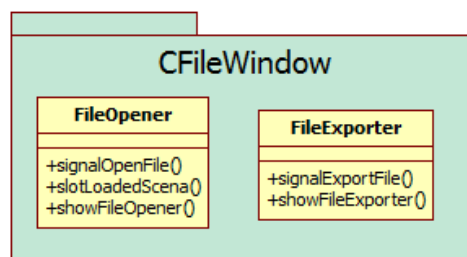


Figura 22: Componente DDDMob::View::CFileWindow

4.17.0.46 Descrizione

Componente parte di View per la finestra di gestione dei file per l'importazione ed esportazione.

4.17.0.47 Classi

- DDDMob::View::CFileWindow::FileOpener;
- DDDMob::View::CFileWindow::FileExporter.



4.17.0.48 Interazioni con altri componenti

- DDDMob::View;
- Qt_G.

4.18 DDDMob::View::CWidgetElement

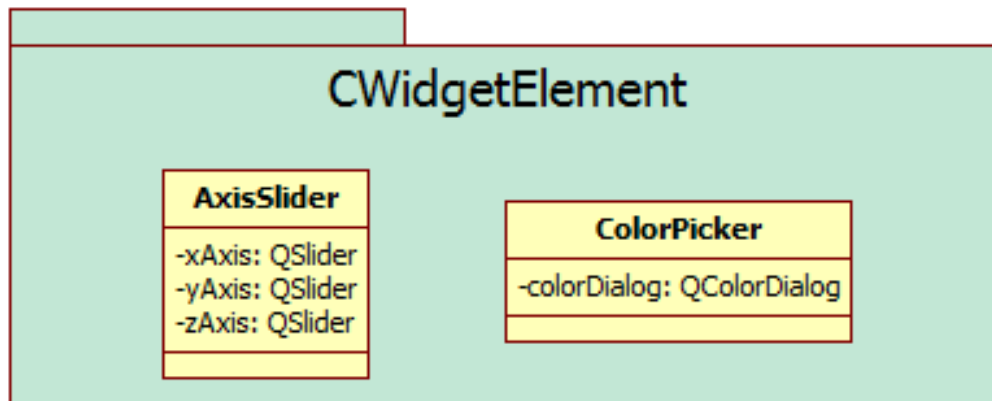


Figura 23: Componente DDDMob::View::CWidgetElement

4.18.0.49 Descrizione

Componente per elementi che sono utilizzati dai widget di View.

4.18.0.50 Classi

- DDDMob::View::CWidgetElement::AxisSlider;
- DDDMob::View::CWidgetElement::ColorPicker.

4.18.0.51 Interazioni con altri componenti

- DDDMob::View::CDockWidgets;
- Qt_G.



4.19 DDDMob::C3DObject

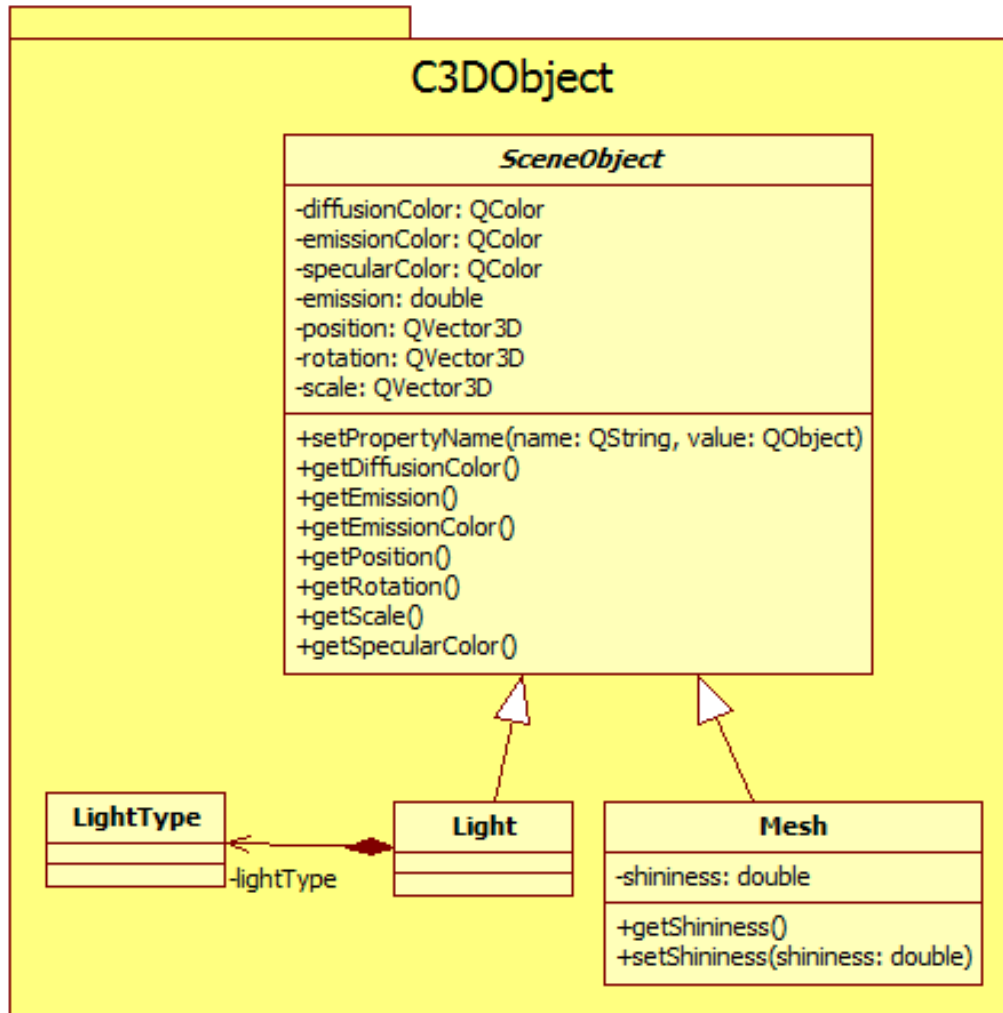


Figura 24: Componente DDDMob::C3DObject

4.19.0.52 Descrizione

Componente condiviso tra Modello e Controller che contiene la definizione dell'oggetto e mette a disposizione le operazioni di modifica.

4.19.0.53 Classi

- DDDMob::C3DObject::SceneObject;
- DDDMob::C3DObject::Light;
- DDDMob::C3DObject::Mesh;
- DDDMob::C3DObject::LightType.



4.19.0.54 Interazioni con altri componenti

- DDDMob::Model::CBasicEditor::CConverter::CLoaderModel.



5 Classi

5.1 Model

5.1.1 DDDMob :: Model :: CHelpModel :: Help

Descrizione

Classe implementata con il Design Pattern_G Singleton che contiene tutte le informazioni del sistema di aiuto e ne permette la ricerca;

Utilizzo

Viene utilizzato un file QHC con tutte le informazioni del sistema d'aiuto. Restituirà i dati del sistema di aiuto richiesti, oppure tutti i dati presenti nel sistema di aiuto. Manda una notifica alla vista per segnalare l'aggiornamento dei dati e la presenza dei risultati richiesti;

Classi ereditate

- QHelpContentModel.

Relazioni con altre classi

DDDMob::View::CHelpWindow::HelpWindow

Relazione entrante, riferimento ai dati del sistema di aiuto del modello;

DDDMob::Controller::Controller

Relazione entrante, riferimento ai dati del sistema di aiuto del modello.

5.1.2 DDDMob :: Model :: CBasicEditor :: CConverter :: CSettingsModel :: Setting

Descrizione

Classe implementata con il Design Pattern_G Singleton che contiene tutte le impostazioni del programma;

Utilizzo

Viene utilizzata per modificare e recuperare i limiti di importazione, precisione di esportazione e colore di sfondo. Riceve richieste di modifica di alcuni elementi o di restituzione dei dati, e notifica alla vista quando gli stessi sono disponibili o aggiornati;

Relazioni con altre classi

...CBasicEditor::CConverter::CSettingsModel::DeviceLimit

Relazione uscente, limiti del dispositivo selezionato;

...CBasicEditor::CConverter::CSettingsModel::DeviceLimit

Relazione uscente, limiti dei dispositivi predefiniti;

...CConverter::CSettingsModel::NumericPrecisionType

Relazione uscente, precisione numerica dell'oggetto esportato;

DDDMob::View::CSettingsWindow::SettingsWindow

Relazione entrante, riferimento alle impostazioni del modello;



DDDMob::Controller::Controller

Relazione entrante, riferimento all'interfaccia per le impostazioni del programma.

5.1.3 DDDMob :: Model :: CBasicEditor :: CConverter :: CSettingsModel :: DeviceLimit

Descrizione

Classe che rappresenta i limiti di un dispositivo;

Utilizzo

Viene utilizzata per salvare i dati che rappresentano i limiti di esportazione relativi a un dispositivo;

Relazioni con altre classi

...Model::CBasicEditor::CConverter::CSettingsModel::Setting

Relazione entrante, limiti del dispositivo selezionato;

...Model::CBasicEditor::CConverter::CSettingsModel::Setting

Relazione entrante, limiti dei dispositivi predefiniti.

5.1.4 DDDMob :: Model :: CBasicEditor :: CConverter :: CSettingsModel :: NumericPrecisionType

Descrizione

Classe che contiene le diverse tipologie di precisione dei numeri;

Utilizzo

Viene utilizzata come `Enumerate`;

Relazioni con altre classi

...Model::CBasicEditor::CConverter::CSettingsModel::Setting

Relazione entrante, precisione numerica dell'oggetto esportato.

5.1.5 DDDMob :: Model :: CBasicEditor :: CConverter :: CExportModel :: SceneExporter

Descrizione

Interfaccia per la componente strategy del Design Pattern_G Strategy per la selezione dell'algoritmo di esportazione della scena_G tra tutti i disponibili;

Utilizzo

Permette di selezionare dinamicamente ed in modo estensibile l'algoritmo di esportazione della scena_G e di conseguenza il formato del file salvato in uscita;

Classi figlie

- DDDMob :: Model :: CBasicEditor :: CConverter :: CExportModel :: ExporterJSON;



- DDDMob :: Model :: CBasicEditor :: CConverter :: CExportModel :: ExporterUBJSON;
- DDDMob :: Model :: CBasicEditor :: CConverter :: CExportModel :: ExporterXML.

Relazioni con altre classi

DDDMob::Model::CBasicEditor::CConverter::CLoaderModel::Scene

Relazione entrante, esportatore della scena_G.

5.1.6 DDDMob :: Model :: CBasicEditor :: CConverter :: CExportModel :: ExporterJSON

Descrizione

Classe che rappresenta un algoritmo di esportazione della scena_G in un file di formato JSON_G. È uno dei componenti concrete component del Design Pattern_G Strategy;

Utilizzo

Viene utilizzata per creare un file JSON_G a partire dalla scena_G 3D;

Classi ereditate

- DDDMob :: Model :: CBasicEditor :: CConverter :: CExportModel :: SceneExporter.

Classi figlie

- DDDMob :: Model :: CBasicEditor :: CConverter :: CExportModel :: ExporterMinifiedJSON.

5.1.7 DDDMob :: Model :: CBasicEditor :: CConverter :: CExportModel :: ExporterUBJSON

Descrizione

Classe che rappresenta un algoritmo di esportazione della scena_G in un file di formato UBJSON. È uno dei componenti concrete component del Design Pattern_G Strategy;

Utilizzo

Viene utilizzata per creare un file UBJSON a partire dalla scena_G 3D;

Classi ereditate

- DDDMob :: Model :: CBasicEditor :: CConverter :: CExportModel :: SceneExporter.



5.1.8 DDDMob :: Model :: CBasicEditor :: CConverter :: CExportModel :: ExporterXML

Descrizione

Classe che rappresenta un algoritmo di esportazione della scena_G in un file di formato XML_G. È uno dei componenti concrete component del Design Pattern_G Strategy;

Utilizzo

Viene utilizzata per creare un file XML_G a partire dalla scena_G 3D;

Classi ereditate

- DDDMob :: Model :: CBasicEditor :: CConverter :: CExportModel :: SceneExporter.

5.1.9 DDDMob :: Model :: CBasicEditor :: CConverter :: CExportModel :: ExporterMinifiedJSON

Descrizione

Classe che rappresenta un algoritmo di esportazione della scena_G in un file di formato JSON_G che contiene JSON_G minificato_G. È uno dei componenti concrete component del Design Pattern_G Strategy;

Utilizzo

Viene utilizzata per creare un file JSON_G che contiene JSON_G minificato_G a partire dalla scena_G 3D;

Classi ereditate

- DDDMob :: Model :: CBasicEditor :: CConverter :: CExportModel :: ExporterJSON.

5.1.10 DDDMob :: Model :: CBasicEditor :: CConverter :: CLoaderModel :: Scene

Descrizione

Classe che rappresenta la scena_G 3D che contiene le mesh e le luci. È il componente receiver del Design Pattern_G Command, componente context del Design Pattern_G Strategy e componente target del Design Pattern_G Adapter. Internamente contiene la lista di comandi eseguiti sulla scena_G 3D;

Utilizzo

Interfaccia per gestire le proprietà degli oggetti 3D. Permette la selezione di un oggetto della scena_G e la sua modifica. Contiene lo stato dei comandi eseguiti sulla scena_G così da poter annullare e ripristinare le modifiche effettuate. Permette l'esportazione in vari formati, selezionati grazie al Design Pattern_G Strategy, e l'importazione nei vari formati permessi dalla libreria esterna utilizzata;

Classi figlie



- DDDMob :: Model :: CBasicEditor :: CConverter :: CLoaderModel :: SceneAdapter.

Relazioni con altre classi

...CBasicEditor::CConverter::CExportModel::SceneExporter

Relazione uscente, esportatore della scena_G;

DDDMob::C3DObject::SceneObject

Relazione uscente, oggetto selezionato;

DDDMob::C3DObject::SceneObject

Relazione uscente, oggetti della scena_G;

DDDMob::Controller::CSceneController::CommandScene

Relazione entrante, riferimento alla scena_G del modello;

DDDMob::Controller::Controller

Relazione entrante, riferimento alla scena_G del modello;

QUndoStack

Relazione uscente, lista delle azioni effettuate;

DDDMob::View::CMainWindow::View3D

Relazione entrante, riferimento alla scena_G del Modello.

5.1.11 DDDMob :: Model :: CBasicEditor :: CConverter :: CLoaderModel :: SceneAdapter

Descrizione

Classe che viene utilizzata come adattatore per la classe QGLAbstractScene della libreria esterna Qt3D. Rappresenta il componente adapter del Design Pattern_G Adapter;

Utilizzo

Viene utilizzata come adattatore tra quanto esposto dalla libreria esterna Qt3D e la classe Scene;

Classi ereditate

- DDDMob :: Model :: CBasicEditor :: CConverter :: CLoaderModel :: Scene.

Relazioni con altre classi

QGLAbstractScene

Relazione uscente, oggetto scena_G fornito da Qt3D.

5.2 Controller

5.2.1 DDDMob :: Controller :: Controller

Descrizione

Classe che rappresenta il componente controller del Design Pattern_G MVC ed il componente client del Design Pattern_G Command;



Utilizzo

Gestisce i Signal_G inviati dalla View ed agisce nel modo corretto. Genera i comandi da eseguire sulla scena_G 3D e memorizza tali comandi nel componente Scene del Model. Concorre nell'applicare le modifiche all'oggetto selezionato della scena_G 3D;

Relazioni con altre classi

DDDMob::View::FacadeView

Relazione uscente, riferimento alla facciata della Vista;

DDDMob::Model::CBasicEditor::CConverter::CLoaderModel::Scene

Relazione uscente, riferimento alla scena_G del modello;

...Model::CBasicEditor::CConverter::CSettingsModel::Setting

Relazione uscente, riferimento all'interfaccia per le impostazioni del programma;

DDDMob::Model::CHelpModel::Help

Relazione uscente, riferimento ai dati del sistema di aiuto del modello.

5.2.2 DDDMob :: Controller :: CSceneController :: CommandTransform

Descrizione

Classe base per i comandi di trasformazione;

Utilizzo

Viene utilizzata per applicare un generico parametro di trasformazione ad un oggetto della scena_G 3D, specificato poi nelle classi che ereditano da questa;

Classi ereditate

- DDDMob :: Controller :: CSceneController :: CommandScene.

Classi figlie

- DDDMob :: Controller :: CSceneController :: CommandPosition;
- DDDMob :: Controller :: CSceneController :: CommandScale;
- DDDMob :: Controller :: CSceneController :: CommandRotation.

5.2.3 DDDMob :: Controller :: CSceneController :: CommandAddLight

Descrizione

Classe che rappresenta il comando di aggiunta di una luce con i parametri di default alla scena_G 3D;

Utilizzo

Viene utilizzata per gestire i Signal_G riguardanti la creazione di una luce ed invocare i corretti metodi del Model;

Classi ereditate

- DDDMob :: Controller :: CSceneController :: CommandAddElement.



5.2.4 DDDMob :: Controller :: CSceneController :: CommandSelectObject

Descrizione

Classe che rappresenta il comando di selezione di un oggetto tra quelli presenti nella scena_G 3D;

Utilizzo

Viene utilizzata per gestire i Signal_G riguardanti la selezione di un oggetto ed invocare i corretti metodi del Model;

Classi ereditate

- DDDMob :: Controller :: CSceneController :: CommandScene.

5.2.5 DDDMob :: Controller :: CSceneController :: CommandEditMesh

Descrizione

Classe che rappresenta il comando di modifica di una mesh;

Utilizzo

Viene utilizzata per gestire i Signal_G riguardanti la modifica di una mesh ed invocare i corretti metodi del Model;

Classi ereditate

- DDDMob :: Controller :: CSceneController :: CommandScene.

Classi figlie

- DDDMob :: Controller :: CSceneController :: CommandShiness.

5.2.6 DDDMob :: Controller :: CSceneController :: CommandEditLight

Descrizione

Classe che rappresenta il comando di modifica della luce selezionata;

Utilizzo

Viene utilizzata per gestire i Signal_G riguardanti la modifica della luce selezionata ed invocare i corretti metodi del Model;

Classi ereditate

- DDDMob :: Controller :: CSceneController :: CommandScene.

Classi figlie

- DDDMob :: Controller :: CSceneController :: CommandLightType.



5.2.7 DDDMob :: Controller :: CSceneController :: CommandScene

Descrizione

Classe che rappresenta un generico comando da eseguire sulla scena_G. È il padre di tutti i possibile componenti concrete command del Design Pattern_G Command;

Utilizzo

Viene utilizzata per applicare un generico comando da eseguire sulla scena_G 3D, specificato poi nelle classi che ereditano da questa;

Classi ereditate

- QUndoCommand.

Classi figlie

- DDDMob :: Controller :: CSceneController :: CommandTransform;
- DDDMob :: Controller :: CSceneController :: CommandSelectObject;
- DDDMob :: Controller :: CSceneController :: CommandEditMesh;
- DDDMob :: Controller :: CSceneController :: CommandEditLight;
- DDDMob :: Controller :: CSceneController :: CommandAddElement;
- DDDMob :: Controller :: CSceneController :: CommandEmission;
- DDDMob :: Controller :: CSceneController :: CommandColor.

Relazioni con altre classi

DDDMob::Model::CBasicEditor::CConverter::CLoaderModel::Scene

Relazione uscente, riferimento alla scena_G del modello.

5.2.8 DDDMob :: Controller :: CSceneController :: CommandAddElement

Descrizione

Classe che rappresenta un comando di aggiunta di un elemento alla scena_G 3D;

Utilizzo

Viene utilizzata per gestire i Signal_G riguardanti l'aggiunta di un elemento ed invocare i corretti metodi del Model;

Classi ereditate

- DDDMob :: Controller :: CSceneController :: CommandScene.

Classi figlie

- DDDMob :: Controller :: CSceneController :: CommandAddLight.



5.2.9 DDDMob :: Controller :: CSceneController :: CommandPosition

Descrizione

Classe che rappresenta un comando per cambiare la posizione dell'oggetto della scena_G 3D selezionato;

Utilizzo

Viene utilizzata per gestire i Signal_G riguardanti la modifica della posizione di un oggetto ed invocare i corretti metodi del Model;

Classi ereditate

- DDDMob :: Controller :: CSceneController :: CommandTransform.

5.2.10 DDDMob :: Controller :: CSceneController :: CommandScale

Descrizione

Classe che rappresenta un comando per cambiare la dimensione dell'oggetto della scena_G 3D selezionato;

Utilizzo

Viene utilizzata per gestire i Signal_G riguardanti la dimensione di un oggetto ed invocare i corretti metodi del Model;

Classi ereditate

- DDDMob :: Controller :: CSceneController :: CommandTransform.

5.2.11 DDDMob :: Controller :: CSceneController :: CommandRotation

Descrizione

Classe che rappresenta un comando per cambiare la rotazione del modello;

Classi ereditate

- DDDMob :: Controller :: CSceneController :: CommandTransform.

5.2.12 DDDMob :: Controller :: CSceneController :: CommandLightType

Descrizione

Classe che rappresenta un comando per cambiare il tipo della luce selezionata;

Utilizzo

Viene utilizzata per gestire i Signal_G riguardanti la modifica della tipologia di una luce ed invocare i corretti metodi del Model;

Classi ereditate

- DDDMob :: Controller :: CSceneController :: CommandEditLight.



5.2.13 DDDMob :: Controller :: CSceneController :: CommandShiness

Descrizione

Classe che rappresenta il comando che cambia il valore di lucentezza della mesh;

Utilizzo

Viene utilizzata per gestire i Signal_G riguardanti la modifica della lucentezza di una mesh ed invocare i corretti metodi del Model;

Classi ereditate

- DDDMob :: Controller :: CSceneController :: CommandEditMesh.

5.2.14 DDDMob :: Controller :: CSceneController :: CommandEmission

Descrizione

Classe che rappresenta il comando per cambiare il valore della quantità di emissione del materiale di un oggetto;

Utilizzo

Viene utilizzata per gestire i Signal_G riguardanti la quantità di emissione del materiale di un oggetto ed invocare i corretti metodi del Model;

Classi ereditate

- DDDMob :: Controller :: CSceneController :: CommandScene.

5.2.15 DDDMob :: Controller :: CSceneController :: CommandColor

Descrizione

Classe che rappresenta il comando per cambiare le caratteristiche legate al colore dell'oggetto;

Utilizzo

Viene utilizzata per gestire i Signal_G riguardanti il colore dell'oggetto;

Classi ereditate

- DDDMob :: Controller :: CSceneController :: CommandScene.

Classi figlie

- DDDMob :: Controller :: CSceneController :: CommandColorEmission;
- DDDMob :: Controller :: CSceneController :: CommandColorSpecular;
- DDDMob :: Controller :: CSceneController :: CommandColorDiffuse.



5.2.16 DDDMob :: Controller :: CSceneController :: CommandColorEmission

Descrizione

Classe che rappresenta il comando per cambiare il colore di emissione dell'oggetto;

Utilizzo

Viene utilizzata per gestire i Signal_G riguardanti il colore di emissione dell'oggetto;

Classi ereditate

- DDDMob :: Controller :: CSceneController :: CommandColor.

5.2.17 DDDMob :: Controller :: CSceneController :: CommandColorSpecular

Descrizione

Classe che rappresenta il comando per cambiare il colore di diffusione_G dell'oggetto;

Utilizzo

Viene utilizzata per gestire i Signal_G riguardanti il colore di diffusione_G dell'oggetto;

Classi ereditate

- DDDMob :: Controller :: CSceneController :: CommandColor.

5.2.18 DDDMob :: Controller :: CSceneController :: CommandColorDiffuse

Descrizione

Comando per cambiare il colore di diffusione_G dell'oggetto;

Utilizzo

Viene utilizzata per gestire i Signal_G riguardanti il colore di diffusione_G dell'oggetto;

Classi ereditate

- DDDMob :: Controller :: CSceneController :: CommandColor.

5.3 View

5.3.1 DDDMob :: View :: FacadeView

Descrizione

Classe che rappresenta la Vista. È la componente facade del Design Pattern_G Facade, ed è implementata con il Design Pattern_G Singleton;



Utilizzo

Viene utilizzata per accedere alla vista in modo protetto, in quanto espone i metodi necessari per l'interazione dall'esterno, nascondendo l'implementazione e la struttura;

Relazioni con altre classi

DDDMob::View::CFileWindow::FileExporter

Relazione uscente, riferimento all'interfaccia di esportazione;

DDDMob::View::CFileWindow::FileOpener

Relazione uscente, riferimento all'interfaccia di importazione;

DDDMob::View::CSettingsWindow::SettingsWindow

Relazione uscente, riferimento all'interfaccia per le impostazioni del programma;

DDDMob::View::CHelpWindow::HelpWindow

Relazione uscente, riferimento all'interfaccia per il sistema di aiuto;

DDDMob::View::About

Relazione uscente, riferimento all'interfaccia con le informazioni sul programma;

DDDMob::Controller::Controller

Relazione entrante, riferimento alla facciata della Vista.

5.3.2 DDDMob :: View :: About

Descrizione

Classe che rappresenta la finestra contenente le informazioni sul programma e sui suoi creatori;

Utilizzo

Viene utilizzata per visualizzare le informazioni sul programma e sui suoi creatori, oltre che la versione del prodotto e delle librerie utilizzate;

Classi ereditate

- QDialog.

Relazioni con altre classi

DDDMob::View::FacadeView

Relazione entrante, riferimento all'interfaccia con le informazioni sul programma.

5.3.3 DDDMob :: View :: CDockWidgets :: Material

Descrizione

Classe che rappresenta il widget contenente le informazioni sul materiale;

Utilizzo

Viene utilizzata per permettere di cambiare le caratteristiche del materiale dell'oggetto. Emette un Signal_G in seguito all'interazione dell'utente con l'elemento dell'interfaccia grafica;



Classi ereditate

- QDockWidget.

Relazioni con altre classi

QSlider

Relazione uscente, elemento grafico che permette di modificare l'intensità del colore del materiale;

DDDMob::View::CWidgetElement::ColorPicker

Relazione uscente, elemento grafico che permette di modificare il colore di diffusione_G del materiale;

DDDMob::View::CWidgetElement::ColorPicker

Relazione uscente, elemento grafico che permette di modificare il colore speculare_G del materiale;

DDDMob::View::CWidgetElement::ColorPicker

Relazione uscente, elemento grafico che permette di modificare il colore di emissione del materiale.

5.3.4 DDDMob :: View :: CDockWidgets :: Selector

Descrizione

Classe che rappresenta il widget contenente il selettore di oggetti;

Utilizzo

Viene utilizzata per selezionare un oggetto della scena_G. Emette un Signal_G in seguito all'interazione dell'utente con l'elemento dell'interfaccia grafica;

Classi ereditate

- QDockWidget.

Relazioni con altre classi

QListWidget

Relazione uscente, elemento grafico che permette di selezionare un oggetto.

5.3.5 DDDMob :: View :: CDockWidgets :: Light

Descrizione

Classe che rappresenta il widget che permette di interagire con le luci della scena_G e di aggiungerne di nuove;

Utilizzo

Viene utilizzata per interagire con le luci della scena_G o aggiungerne una nuova. Emette un Signal_G in seguito all'interazione dell'utente con l'elemento dell'interfaccia grafica;

Classi ereditate

- QDockWidget.

Relazioni con altre classi



QPushButton

Relazione uscente, elemento grafico che permette di aggiungere una luce;

QRadioButton

Relazione uscente, elemento grafico che permette di modificare il tipo di luce.

5.3.6 DDDMob :: View :: CDockWidgets :: Action

Descrizione

Classe che rappresenta il widget che consente di visualizzare le azioni effettuate, di annullarle e ripristinarle;

Utilizzo

Viene utilizzata per visualizzare le azioni effettuate, annullarle e ripristinarle. Emette un Signal_G in seguito all'interazione dell'utente con l'elemento dell'interfaccia grafica;

Classi ereditate

- QDockWidget.

Relazioni con altre classi

QUndoView

Relazione uscente, elemento grafico che permette di visualizzare le operazioni effettuate.

5.3.7 DDDMob :: View :: CDockWidgets :: Transformation

Descrizione

Classe che rappresenta un widget contenente gli slider per modificare alcune caratteristiche dell'oggetto;

Utilizzo

Viene utilizzata per permettere di applicare una trasformazione alla scena $_G$. Emette un Signal_G in seguito all'interazione dell'utente con l'elemento dell'interfaccia grafica;

Classi ereditate

- QDockWidget.

Relazioni con altre classi

DDDMob::View::CWidgetElement::AxisSlider

Relazione uscente, elemento grafico che permette di modificare la rotazione dell'elemento;

DDDMob::View::CWidgetElement::AxisSlider

Relazione uscente, elemento grafico che permette di modificare la dimensione;

DDDMob::View::CWidgetElement::AxisSlider

Relazione uscente, elemento grafico che permette di modificare la traslazione dell'elemento.



5.3.8 DDDMob :: View :: CMainWindow :: WindowWithMenu

Descrizione

Classe che decora la finestra principale. Rappresenta uno dei componenti concrete decorator del Design Pattern_G Decorator;

Utilizzo

Viene utilizzata per aggiungere un menù alla finestra;

Classi ereditate

- QMenuBar;
- DDDMob :: View :: CMainWindow :: Decorator.

Relazioni con altre classi

QMenu

Relazione uscente, menù nella barra dei menù.

5.3.9 DDDMob :: View :: CMainWindow :: WindowWithDock

Descrizione

Classe che decora la finestra principale. Rappresenta uno dei componenti concrete decorator del Design Pattern_G Decorator;

Utilizzo

Decoratore che aggiunge elementi ancorabili alla finestra;

Classi ereditate

- DDDMob :: View :: CMainWindow :: Decorator.

Relazioni con altre classi

QDockWidget

Relazione uscente, rappresenta i DockWidgets presenti nella finestra.

5.3.10 DDDMob :: View :: CMainWindow :: WindowWithStatus

Descrizione

Classe che decora la finestra principale. Rappresenta uno dei componenti concrete decorator del Design Pattern_G Decorator;

Utilizzo

Viene utilizzata per aggiungere ad una finestra la barra di stato;

Classi ereditate

- QStatusBar;
- DDDMob :: View :: CMainWindow :: Decorator.



5.3.11 DDDMob :: View :: CMainWindow :: Decorator

Descrizione

Classe astratta che rappresenta le varie decorazioni applicabili all'interfaccia utente. Rappresenta il componente decorator del Design Pattern_G Decorator;

Utilizzo

Classe che rappresenta le possibili decorazioni applicabili all'interfaccia utente, specificate poi dalle classi che ereditano da questa;

Classi ereditate

- DDDMob :: View :: CMainWindow :: Window.

Classi figlie

- DDDMob :: View :: CMainWindow :: WindowWithMenu;
- DDDMob :: View :: CMainWindow :: WindowWithDock;
- DDDMob :: View :: CMainWindow :: WindowWithStatus.

Relazioni con altre classi

DDDMob::View::CMainWindow::Window

Relazione uscente, oggetto window innestato nel Decorator.

5.3.12 DDDMob :: View :: CMainWindow :: Window

Descrizione

Interfaccia che rappresenta la finestra principale del programma, ottenuta dalla vista 3D decorata con le decorazioni offerte dalla classe Decorator. Rappresenta il componente component del Design Pattern_G Decorator;

Classi figlie

- DDDMob :: View :: CMainWindow :: Decorator;
- DDDMob :: View :: CMainWindow :: View3D.

Relazioni con altre classi

DDDMob::View::CMainWindow::Decorator

Relazione entrante, oggetto window innestato nel Decorator.

5.3.13 DDDMob :: View :: CMainWindow :: View3D

Descrizione

Classe che rappresenta la vista contenente la scena_G 3D. È il componente concrete component del Design Pattern_G Decorator;

Utilizzo

Viene utilizzata per visualizzare l'anteprima della scena_G 3D del Model. Manda Signal_G di richiesta di aggiornamento dei dati di visualizzazione della scena_G 3D al Model. Può leggere i dati aggiornati direttamente dal Model dopo aver ricevuto un Signal_G di aggiornamento;



Classi ereditate

- QMainWindow;
- DDDMob :: View :: CMainWindow :: Window.

Relazioni con altre classi

DDDMob::Model::CBasicEditor::CConverter::CLoaderModel::Scene

Relazione uscente, riferimento alla scena_G del Modello.

5.3.14 DDDMob :: View :: CHelpWindow :: HelpWindow

Descrizione

Classe che rappresenta la finestra utilizzata per mostrare il sistema di aiuto;

Utilizzo

Viene utilizzata per visualizzare il sistema di aiuto. La sua apertura deriva dalla ricezione di un Signal_G da parte del Controller. Manda Signal_G di richiesta di aggiornamento dei dati del sistema di aiuto al Model. Può leggere i dati aggiornati direttamente dal Model dopo aver ricevuto un Signal_G di aggiornamento;

Classi ereditate

- QWindow.

Relazioni con altre classi

DDDMob::View::FacadeView

Relazione entrante, riferimento all'interfaccia per il sistema di aiuto;

DDDMob::Model::CHelpModel::Help

Relazione uscente, riferimento ai dati del sistema di aiuto del modello.

5.3.15 DDDMob :: View :: CSettingsWindow :: SettingsWindow

Descrizione

Classe che rappresenta la finestra per modificare le impostazioni del programma;

Utilizzo

Viene utilizzata per visualizzare le impostazioni del programma. La sua apertura deriva dalla ricezione di un Signal_G da parte del Controller. Manda Signal_G di richiesta di aggiornamento dei dati di configurazione al Model. Può leggere i dati aggiornati direttamente dal Model dopo aver ricevuto un Signal_G di aggiornamento;

Classi ereditate

- QWindow.

Relazioni con altre classi

DDDMob::View::FacadeView

Relazione entrante, riferimento all'interfaccia per le impostazioni del programma;



...Model::CBasicEditor::CConverter::CSettingsModel::Setting

Relazione uscente, riferimento alle impostazioni del modello.

5.3.16 DDDMob :: View :: CErrorWindow :: ErrorMessageBox

Descrizione

Classe che rappresenta una finestra che riporta messaggi d'errore per l'utente;

Utilizzo

La sua apertura deriva dalla ricezione di un Signal_G da parte del Model, generato nel caso in cui si verifichi una inconsistenza durante l'azione del Model. Il Signal_G include un codice di errore e visualizza il relativo messaggio di errore. Il messaggio di errore verrà recuperato mediante QtLinguist .

5.3.17 DDDMob :: View :: CFileWindow :: FileOpener

Descrizione

Classe che rappresenta la finestra di apertura file;

Utilizzo

Viene utilizzata per aprire un file. La sua apertura deriva dalla ricezione di un Signal_G da parte del Controller. Manda Signal_G di richiesta di apertura di un file al Model;

Classi ereditate

- QFileDialog .

Relazioni con altre classi

DDDMob::View::FacadeView

Relazione entrante, riferimento all'interfaccia di importazione.

5.3.18 DDDMob :: View :: CFileWindow :: FileExporter

Descrizione

Classe che rappresenta la finestra per l'esportazione dei file convertiti;

Utilizzo

Viene utilizzata per visualizzare la finestra di salvataggio ed esportazione del file. La sua apertura deriva dalla ricezione di un Signal_G da parte del Controller. Manda Signal_G di richiesta di esportazione dei dati della scena $_G$ al Model;

Classi ereditate

- QFileDialog .

Relazioni con altre classi

DDDMob::View::FacadeView

Relazione entrante, riferimento all'interfaccia di esportazione.



5.3.19 DDDMob :: View :: CWidgetElement :: AxisSlider

Descrizione

Classe che rappresenta il widget per modificare dei valori sugli assi x, y, z;

Utilizzo

Viene utilizzata per inviare il signal_G opportuno al controller per la modifica dei valori;

Classi ereditate

- QGroupBox.

Relazioni con altre classi

DDDMob::View::CDockWidgets::Transformation

Relazione entrante, elemento grafico che permette di modificare la rotazione dell'elemento;

DDDMob::View::CDockWidgets::Transformation

Relazione entrante, elemento grafico che permette di modificare la dimensione;

DDDMob::View::CDockWidgets::Transformation

Relazione entrante, elemento grafico che permette di modificare la traslazione dell'elemento.

5.3.20 DDDMob :: View :: CWidgetElement :: ColorPicker

Descrizione

Classe che rappresenta il widget che consente di selezionare un colore;

Utilizzo

Viene utilizzata per permette di modificare il colore selezionato. Modifiche al colore selezionato provocano un signal_G appropriato;

Classi ereditate

- QPushButton.

Relazioni con altre classi

DDDMob::View::CDockWidgets::Material

Relazione entrante, elemento grafico che permette di modificare il colore di diffusione_G del materiale;

DDDMob::View::CDockWidgets::Material

Relazione entrante, elemento grafico che permette di modificare il colore speculare_G del materiale;

DDDMob::View::CDockWidgets::Material

Relazione entrante, elemento grafico che permette di modificare il colore di emissione del materiale.



5.4 C3DObject

5.4.1 DDDMob :: C3DObject :: SceneObject

Descrizione

Classe astratta che rappresenta gli oggetti che è possibile trovare nella scena_G 3D;

Utilizzo

Mette a disposizione i metodi per la modifica degli attributi e per le trasformazioni all'oggetto;

Classi figlie

- DDDMob :: C3DObject :: Light;
- DDDMob :: C3DObject :: Mesh.

Relazioni con altre classi

DDDMob::Model::CBasicEditor::CConverter::CLoaderModel::Scene
Relazione entrante, oggetto selezionato;

DDDMob::Model::CBasicEditor::CConverter::CLoaderModel::Scene
Relazione entrante, oggetti della scena_G.

5.4.2 DDDMob :: C3DObject :: Light

Descrizione

Classe che rappresenta una luce presente nella scena_G 3D;

Utilizzo

Mette a disposizione i metodi per la modifica degli attributi propri della luce;

Classi ereditate

- DDDMob :: C3DObject :: SceneObject.

Relazioni con altre classi

DDDMob::C3DObject::LightType
Relazione uscente, tipo della luce.

5.4.3 DDDMob :: C3DObject :: Mesh

Descrizione

Classe che rappresenta una mesh poligonale presente nella scena_G 3D;

Utilizzo

Mette a disposizione i metodi per la modifica degli attributi propri delle mesh;

Classi ereditate

- DDDMob :: C3DObject :: SceneObject.



5.4.4 DDDMob :: C3DObject :: LightType

Descrizione

Definisce il tipo di luce;

Relazioni con altre classi

DDDMob::C3DObject::Light

Relazione entrante, tipo della luce.



6 Diagrammi di attività

Vengono ora illustrati i diagrammi di attività che descrivono le interazioni dell'utente con l'applicativo 3DMob. È stato disegnato un diagramma ad alto livello che descrive le attività possibili, le quali vengono poi illustrate tramite dei sotto-diagrammi specifici. Per agevolare la lettura dei diagrammi si specifica che i riquadri che contengono testo in grassetto sono da considerarsi attività ad alto livello, descritte approfonditamente in sotto-attività tramite i relativi diagrammi. Mentre i riquadri in testo normale sono da considerarsi singole azioni.



6.0.5 Attività principali

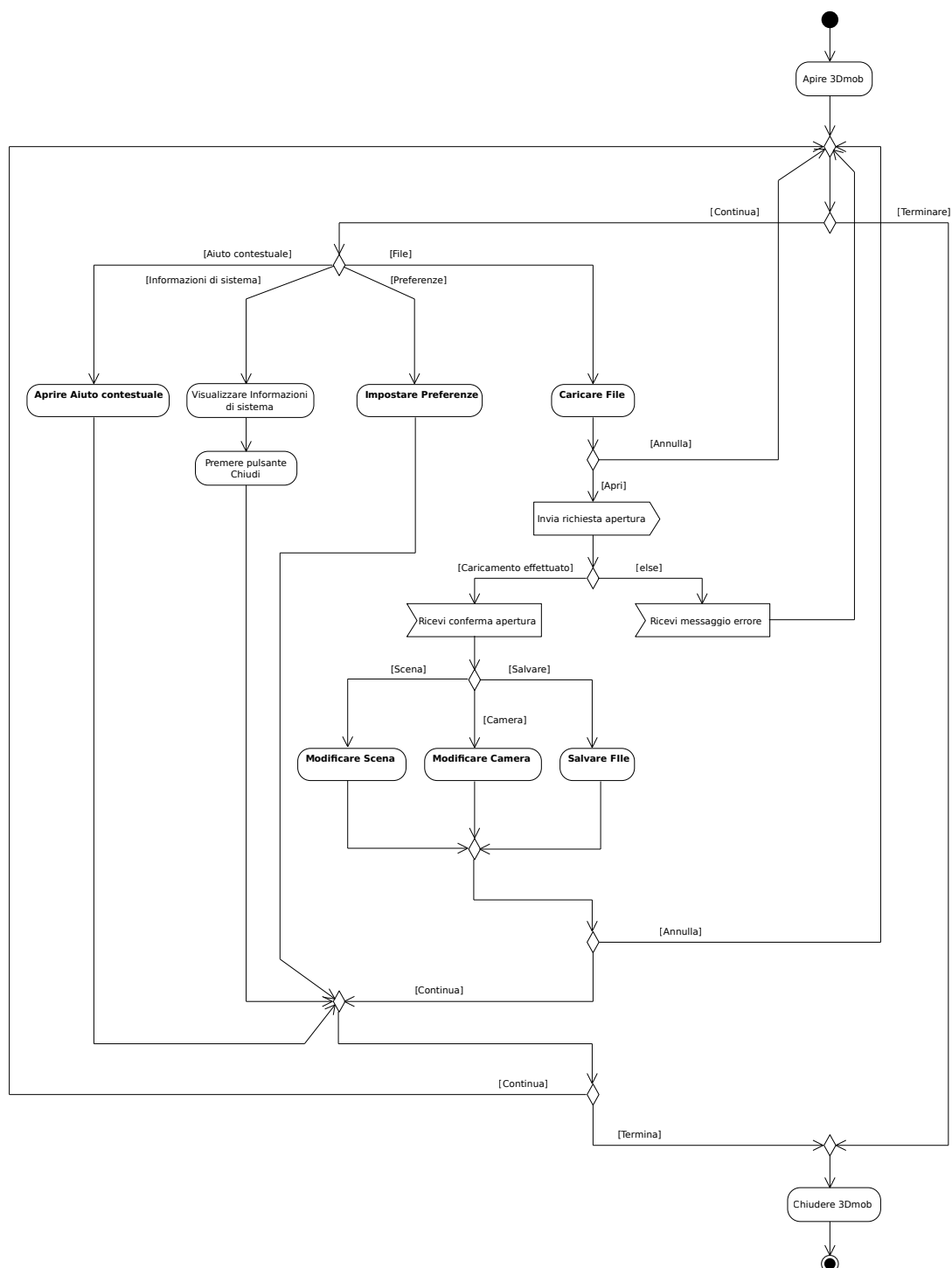


Figura 25: Diagramma attività - Attività principali dell'applicativo 3DMob

L'utente, dopo aver effettuato l'accesso al sistema, ha la possibilità di *Caricare un file*, *Aprire il sistema di Aiuto contestuale*, *Impostare le Preferenze di sistema* e *Visualizzare le informazioni di sistema*. Queste sono le funzioni principali dell'applicazione e possono



essere utilizzate parallelamente senza interrompere lo sviluppo delle singole attività (figura 25).

Nel momento in cui un file viene caricato correttamente l'utente è in grado di *Modificare la scena_G*, *Modificare la camera_G* o *Caricare un altro file*. Una volta applicate le modifiche desiderate è consentito salvare la scena_G ottenuta. È possibile esportare la scena_G anche senza aver apportato modifiche. Dopo aver effettuato correttamente il salvataggio della scena_G è possibile caricare un nuovo file da elaborare oppure uscire dal programma.

6.0.6 Caricare File

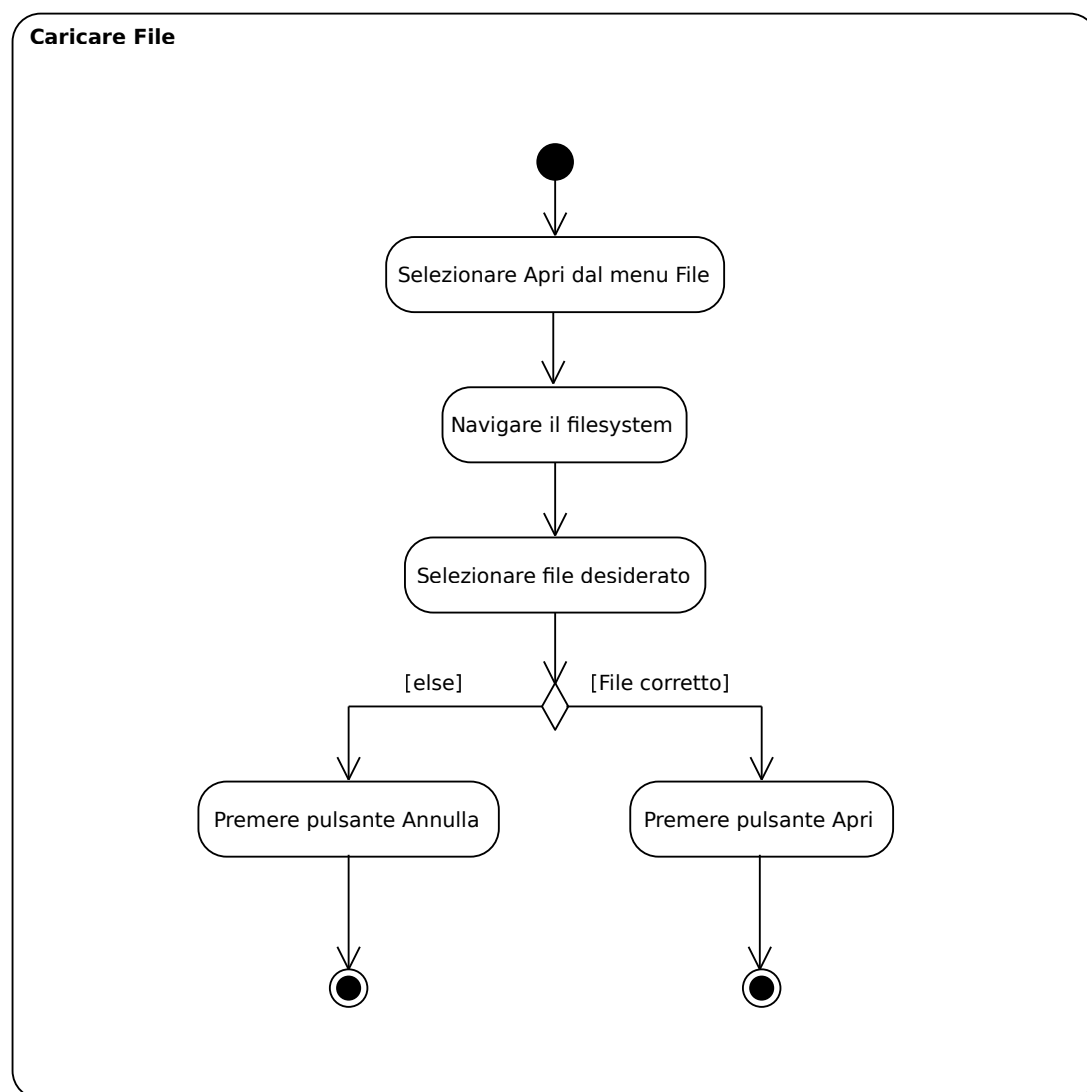


Figura 26: Diagramma attività - Caricamento di un file

L'attività di caricamento in figura 26 di un file comprende le azioni di navigazione del filesystem, di selezione del file scelto e della conferma di apertura. L'utente può opzionalmente annullare il caricamento in qualsiasi momento e di conseguenza ritornare alla finestra principale dell'applicazione.



La modifica della scena_G in figura 27 è l'attività più articolata che l'utente può effettuare tramite l'interfaccia grafica. Ha la possibilità di selezionare un elemento della scena_G ed eseguire su di esso delle modifiche diverse a seconda che sia un oggetto oppure una luce. Inoltre è possibile aggiungere una fonte luminosa oppure agire sui pulsanti di *Annulla* e *Ripristina* per annullare azioni o ripristinare quelle annullate in ordine cronologico.

6.0.8 Modificare Camera

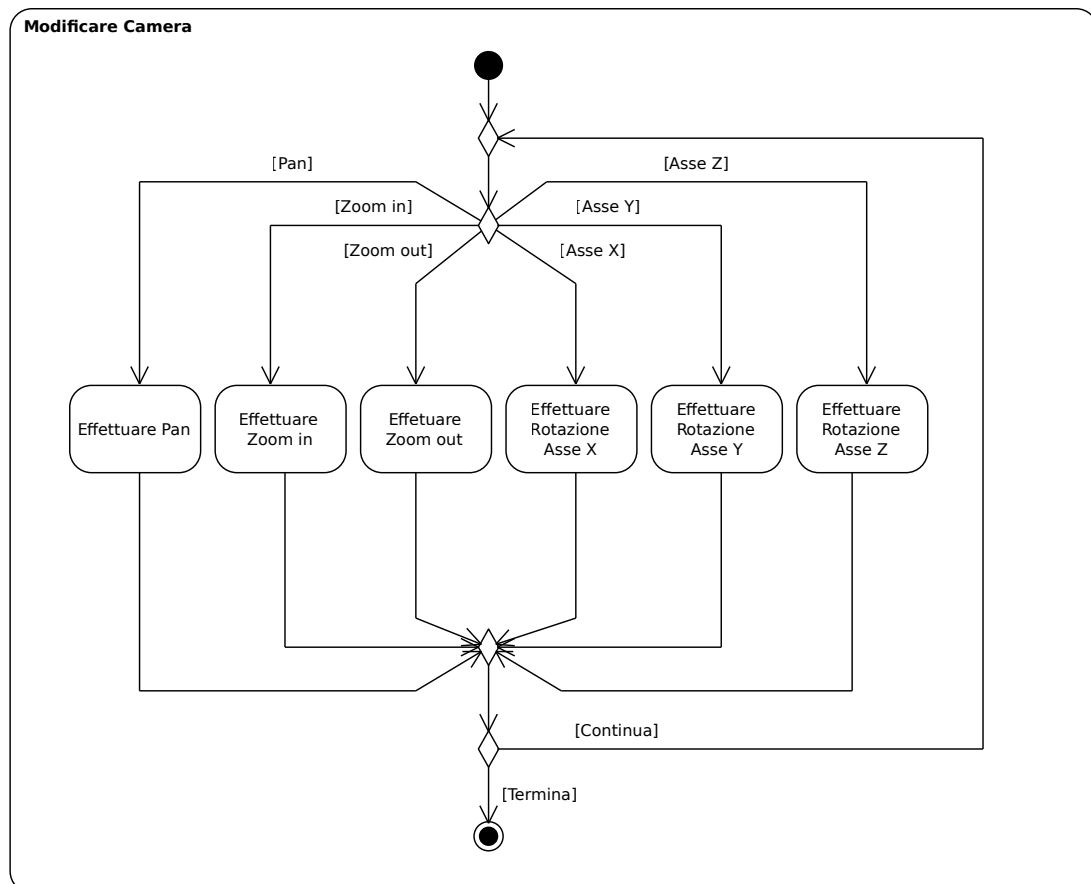


Figura 28: Diagramma attività - Modifica della camera

La modifica della camera_G in figura 28 è un'attività eseguibile direttamente sull'anteprima della scena_G e comporta dei cambiamenti nella visualizzazione del modello che non implicano modifiche reali alla scena_G caricata.



6.0.9 Salvare File

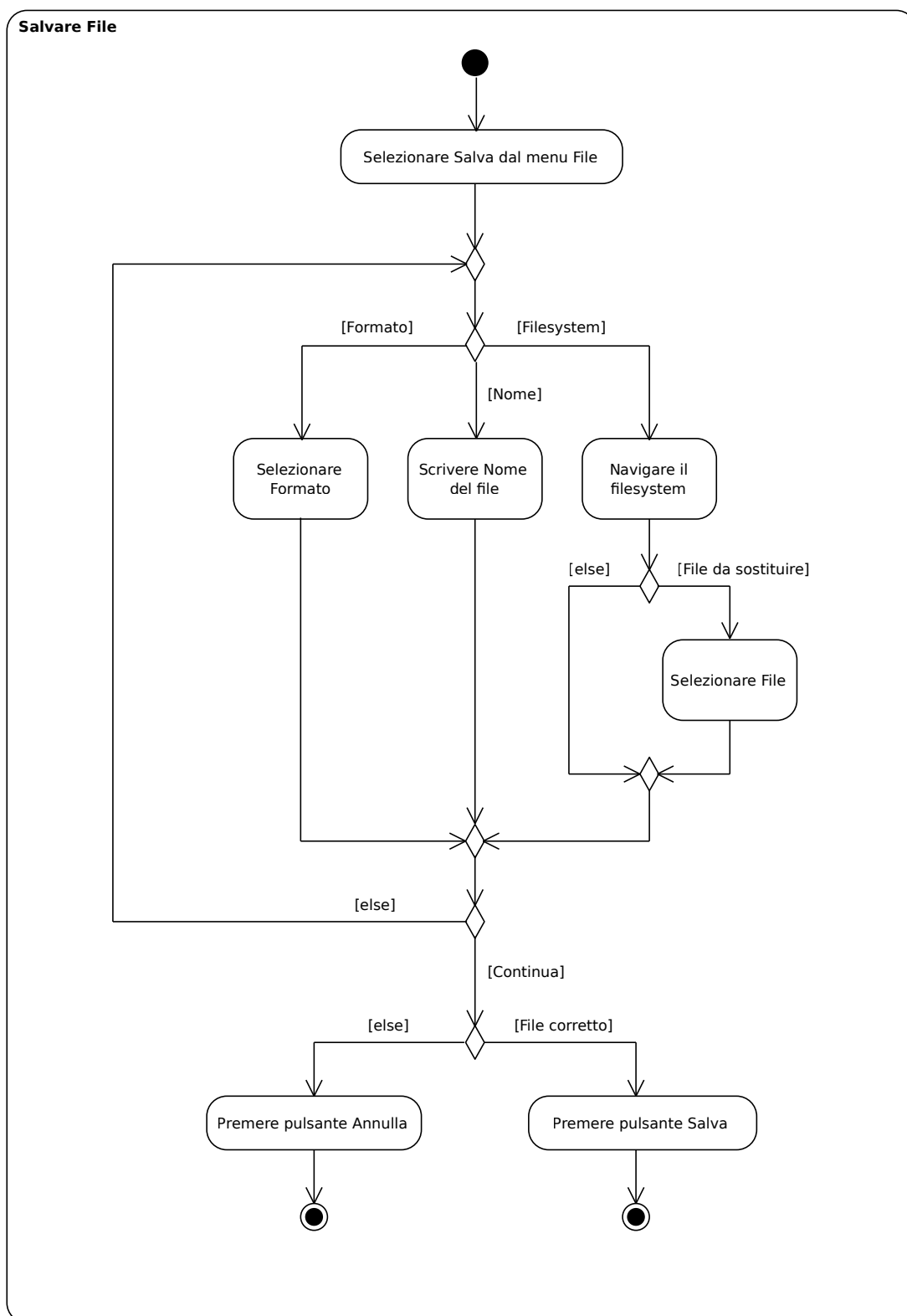


Figura 29: Diagramma attività - Salvataggio della scena



Il salvataggio di una scena_G descritto in figura 29 viene svolto attraverso una finestra che permette di navigare il filesystem per selezionare l'esatta posizione in cui salvare oppure selezionare un file da sovrascrivere. È inoltre possibile scrivere il percorso del file e scegliere il formato del file in output.

La pressione del pulsante *Conferma* avvierà il processo di salvataggio mentre la pressione del *Annulla* riporterà l'utente alla finestra principale.

6.0.10 Impostare Preferenze

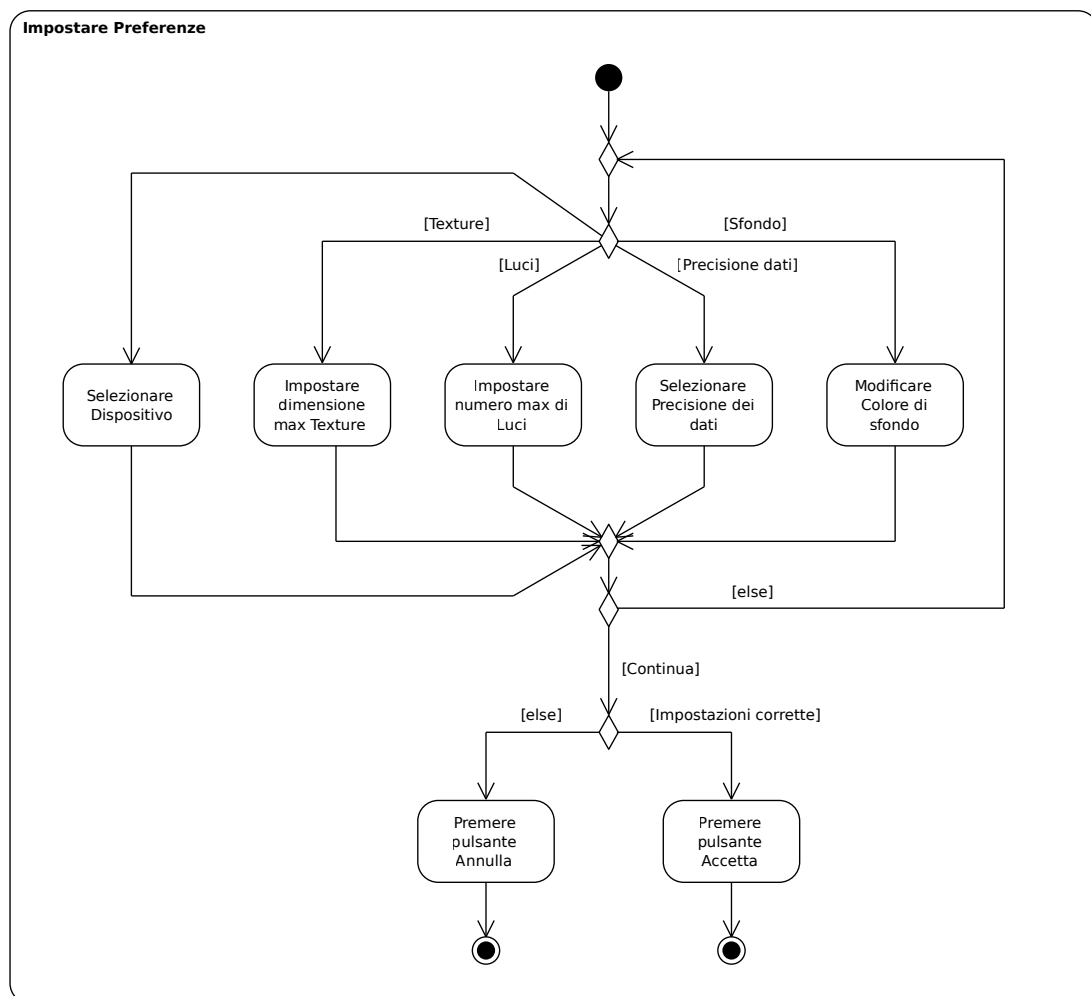


Figura 30: Diagramma attività - Impostazione delle preferenze del sistema

L'utente, come illustrato in figura 30, può visualizzare la finestra di impostazioni generali dell'applicazione nella quale può impostare la precisione dei dati della scena_G e i limiti intrinseci dei dispositivi sui quali si vuole esportare il modello tridimensionale di partenza.



6.0.11 Aprire Aiuto contestuale

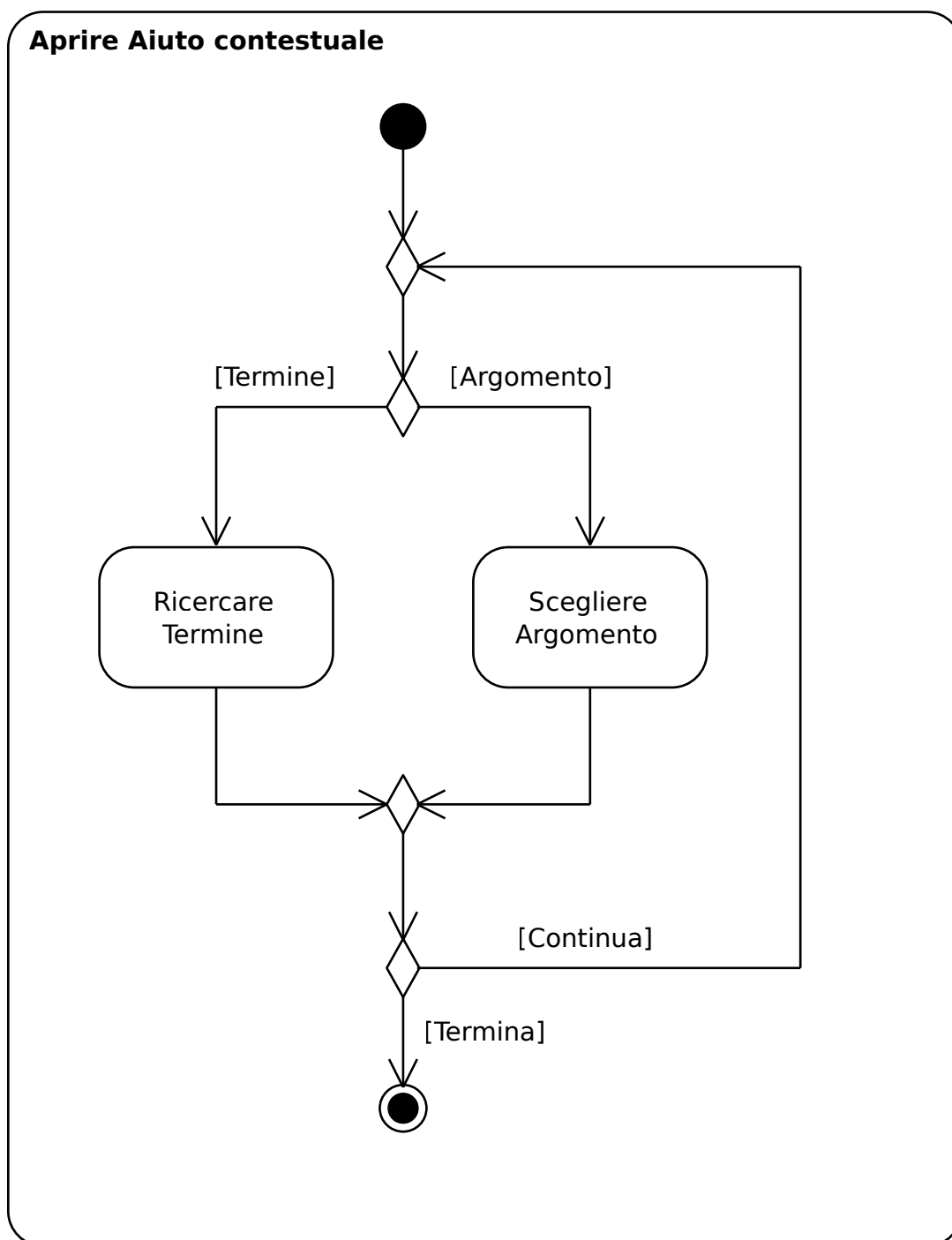


Figura 31: Diagramma attività - Apertura della finestra di aiuto contestuale

L'aiuto contestuale descritto in figura 31 consente di ricercare un termine e visualizzarne le relative informazioni, oppure di scegliere un particolare argomento del quale si vuole approfondire la conoscenza.



7 Design Pattern

I Design Pattern_G sono soluzioni a problemi ricorrenti. Adottare i Design Pattern_G semplifica l'attività di progettazione, favorisce il riutilizzo del codice e rende l'architettura più manutenibile.

I Design Pattern_G possono essere suddivisi in:

- **Design Pattern_G architetturali:** definiscono l'architettura dell'applicazione ad un livello più elevato;
- **Design Pattern_G creazionali:** consentono di nascondere i costruttori delle classi, permettendo di creare oggetti senza conoscere la loro implementazione;
- **Design Pattern_G strutturali:** consentono di riutilizzare classi pre-esistenti, fornendo un'interfaccia più adatta;
- **Design Pattern_G comportamentali:** definiscono soluzioni per le interazioni tra oggetti.

Per una descrizione generale e più approfondita dei Design Pattern_G utilizzati si veda l'Appendice A.

Nella realizzazione del progetto 3DMob si è deciso di implementare i seguenti Design Pattern_G.

7.1 Design Pattern architetturali

7.1.1 MVC

- **Scopo dell'utilizzo:** È stato scelto il pattern MVC per separare la logica dell'applicazione dalla rappresentazione grafica;
- **Contesto d'utilizzo:** Il pattern MVC viene utilizzato per l'architettura generale dell'applicazione. Viene utilizzato il sistema Signal_G e Slot di Qt_G, descritto nella sezione 2.2.1, per far comunicare i componenti tra loro.

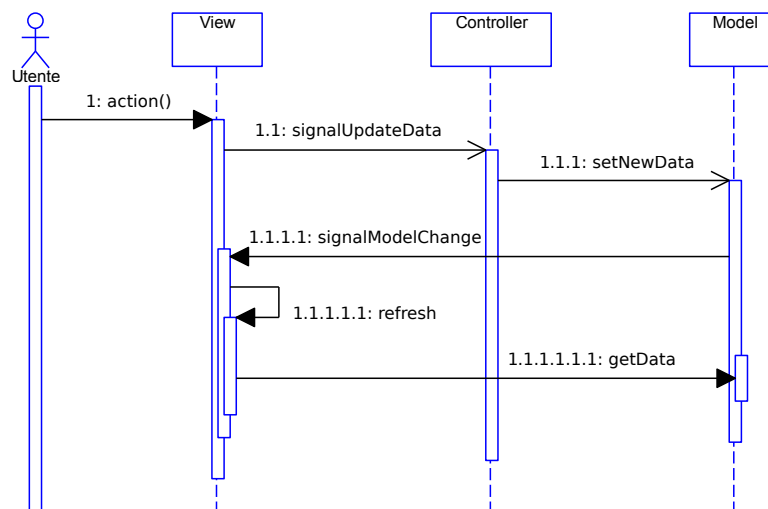


Figura 32: Diagramma di sequenza - Modifica della scena



Ogni modifica effettuata dall'utente sulla View viene inviata al Controller che invia un comando al Model. Il Model notifica la View di ogni cambiamento e la View va a recuperare i dati aggiornati dal Model.

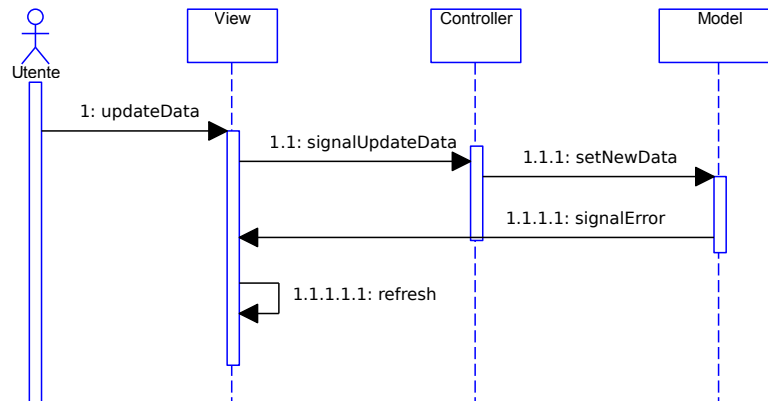


Figura 33: Diagramma di sequenza - Modifica della scena con errore

Nel caso si verifichi un errore il Model notifica la View dell'errore.

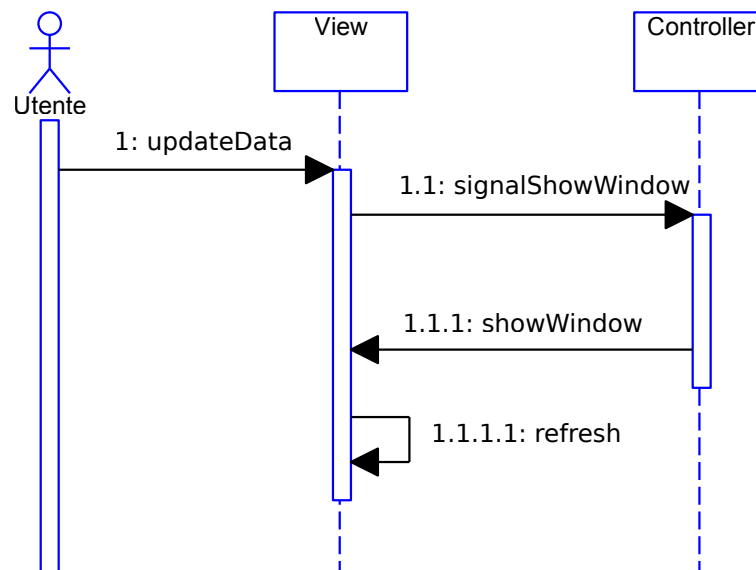


Figura 34: Diagramma di sequenza - Selezione ed apertura della vista

Nell'apertura delle finestre Settings, Help e About, viene inviato un Signal_G dalla View al Controller. Successivamente il Controller seleziona la nuova finestra e lo comunica alla View.



7.2 Design Pattern creazionali

7.2.1 Singleton

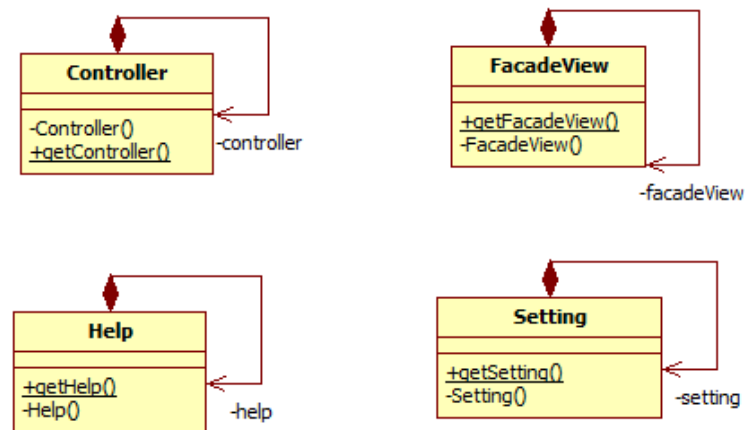


Figura 35: Diagramma del Design Pattern Singleton

- **Scopo dell'utilizzo:** Viene usato il pattern Singleton per le classi che devono avere un'unica istanza durante l'esecuzione dell'applicazione;
- **Contesto d'utilizzo:** Le classi che devono avere un'unica istanza sono:
 - Help;
 - Settings;
 - FacadeView;
 - Controller.

7.3 Design Pattern strutturali

7.3.1 Adapter

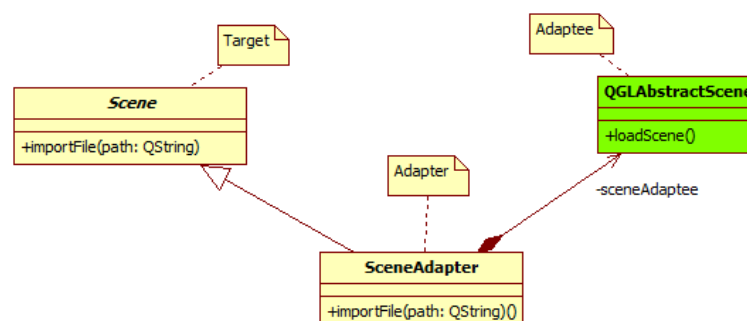


Figura 36: Applicazione di Adapter in 3DMob

- **Scopo dell'utilizzo:** Il pattern Adapter viene utilizzato per adattare una classe riutilizzando un oggetto già esistente. Questo semplifica l'eventuale processo



di sostituzione dell'oggetto esistente, creando un'interfaccia stabile per il resto dell'applicazione;

- **Contesto d'utilizzo:** È stato usato per adattare la classe che rappresenta la scena $_{G}$ in Qt3D. `SceneAdapter` adatta `QGLAbstractScene`.

7.3.2 Facade

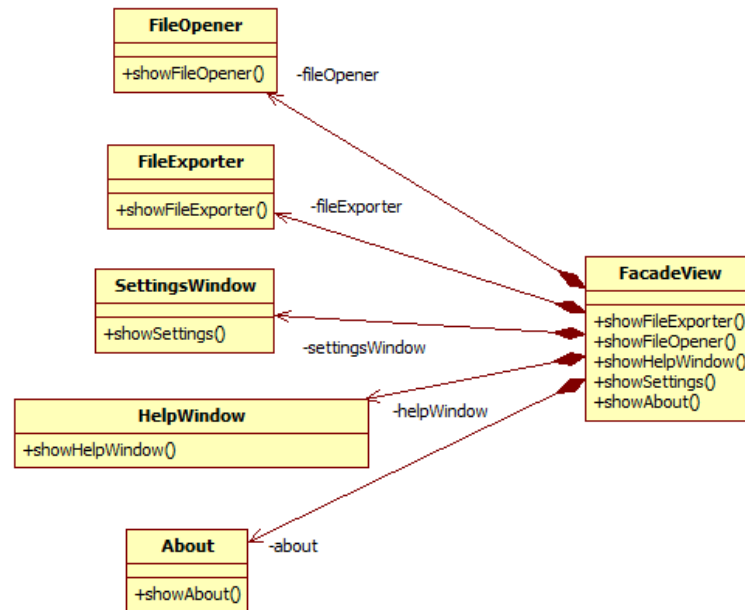


Figura 37: Applicazione di Facade in 3DMob

- **Scopo dell'utilizzo:** Il pattern Facade viene usato per fornire un'interfaccia unica a più classi;
- **Contesto d'utilizzo:** `FacadeView` è una Facade che presenta un'interfaccia per tutti gli oggetti che compongono la View:

- `FileOpener`;
- `FileExporter`;
- `SettingsWindow`;
- `HelpWindow`;
- `About`.



7.3.3 Decorator

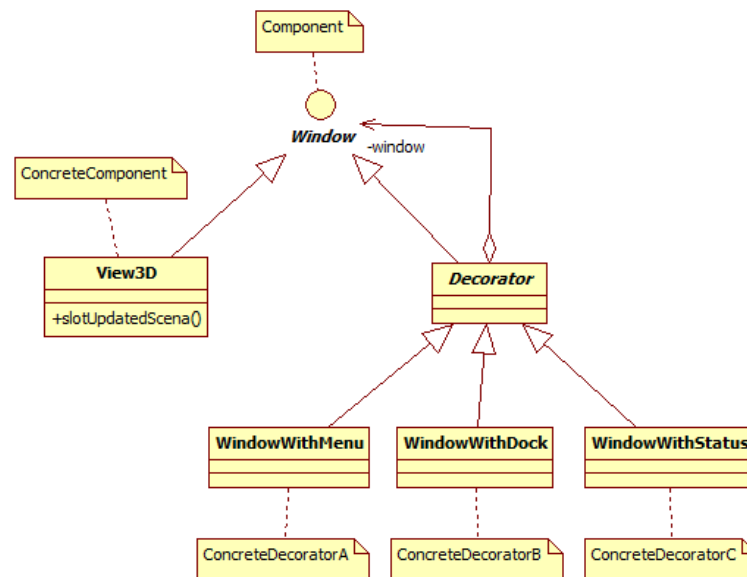


Figura 38: Applicazione di Decorator in 3DMob

- **Scopo dell'utilizzo:** Il pattern Decorator viene usato per aggiungere delle caratteristiche minori ad un oggetto di base, rendendolo facilmente estensibile;
- **Contesto d'utilizzo:** Viene usato il pattern Decorator per estendere `View3D`, la classe che rappresenta la vista 3D con interfaccia minimale. `View3D` viene esteso con le seguenti classi ConcreteDecorator:
 - `WindowWithMenu`;
 - `WindowWithDock`;
 - `WindowWithStatus`.

7.4 Design Pattern comportamentali

7.4.1 Command

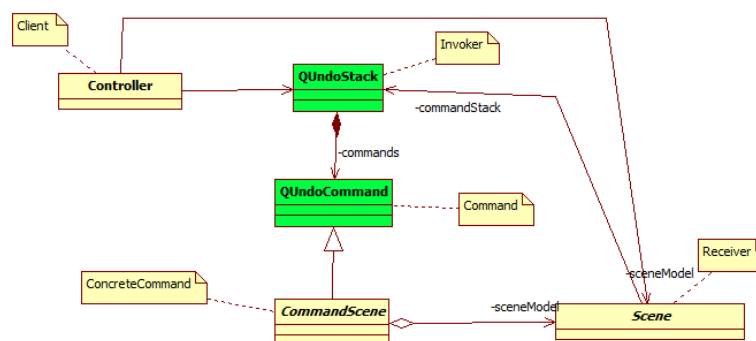


Figura 39: Applicazione di Command in 3DMob



- **Scopo dell'utilizzo:** Il pattern Command viene utilizzato per separare il codice di un'azione dal codice che richiede l'esecuzione dello stesso;
- **Contesto d'utilizzo:** Viene usato per la gestione dei comandi che modificano la scena_G. Le classi che implementano i comandi sono:

- `CommandAddLight;`
- `CommandSelectObject;`
- `CommandPosition;`
- `CommandScale;`
- `CommandRotation;`
- `CommandLightType;`
- `CommandShininess;`
- `CommandEmission;`
- `CommandColor;`
- `CommandColorEmission;`
- `CommandColorSpecular;`
- `CommandColorDiffuse.`

La classe incaricata di eseguire i comandi è `QUndoStack`. Il `Controller` è il Client del Command e riceve un `SignalG` dalla View con cui interagisce l'utente.

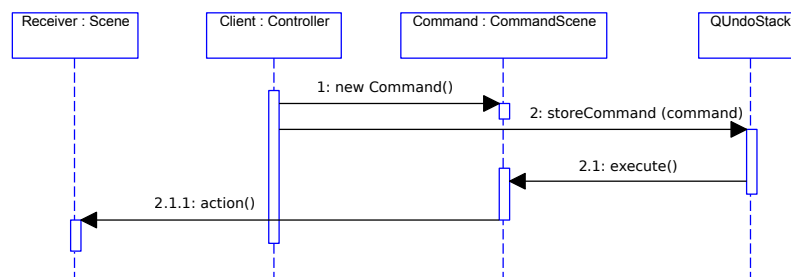


Figura 40: Diagramma di sequenza - Gestore di comandi della scena

Dopo che la View ha inviato un `SignalG` al Controller, viene creato un `CommandScene` che viene inserito nella `QUndoStack` presente nel Model. `QUndoStack` invoca il comando e modifica la `Scene`.



7.4.2 Strategy

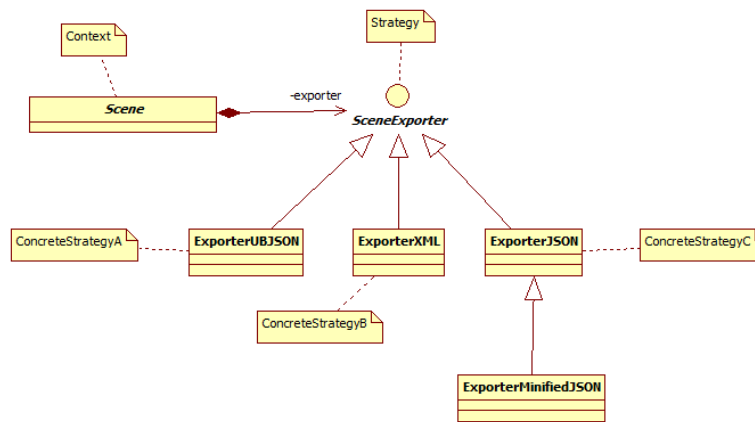


Figura 41: Applicazione di Strategy in 3DMob

- **Scopo dell'utilizzo:** Il pattern Strategy viene usato per isolare più algoritmi che svolgono la stessa funzione dal codice che esegue la funzione;
- **Contesto d'utilizzo:** Viene usato per gestire gli algoritmi di esportazione in `SceneExporter`. Dopo che l'utente seleziona il tipo di file da esportare, viene selezionato l'algoritmo di conversione corrispondente tra una delle `ConcreteStrategy`:
 - `ExporterUBJSON`;
 - `ExporterXML`;
 - `ExporterJSON`;
 - `ExporterMinifiedJSON`.



8 Stime di fattibilità e di bisogno di risorse

L'architettura definita precedentemente ha raggiunto un livello di dettaglio sufficiente per fornire una stima sulla fattibilità e di bisogno di risorse.

L'analisi dell'architettura progettata ha permesso di constatare che le tecnologie che si è scelto di adottare risultano sufficientemente adeguate per la realizzazione del prodotto e riescono a ricoprire le esigenze progettuali.

Gli strumenti scelti sono conosciuti dalla maggioranza dei componenti del gruppo che si impegneranno comunque ad approfondire le loro conoscenze inerenti alla grafica 3D_G e le librerie Qt3D.

Gli strumenti utilizzati sono:

- Qt_G framework_G per la stesura del codice;
- Qt3D, libreria di Qt_G, per importare file 3D e visualizzare la scena_G all'interno del software;
- Qmake per la compilazione automatica del codice e la produzione del Makefile;
- QtPlugins per estendere le funzionalità di Qt_G.

Tali strumenti potranno garantire una realizzazione efficace di tutti gli aspetti architeturali.



9 Tracciamento

Seguono le tabelle di tracciamento tra componenti e requisiti. Per semplicità di lettura, requisiti associati a figli di un componente non sono riportati anche nel padre.

I componenti riguardanti il Controller risultano senza requisiti tracciati per la natura di semplice gestore del flusso di informazioni tra View e Model.

I componenti DDDMob::View::CDockWidgets e DDDMob::View::CWidgetElement hanno il solo scopo di contenere classi generiche per l'input utente da interfaccia grafica. Ciò significa che questi non abbiano requisiti associati ma vengono utilizzati da altri componenti della View per soddisfare i propri requisiti.

9.1 Tracciamento componenti - requisiti

Componenti	Requisiti
DDDMob	
↳ DDDMob::Model	
↳ DDDMob::Model::CHelpModel	R2F7.3
↳ DDDMob::Model::CBasicEditor	
↳ DDDMob::Model::CBasicEditor::CConverter	
↳ DDDMob::Model::CBasicEditor::CConverter::CSettingsModel	R0F1.1 R0F1.2 R0V2 R1F7.1.1.1 R0V2.1 R0V2.2 R2F1.6
↳ DDDMob::Model::CBasicEditor::CConverter::CExportModel	R0F1 R0F1.3.1 R0F1.3.2 R0F1.3.3 R0F1.3.4 R0Q4 R1Q5 R1F1.3.5 R0F1.3
↳ DDDMob::Model::CBasicEditor::CConverter::CLoaderModel	R0F8.1 R2F8.2 R1F8.4.1 R0F8.3.3 R0V2.1 R0V2.2
↳ DDDMob::Controller	
↳ DDDMob::Controller::CSceneController	
↳ DDDMob::View	
↳ DDDMob::View::CDockWidgets	



↳ DDDMob::View::CMainWindow	R2F7.1 R2F7.1.2.1 R2F7.1.2.2 R2F7.1.2.3 R1F8.3 R1F8.3.1.1.1 R1F8.3.1.1.2 R1F8.3.1.1.3 R1F8.3.1.2.1 R1F8.3.1.1.4 R1F8.4 R1F8.3.1.1.5 R1F8.3.1.1.5.1 R2F8.3.1 R2F7.1.1 R2F8.3.2 R0F8.3.3 R1F8.3.4 R2F7.1.2 R1F8.3.1.2.2 R1F8.3.5 R1F8.3.1.2.3 R2F8.3.1.1 R2F8.3.1.2 R0F8.3.1.2.4 R2F8.3.1.2.5 R1F8.3.1.1.4.1 R1F8.3.1.1.4.2 R2F8.3.4.1
↳ DDDMob::View::CHelpWindow	R2F7.3 R2F7.3.1 R2F7.3.2
↳ DDDMob::View::CSettingsWindow	R0F1.1 R0F1.2 R1F7.1.1.1 R2F1.6 R1F1.6.3
↳ DDDMob::View::CErrorWindow	R2F7.2 R2F7.2.1 R2F7.2.2 R2F7.2.3
↳ DDDMob::View::CFileWindow	R0F8 R0F1.4 R2F1.5 R2F1.7
↳ DDDMob::View::CWidgetElement	



→ DDDMob::C3DObject	R1F8.3 R1F8.3.1.2.1 R2F8.3.1 R1F8.3.1.2.2 R1F8.3.1.2.3 R2F8.3.1.1 R2F8.3.1.2 R0F8.3.1.2.4 R2F8.3.1.2.5
---------------------	--

Tabella 2: Tabella componenti/requisiti



9.2 Tracciamento requisiti - componenti

Requisito	Descrizione	Componenti
R0F1	Il programma deve essere in grado di salvare la scena _G in un file secondo uno dei formati permessi	DDDMob::Model:- CBasicEditor:- CConverter:- CExportModel
↳ R0F1.1	L'utente deve poter decidere se salvare la scena _G in formato leggibile o in formato minificato _G	DDDMob::Model:- CBasicEditor:- CConverter:- CSettingsModel DDDMob::View:- CSettingsWindow
↳ R0F1.2	L'utente deve poter decidere se salvare la scena _G utilizzando dati in virgola mobile con precisione singola o precisione doppia	DDDMob::Model:- CBasicEditor:- CConverter:- CSettingsModel DDDMob::View:- CSettingsWindow
↳ R0F1.3	Il programma deve mantenere nell'esportazione le caratteristiche del solido	DDDMob::Model:- CBasicEditor:- CConverter:- CExportModel
↳ R0F1.3.1	Il programma dovrà esportare file che mantengano le normali e i vertici del solido presenti nell'oggetto importato	DDDMob::Model:- CBasicEditor:- CConverter:- CExportModel
↳ R0F1.3.2	Il programma dovrà esportare file che mantengano le caratteristiche delle texture presenti nell'oggetto importato, ovvero l'istogramma della texture importata dovrà essere uguale a quello della texture esportata	DDDMob::Model:- CBasicEditor:- CConverter:- CExportModel
↳ R0F1.3.3	Il programma dovrà esportare file che rispettino i parametri dei materiali (colore e emissività) presenti nel modello importato, ovvero siano gli stessi a meno di modifiche utente	DDDMob::Model:- CBasicEditor:- CConverter:- CExportModel
↳ R0F1.3.4	Il programma dovrà esportare file che rispettino le sorgenti di luce presenti nel modello importato, ovvero mantengano le fonti di luce non in soprannumero rispetto al massimo configurato	DDDMob::Model:- CBasicEditor:- CConverter:- CExportModel



→ R1F1.3.5	Il programma deve mantenere le animazioni presenti nel file importato durante la fase di esportazione	DDDMob::Model:- CBasicEditor:- CConverter:- CExportModel
→ R0F1.4	L'utente deve poter scegliere il formato JSON _G per l'esportazione	DDDMob::View:- CFileWindow
→ R2F1.5	L'utente deve poter scegliere il formato XML _G per l'esportazione	DDDMob::View:- CFileWindow
→ R2F1.6	L'utente deve poter configurare i limiti di importazione della scena _G	DDDMob::Model:- CBasicEditor:- CConverter:- CSettingsModel DDDMob::View:- CSettingsWindow
→ R1F1.6.3	L'utente deve poter selezionare i limiti di importazione da una lista di valori predefiniti	DDDMob::View:- CSettingsWindow
→ R2F1.7	L'utente deve poter scegliere il formato binario _G Universal Binary JSON _G Specification (http://ubjson.org/) per l'esportazione	DDDMob::View:- CFileWindow
R0V2	Il programma, di default, deve esportare file che rispettino i limiti dell'iPhone 4S	DDDMob::Model:- CBasicEditor:- CConverter:- CSettingsModel
→ R0V2.1	Non devono essere presenti più di 8 luci nella scena _G esportata	DDDMob::Model:- CBasicEditor:- CConverter:- CLoaderModel DDDMob::Model:- CBasicEditor:- CConverter:- CSettingsModel
→ R0V2.2	Non devono essere presenti texture di larghezza e altezza superiore a 4096 pixel nella scena _G esportata	DDDMob::Model:- CBasicEditor:- CConverter:- CLoaderModel DDDMob::Model:- CBasicEditor:- CConverter:- CSettingsModel



R0Q4	Il programma deve esportare i dati in un formato che segua il workflow_G necessario al rendering _G della scena _G in OpenGL ES _G 2.0	DDDMob::Model:- CBasicEditor:- CConverter:- CExportModel
R1Q5	Il programma deve esportare i dati in un formato che segua il workflow_G necessario al rendering _G della scena _G in OpenGL ES _G 3.0	DDDMob::Model:- CBasicEditor:- CConverter:- CExportModel
↳ R2F7.1	Il programma deve mostrare un'anteprima della scena _G che verrà esportata	DDDMob::View:- CMainWindow
↳ R2F7.1.1	L'anteprima della scena _G da esportare deve prevedere uno sfondo	DDDMob::View:- CMainWindow
↳ R1F7.1.1.1	L'utente deve poter modificare il colore dello sfondo dell'anteprima	DDDMob::Model:- CBasicEditor:- CConverter:- CSettingsModel DDDMob::View:- CSettingsWindow
↳ R2F7.1.2	L'anteprima della scena _G da esportare deve essere modificabile	DDDMob::View:- CMainWindow
↳ R2F7.1.2.1	L'utente deve poter ruotare liberamente l'anteprima	DDDMob::View:- CMainWindow
↳ R2F7.1.2.2	L'utente deve poter modificare lo zoom dell'anteprima	DDDMob::View:- CMainWindow
↳ R2F7.1.2.3	L'utente deve poter spostare la camera _G e navigare nell'anteprima	DDDMob::View:- CMainWindow
↳ R2F7.2	Deve essere previsto un sistema di notifica degli errori e delle informazioni	DDDMob::View:- CErrorWindow
↳ R2F7.2.1	Il sistema di notifica deve poter segnalare gli errori nell'apertura del file	DDDMob::View:- CErrorWindow
↳ R2F7.2.2	Il sistema di notifica deve poter segnalare i malfunzionamenti del sistema	DDDMob::View:- CErrorWindow
↳ R2F7.2.3	Il sistema di notifica deve poter segnalare le perdite di informazione nella conversione	DDDMob::View:- CErrorWindow
↳ R2F7.3	Deve essere previsto un sistema di aiuto per gli utenti	DDDMob::Model:- CHelpModel DDDMob::View:- CHelpWindow



↳ R2F7.3.1	L'utente deve poter cercare un termine nel sistema d'aiuto	DDDMob::View::CHelpWindow
↳ R2F7.3.2	L'utente deve poter visualizzare le informazioni relative ad un argomento di interesse per l'utente nel sistema d'aiuto	DDDMob::View::CHelpWindow
R0F8	L'utente deve poter scegliere il file da aprire o importare	DDDMob::View::CFileWindow
↳ R0F8.1	Deve essere possibile importare un file 3ds _G	DDDMob::Model::CBasicEditor::CConverter::CLoaderModel
↳ R2F8.2	Il programma deve essere in grado di leggere i file in JSON _G che crea	DDDMob::Model::CBasicEditor::CConverter::CLoaderModel
↳ R1F8.3	L'utente deve poter modificare la scena _G importata	DDDMob::C3DObject DDDMob::View::CMainWindow
↳ R2F8.3.1	L'utente deve poter modificare un oggetto od una fonte di luce	DDDMob::C3DObject DDDMob::View::CMainWindow
↳ R2F8.3.1.1	L'utente deve poter modificare un oggetto	DDDMob::C3DObject DDDMob::View::CMainWindow
↳ R1F8.3.1.1.1	L'utente deve poter ruotare l'oggetto selezionato negli assi X, Y e Z	DDDMob::View::CMainWindow
↳ R1F8.3.1.1.2	L'utente deve poter traslare l'oggetto selezionato negli assi X, Y e Z	DDDMob::View::CMainWindow
↳ R1F8.3.1.1.3	L'utente deve poter scalare le dimensioni dell'oggetto selezionato	DDDMob::View::CMainWindow
↳ R1F8.3.1.1.4	Deve essere possibile modificare le caratteristiche dei materiali dell'oggetto selezionato	DDDMob::View::CMainWindow
↘ R1F8.3.1.1.4.1	Deve essere possibile modificare il colore speculare _G dei materiali di un oggetto nella scena _G	DDDMob::View::CMainWindow
↘ R1F8.3.1.1.4.2	Deve essere possibile modificare il colore di diffusione _G per i materiali dell'oggetto selezionato.	DDDMob::View::CMainWindow



↪ R1F8.3.1.1.5	Deve essere possibile modificare l'opacità dell'oggetto selezionato	DDDMob::View::CMainWindow
↪ R1F8.3.1.1.5.1	Deve essere possibile modificare l'opacità di parte dell'oggetto selezionato	DDDMob::View::CMainWindow
↪ R2F8.3.1.2	L'utente deve poter modificare una fonte di luce	DDDMob::C3DObject DDDMob::View::CMainWindow
↪ R1F8.3.1.2.1	L'utente deve poter modificare l'intensità della luce selezionata	DDDMob::C3DObject DDDMob::View::CMainWindow
↪ R1F8.3.1.2.2	L'utente deve poter modificare il colore delle luci	DDDMob::C3DObject DDDMob::View::CMainWindow
↪ R1F8.3.1.2.3	L'utente deve poter modificare la tipologia della fonte di luce	DDDMob::C3DObject DDDMob::View::CMainWindow
↪ R0F8.3.1.2.4	L'utente deve poter ruotare la luce selezionata negli assi X, Y e Z	DDDMob::C3DObject DDDMob::View::CMainWindow
↪ R2F8.3.1.2.5	L'utente deve poter traslare la luce selezionata negli assi X, Y e Z	DDDMob::C3DObject DDDMob::View::CMainWindow
↪ R2F8.3.2	L'utente deve poter annullare o ripristinare le azioni di modifica effettuate su oggetti o luci della scena _G	DDDMob::View::CMainWindow
↪ R0F8.3.3	Tutte le modifiche effettuate devono riflettersi sull'anteprima	DDDMob::Model::CBasicEditor::CConverter::CLoaderModel DDDMob::View::CMainWindow
↪ R1F8.3.4	L'utente deve poter aggiungere una fonte di luce alla scena _G con valori di materiale predefiniti (diffusione, speculare ed emissione a bianco; lucenezza a 64 su 128; valore di emissione a 0) e alla posizione (0,0,0)	DDDMob::View::CMainWindow
↪ R2F8.3.4.1	L'utente deve poter scegliere il tipo di fonte luminosa da aggiungere tra omni e spotlight _G	DDDMob::View::CMainWindow



↳ R1F8.3.5	L'utente deve poter selezionare l'oggetto o la fonte di luce che desidera modificare	DDDMob::View::-CMainWindow
↳ R1F8.4	Deve essere possibile importare un file nel formato Wavefront obj	DDDMob::View::-CMainWindow
↳ R1F8.4.1	IL sistema deve poter importare un file nel formato mtl riferito all'interno del file obj importato	DDDMob::Model::-CBasicEditor::-CConverter::-CLoaderModel

Tabella 3: Tabella requisiti / componenti



A Descrizione Design Pattern

A.1 Design Pattern architetturali

A.1.1 MVC

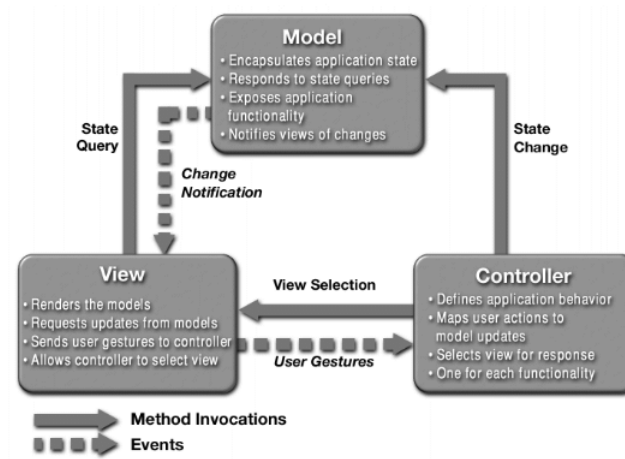


Figura 42: Diagramma del Design Pattern MVC

- **Scopo:** Disaccoppiare le tre componenti seguenti:
 - Model: dati di business e regole di accesso;
 - View: rappresentazione grafica;
 - Controller: reazioni della UI agli input utente.
- **Motivazione:** Lo scopo di molte applicazioni è quello di recuperare dati e visualizzarli in maniera opportuna a seconda delle esigenze degli utenti. Poiché il flusso chiave di informazione avviene tra il dispositivo su cui sono memorizzati i dati e l'interfaccia utente, si è portati a legare insieme queste due parti per ridurre la quantità di codice e migliorare le performance dell'applicazione. Questo approccio, apparentemente naturale, presenta alcuni problemi significativi; uno di questi è che l'interfaccia utente tende a cambiare più in fretta rispetto al sistema di memorizzazione dei dati. Un altro problema, che si ha nel mettere insieme i dati e l'interfaccia utente, è che le applicazioni aziendali tendono ad incorporare logica di business che va al di là della semplice trasmissione di dati. C'è la necessità, quindi, di rendere modulari le funzionalità dell'interfaccia utente in maniera tale da poter facilmente modificare le singole parti. La soluzione a tutto ciò è costituita dal pattern Model-View-Controller (MVC) che separa la modellazione del dominio, la presentazione, e le azioni basate sugli input degli utenti all'interno di tre classi separate;
- **Applicabilità:** Il pattern MVC può essere utilizzato nei seguenti casi:
 - Quando si vuole trattare un gruppo di oggetti come un oggetto singolo;
 - Quando si vuole disaccoppiare View e Model instaurando un protocollo di sottoscrizione e notifica tra loro;
 - Quando si vogliono agganciare più View a un Model per fornire più rappresentazioni del Model stesso.



A.2 Design Pattern creazionali

A.2.1 Singleton

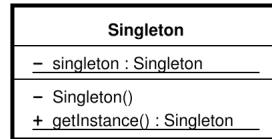


Figura 43: Diagramma del Design Pattern Singleton

- **Scopo:** Assicurare che una classe abbia una sola istanza e fornire un punto d'accesso globale a tale istanza;
- **Motivazione:** È importante poter assicurare che per alcune classi esista una sola istanza. Per far ciò la classe stessa ha la responsabilità di creare le proprie istanze, assicurare che nessun'altra istanza possa essere creata e fornire un modo semplice per accedere all'istanza;
- **Applicabilità:** Il pattern Singleton può essere utilizzato nei seguenti casi:
 - Quando deve esistere esattamente un'istanza di una classe e tale istanza deve essere resa accessibile ai client attraverso un punto di accesso noto a tutti gli utilizzatori;
 - Quando l'unica istanza deve poter essere estesa attraverso la definizione di sottoclassi e i client devono essere in grado di utilizzare le istanze estese senza dover modificare il proprio codice.

A.3 Design Pattern strutturali

A.3.1 Adapter

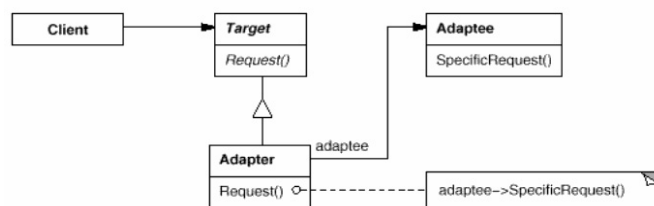


Figura 44: Diagramma del Design Pattern Adapter

- **Scopo:** Convertire l'interfaccia di una classe in un'altra interfaccia richiesta dal client. Consente a classi diverse di operare insieme quando ciò non sarebbe altrimenti possibile a causa di interfacce incompatibili;
- **Motivazione:** A volte una classe di supporto, che è stata progettata con obiettivi di riuso, non può essere riusata semplicemente perché la sua interfaccia non è compatibile con l'interfaccia richiesta da un'applicazione;



- **Applicabilità:** Il pattern Adapter può essere utilizzato nei seguenti casi:
 - quando si vuole usare una classe esistente, ma la sua interfaccia non è compatibile con quella desiderata;
 - quando si vuole creare una classe riusabile in grado di cooperare con classi non correlate o impreviste, cioè con classi che non necessariamente hanno interfacce compatibili;
 - per gli oggetti adapter quando si devono utilizzare diverse sottoclassi esistenti, ma non è pratico adattare la loro interfaccia creando una sottoclasse per ciascuna di esse.

A.3.2 Facade

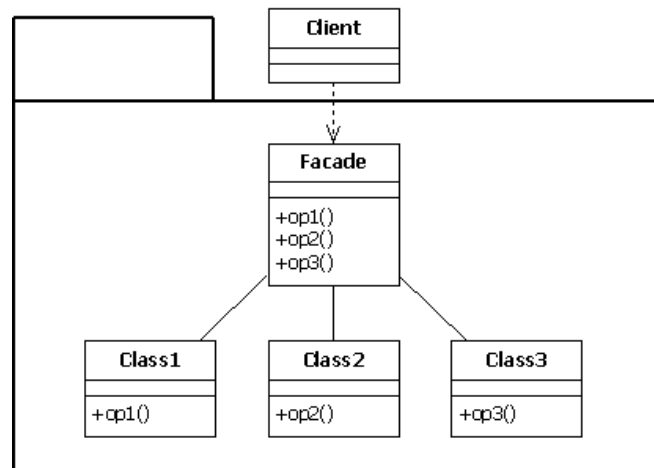


Figura 45: Diagramma del Design Pattern Facade

- **Scopo:** Fornire un'interfaccia unificata per un insieme di interfacce presenti in un sottosistema. Definisce un'interfaccia di livello più alto che rende il sottosistema più semplice da utilizzare;
- **Motivazione:** Suddividere un sistema in sottosistemi aiuta a ridurre la complessità. Un obiettivo comune di progettazione è la minimizzazione delle comunicazioni e delle dipendenze fra i diversi sottosistemi. Un modo per raggiungere questo obiettivo è introdurre un oggetto facade, che fornisce un'interfaccia unica e semplificata per accedere alle funzionalità offerte da un sottosistema;
- **Applicabilità:** Il pattern Facade può essere utilizzato nei seguenti casi:
 - Quando si vuole fornire un'interfaccia semplice a un sottosistema complesso poiché fornisce una vista semplice di base su un sottosistema che si rivela essere sufficiente per la maggior parte dei client;
 - Nei casi in cui ci sono molte dipendenze fra i client e le classi che implementano un'astrazione in quanto si disaccoppia il sottosistema dai client e dagli altri sistemi, promuovendo portabilità e indipendenza dei sottosistemi;
 - Quando si vogliono organizzare i sottosistemi in una struttura a livelli.



A.3.3 Decorator

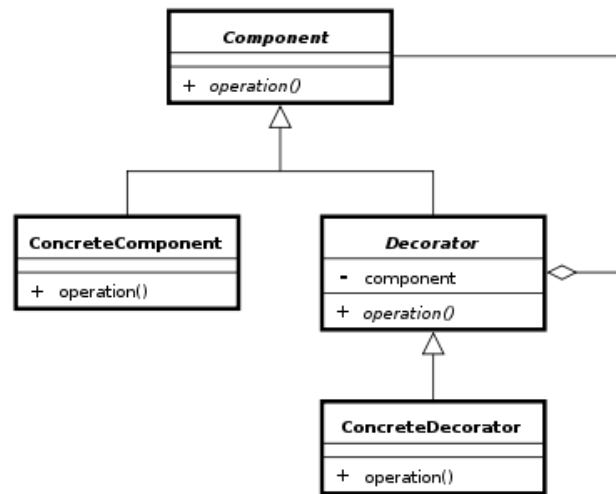


Figura 46: Diagramma del Design Pattern Decorator

Bridge

- **Scopo:** Aggiungere dinamicamente responsabilità a un oggetto. I Decorator forniscono un'alternativa flessibile alla definizione di sottoclassi come strumento per l'estensione delle funzionalità;
- **Motivazione:** Talvolta si vogliono aggiungere responsabilità a singoli oggetti e non a un'intera classe.

Un modo per aggiungere responsabilità consiste nel racchiudere il componente da decorare in un altro. L'oggetto contenitore è chiamato Decorator. Il Decorator ha un'interfaccia conforme a quella dell'elemento decorato, in modo da rendere trasparente la sua presenza ai client. Il decorator trasferisce le richieste al componente decorato e può svolgere azioni aggiuntive prima o dopo il trasferimento della richiesta;

- **Applicabilità:** Il pattern Decorator può essere utilizzato nei seguenti casi:
 - Si vuole poter aggiungere responsabilità a singoli oggetti dinamicamente ed in modo trasparente;
 - Si vuole poter togliere responsabilità agli oggetti;
 - si vuole definire un gran numero di estensioni indipendenti.



A.4 Design Pattern comportamentali

A.4.1 Command

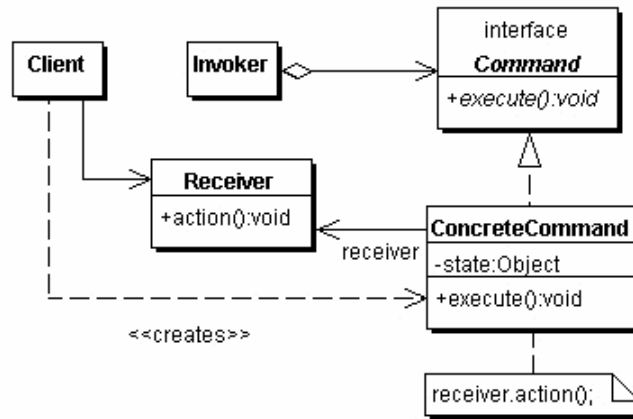


Figura 47: Diagramma del Design Pattern Command

- **Scopo:** Incapsula una richiesta in un oggetto, consentendo di parametrizzare i client con richieste diverse, accodare o mantenere uno storico delle richieste e gestire richieste cancellabili;
- **Motivazione:** Talvolta è necessario inoltrare richieste a oggetti senza conoscere nulla dell'operazione richiesta o del destinatario della richiesta. Il pattern Command permette agli oggetti dell'ambiente di inoltrare richieste a oggetti sconosciuti dell'applicazione trasformando la richiesta in un oggetto;
- **Applicabilità:** Il pattern Command può essere utilizzato nei seguenti casi:
 - Per parametrizzare gli oggetti rispetto a un'azione da compiere;
 - Per specificare, accordare ed eseguire le richieste in tempi diversi;
 - Per consentire l'annullamento di operazioni;
 - Per organizzare un sistema in operazioni d'alto livello a loro volta basate su operazioni primitive.

A.4.2 Strategy

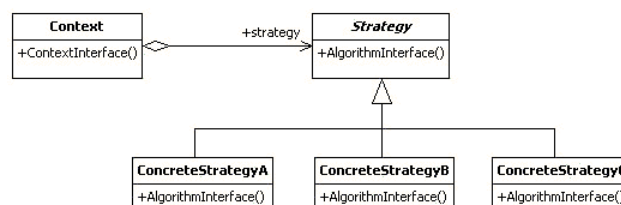


Figura 48: Diagramma del Design Pattern Strategy

- **Scopo:** Definire una famiglia di algoritmi, incapsularli e renderli intercambiabili. Permette agli algoritmi di variare indipendentemente dai client che ne fanno uso;



- **Motivazione:** Esistono molti algoritmi per risolvere un problema. Codificare statisticamente ognuno di questi algoritmi nelle classi che ne fanno richiesta non é auspicabile per svariati motivi. Si possono evitare questi problemi definendo delle classi che incapsulano svariati algoritmi chiamati Strategy;
- **Applicabilità:** Il pattern Strategy può essere utilizzato nei seguenti casi:
 - Molte classi correlate differiscono fra loro solo per il comportamento;
 - Sono necessarie più varianti di un algoritmo;
 - Un algoritmo usa una struttura dati che non dovrebbe essere resa nota ai client;
 - Una classe definisce molti comportamenti che compaiono all'interno di scelte condizionali multiple.



B Mockup interfaccia grafica

B.1 Finestra principale

La finestra principale di 3DMob contiene il visualizzatore interattivo della scena_G 3D (figura 49).

Le funzionalità di modifica della scena_G sono raggruppate in ordine logico in dei **QDock Widget** che permettono all'utente di personalizzare l'interfaccia.

In particolare l'utente può:

- Aprire un nuovo file da importare;
- Selezionare un oggetto o una luce della scena_G;
- Modificare l'oggetto selezionato applicando rotazioni, traslazioni, ridimensionamenti;
- Modificare la luce selezionata applicando rotazioni o modifiche alle sue caratteristiche come lucentezza ed emissione;
- Aggiungere una nuova luce, scelta tra omnidirezionale e spotlight_G;
- Aggiungere un nuovo materiale all'oggetto selezionato;
- Navigare la scena_G.

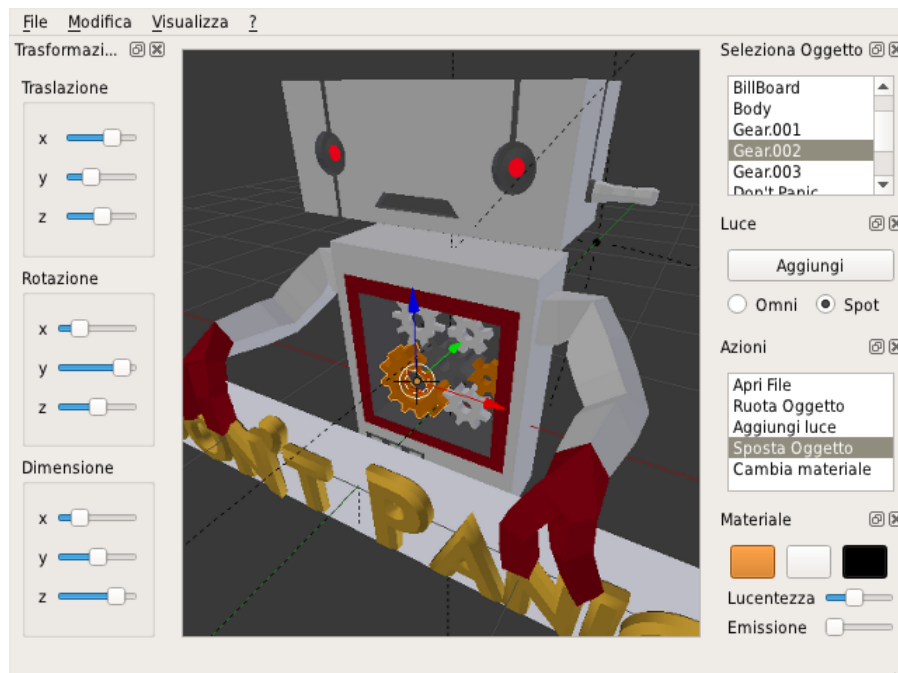


Figura 49: Mockup della finestra principale dell'applicazione 3DMob

Alcune funzionalità saranno raggiungibili anche da un menu a tendina presente nella finestra principale (figura 50).

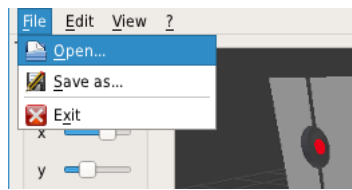


Figura 50: Mockup del menu file

B.2 Finestra di configurazione

Le impostazioni di sistema sono accessibili tramite l'apposita finestra (figura 51). Tale finestra permette di configurare le impostazioni di sistema scegliendo se salvare la scena_G esportando i dati in virgola mobile con precisione singola o doppia e impostando i limiti di esportazione che, di default, sono impostati su valori compatibili con iPhone 4S.

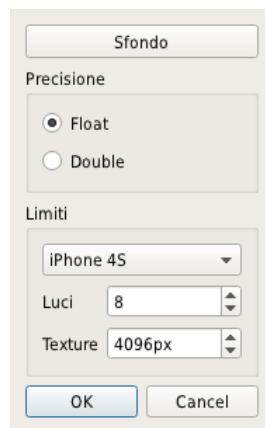


Figura 51: Mockup della finestra per la configurazione delle impostazioni

B.3 Finestra di selezione file da importare

Quando un utente decide di importare un file, viene visualizzata una finestra (figura 52) che permette di navigare il file system e di selezionare un file. L'utente può selezionare il file navigando nel file system e facendo un doppio click sul file desiderato o scrivendo per intero indirizzo e nome del file o solo nome del file se presente della directory corrente visualizzata nella finestra.

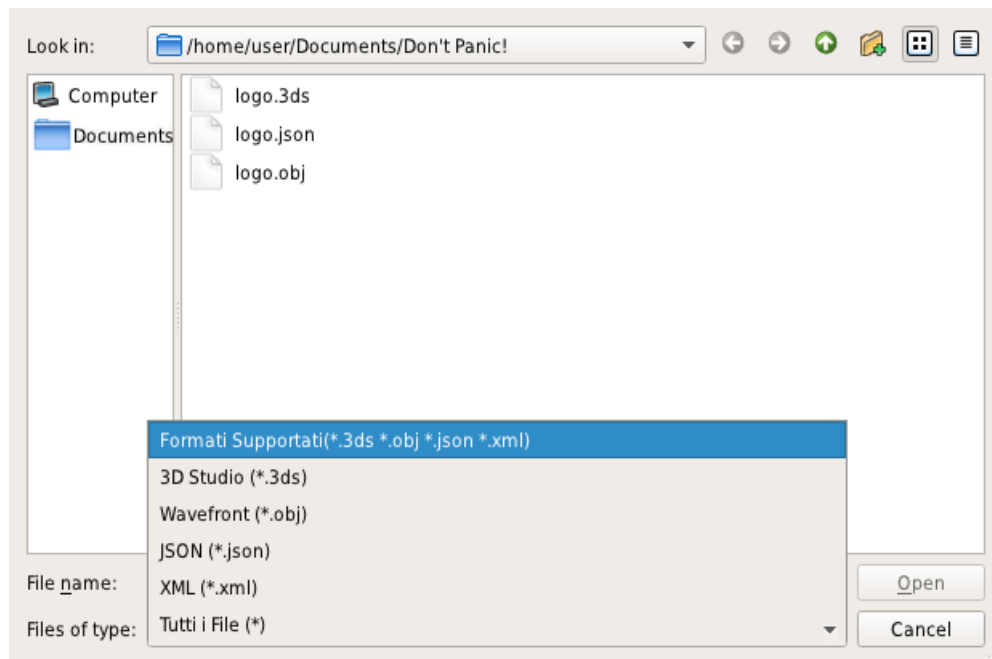


Figura 52: Mockup della finestra usata per selezionare il file in fase di importazione

B.4 Finestra di esportazione file

Quando un utente decide di esportare un file, viene visualizzata una finestra (figura 53) che permette di navigare il file system e di inserire il nome del file in cui verrà esportata la scena_G. L'utente può scrivere per intero indirizzo e nome del file o solo nome del file se presente della directory corrente visualizzata nella finestra o selezionare un file esistente se lo si vuole sovrascrivere.

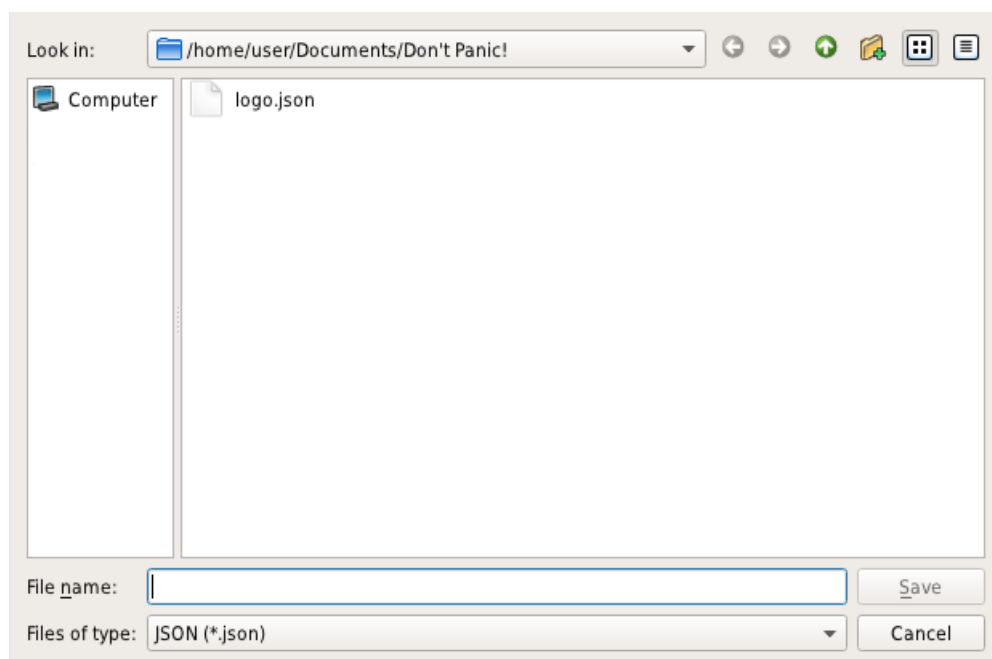


Figura 53: Mockup della finestra usata per selezionare il file in fase di esportazione

B.5 Finestra di aiuto

L'utente può accedere al sistema di help utilizzando l'apposita finestra (figura 54) che fornisce funzionalità di visualizzazione, ricerca e navigazione dei contenuti del manuale.

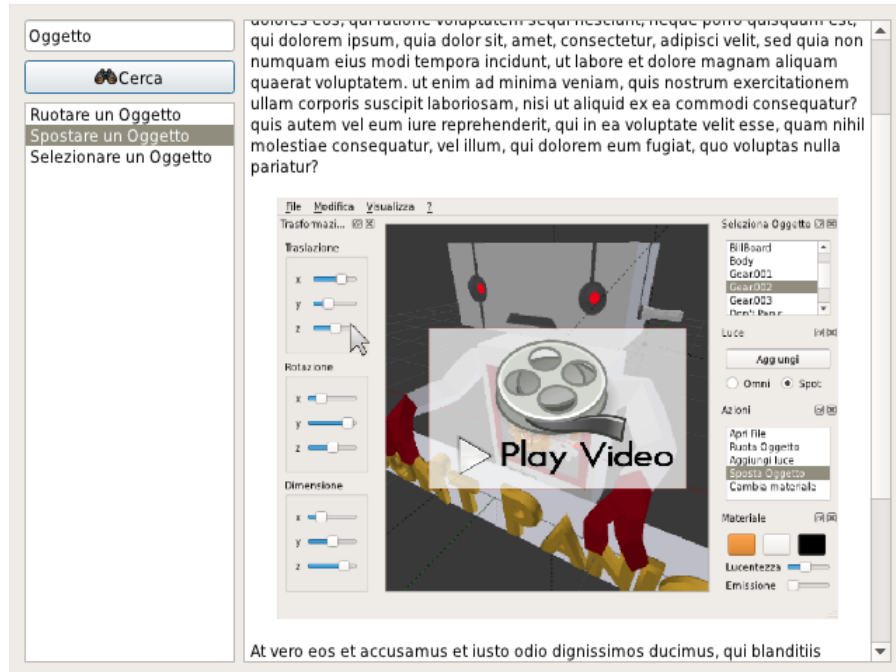


Figura 54: Mockup della finestra di aiuto

B.6 Finestra di informazioni sul sistema

L'utente può accedere alle informazioni di sistema che vengono visualizzate tramite l'apposita finestra (figura 55) che fornisce informazioni sul sistema quali ad esempio la licenza e la versione.



Figura 55: Mockup della finestra di informazioni di sistema

B.7 Finestra dei messaggi di sistema

L'utente riceve messaggi informativi dal sistema a mezzo di una finestra di popup_G (figura 56).



Questi messaggi riguardano lo stato del sistema, dei file aperti e delle conversioni. Nessuno di questi messaggi è di natura bloccante per il sistema. Tra i messaggi che potrebbero essere segnalati troviamo ad esempio la segnalazione del caricamento di un file con una scena_G che eccede i limiti impostati nel sistema.

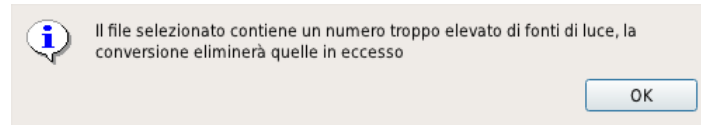


Figura 56: Mockup della finestra di messaggio di sistema

B.8 Finestra degli errori di sistema

L'utente riceve messaggi di errori dal sistema a mezzo di una finestra di popup_G (figura 57).

Questi messaggi di errore del sistema sono di natura bloccante in relazione all'azione appena eseguita dall'utente o attività del sistema in corso. Tra gli errori che potrebbero essere segnalati troviamo l'errore di formato del file da aprire o l'errore nel salvataggio.

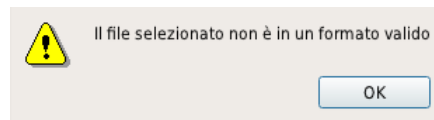


Figura 57: Mockup della finestra di errore di sistema