OO - Analyse
mario world wars

Projekt: **mario world wars**

Voraussetzung: Pflichtenheft

Autoren: *Thomas Vogel, Michael Palmer, Florian Oeser*

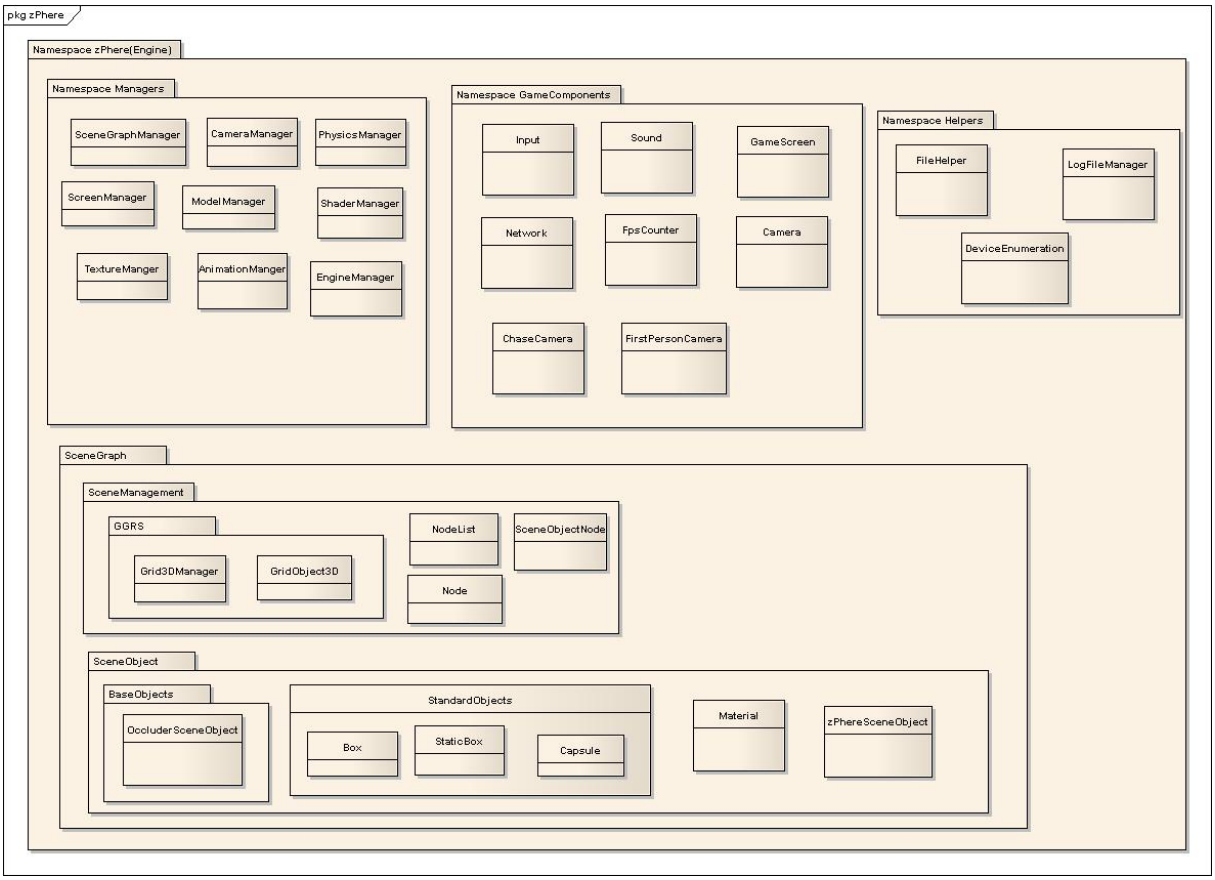
Kontakt: **mario@link2wall.de**

Letzte Änderung: **14.10.08**

Inhaltsverzeichnis

1 Paketdiagramm

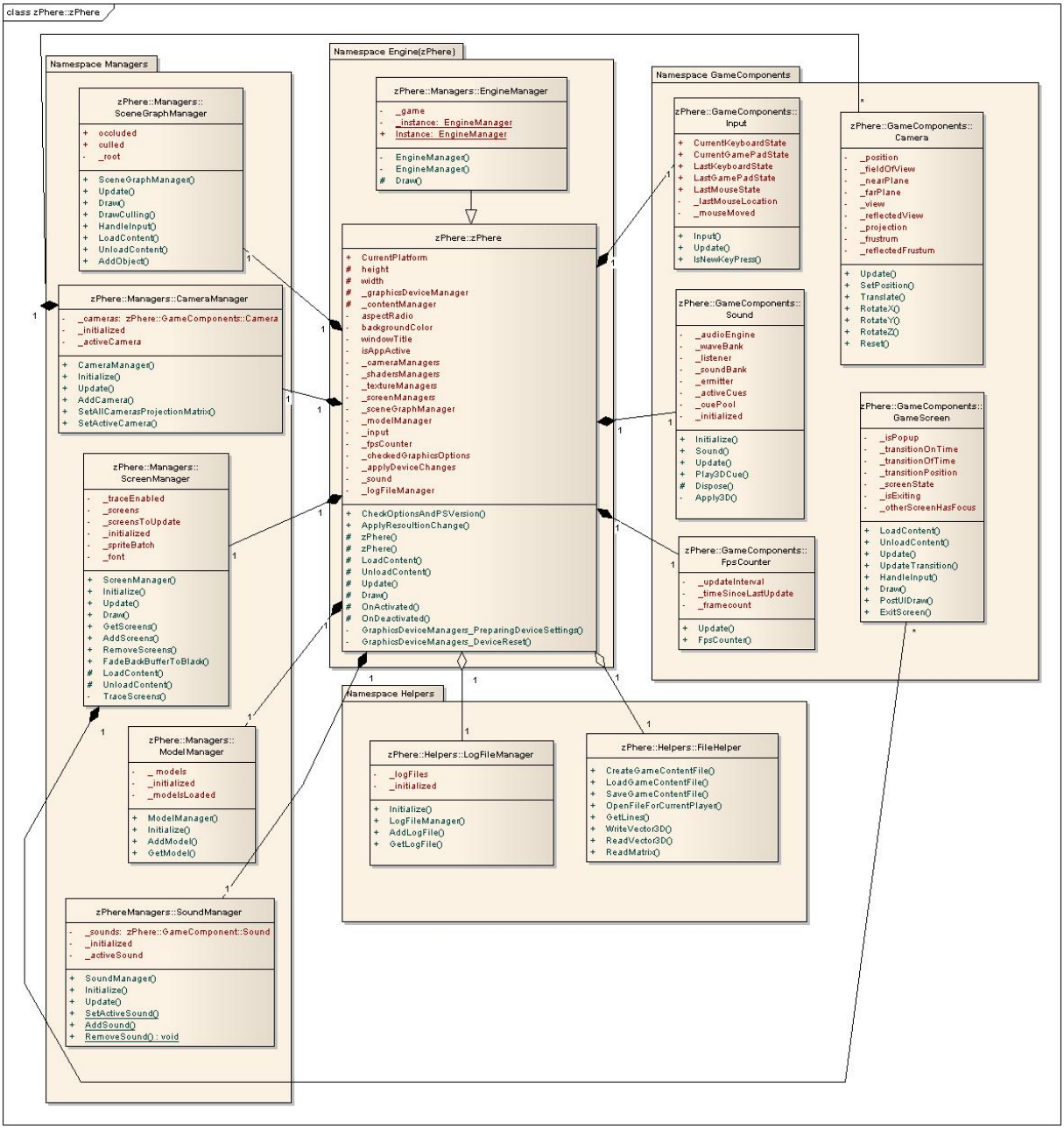
Das hier gezeigt Paketdiagramm zeigt alle Klassen die in unserer Engine zum Tragen kommen werden.



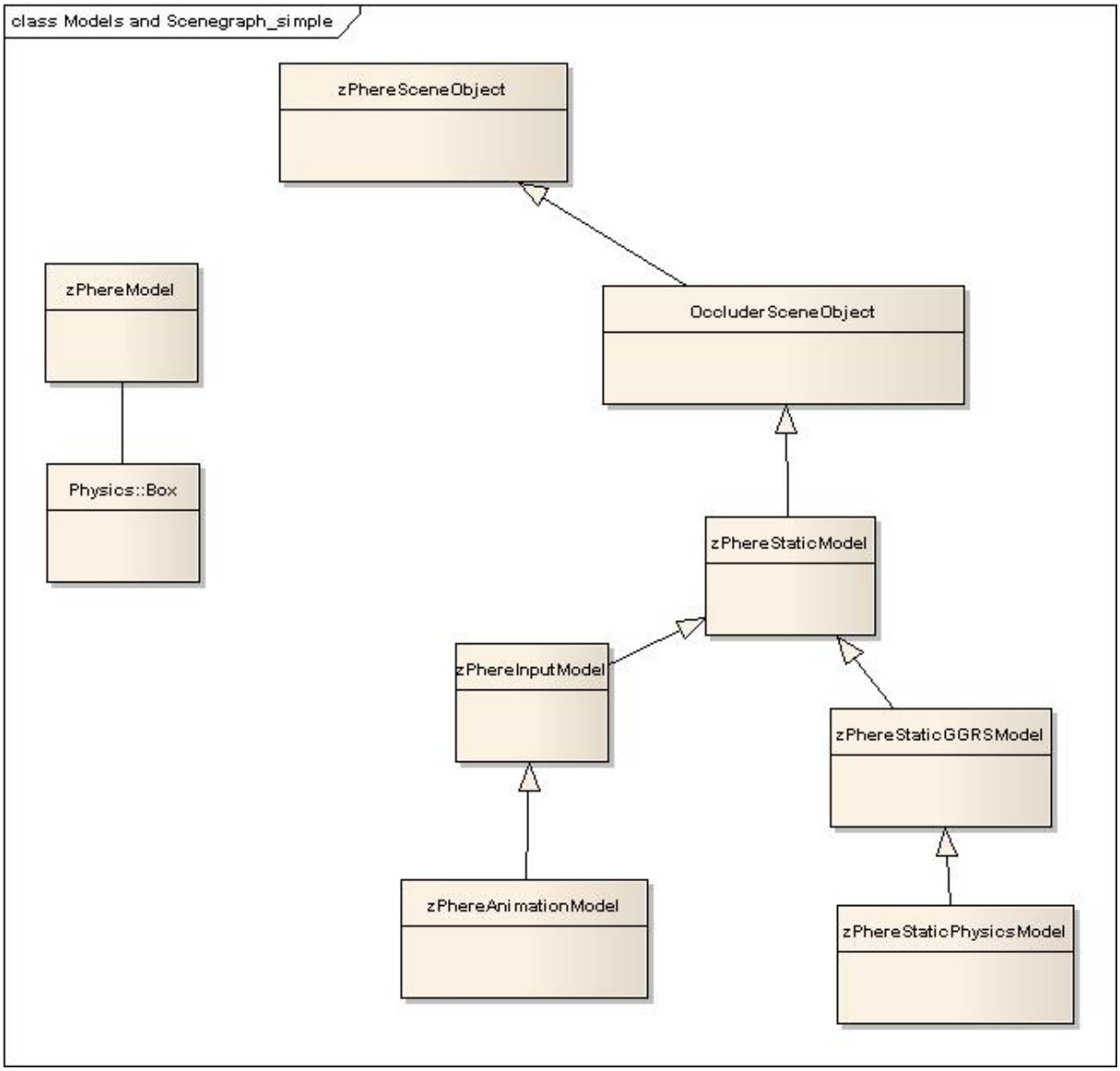
2 Klassendiagramm

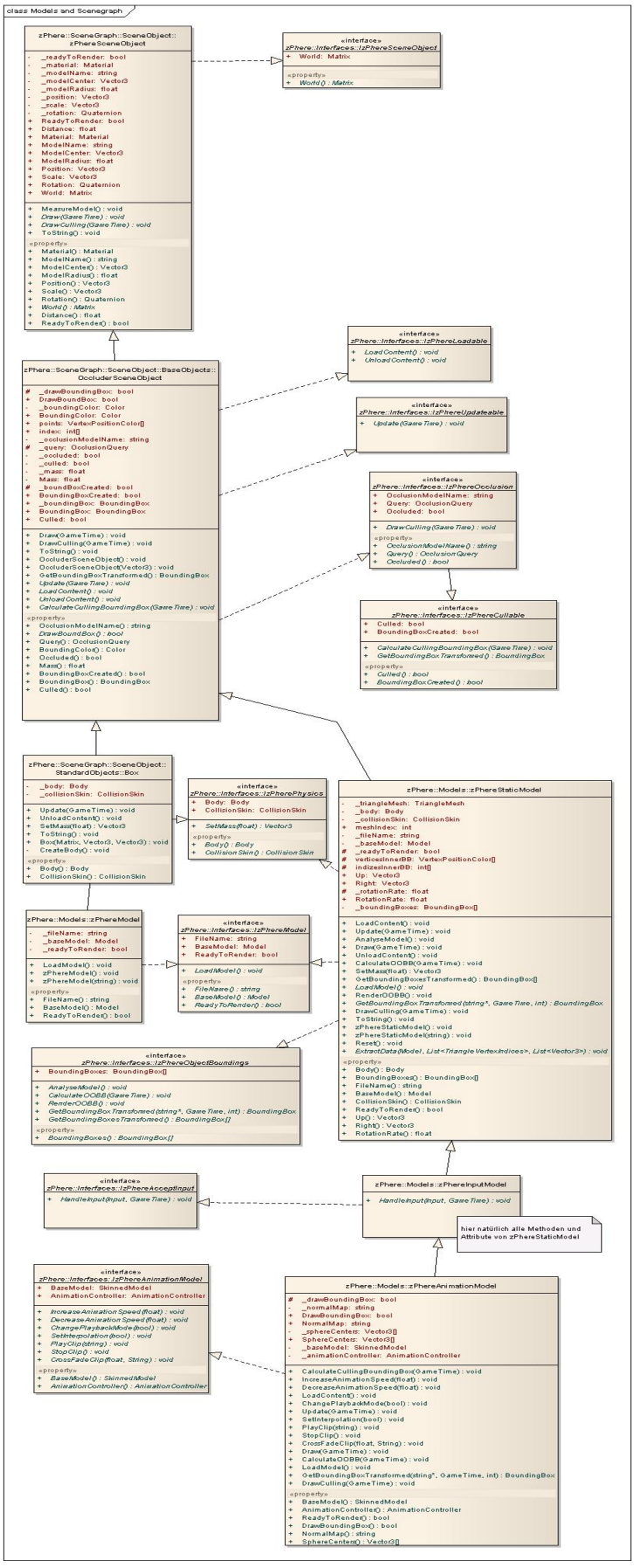
2.1 Engine

Dieses Klassendiagramm zeigt einen detaillierten Aufbau unserer Engine mit dem Namen 'zPhere'. Die Klassennamen entsprechen den Namespacenamen die im Sourcecode verwendet werden. Auf die Darstellung des Textur- und Shadermanagers haben wir verzichtet da, beide exakt dem Modelmanager entsprechen. Ausserdem haben wir keine Darstellung zu der Klasse 'DeviceEnumeration' und 'Network', da wir zum jetzigen Entwicklungsstand das Zusammenspiel mit dem Gesamtsystem nicht abschätzen können.

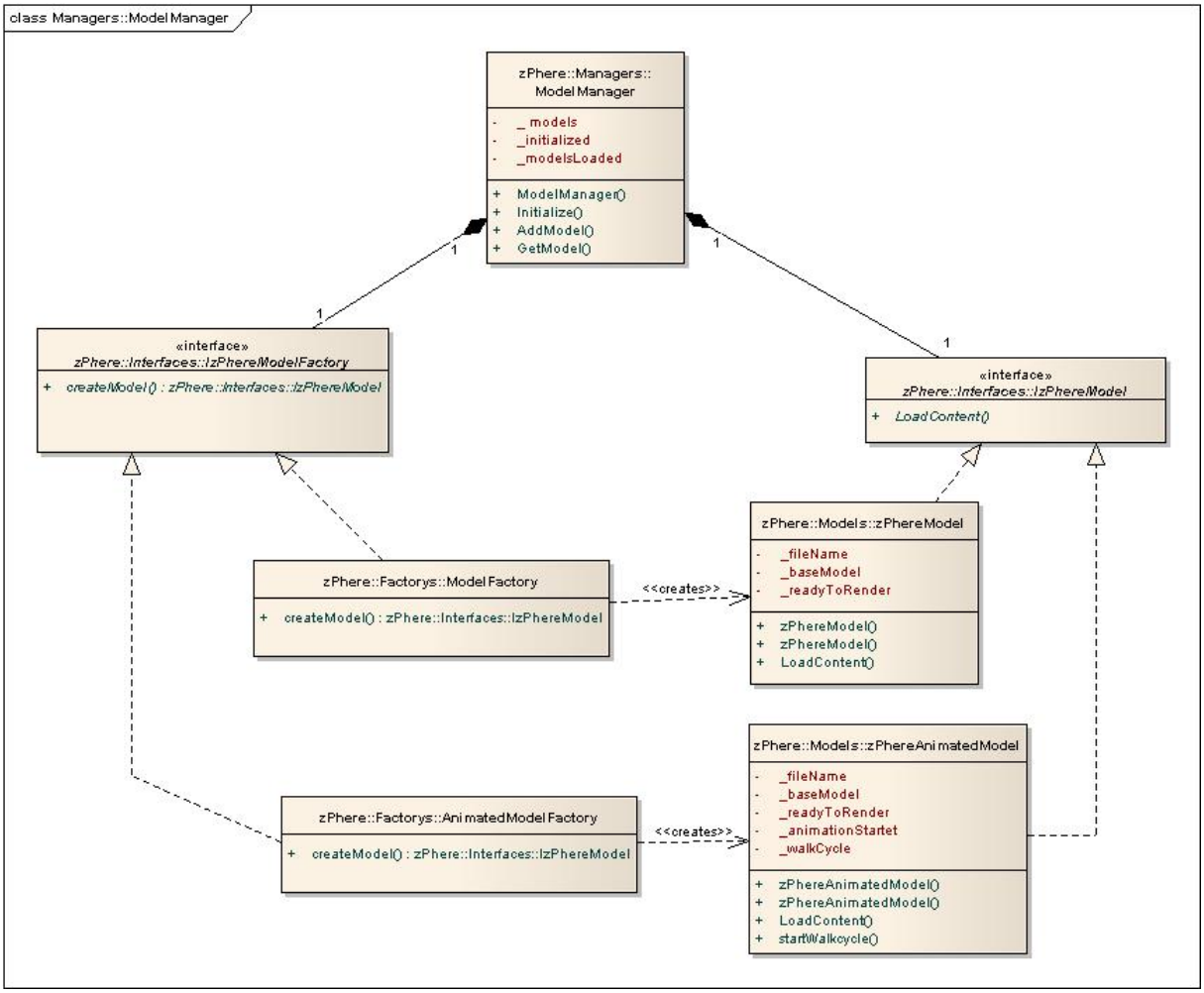


2.1.1 Der Scenegraph

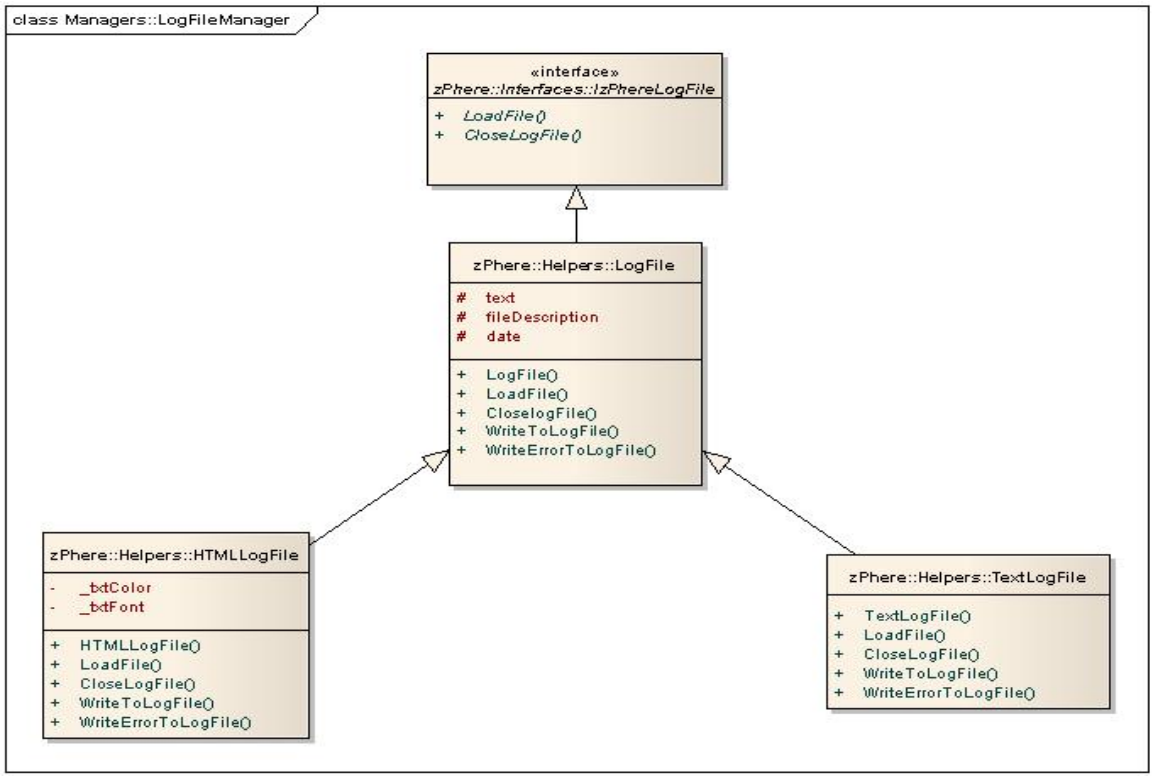



class Models and Scenegraph
zPhere::SceneGraph::SceneObject::zPhereSceneObject
«interface» zPhere::Interfaces::IzPhereSceneObject
«interface» zPhere::Interfaces::IzPhereLoadable
zPhere::SceneGraph::SceneObject::BaseObjects::OccluderSceneObject
«interface» zPhere::Interfaces::IzPhereUpdateable
«interface» zPhere::Interfaces::IzPhereOcclusion
«interface» zPhere::Interfaces::IzPhereCullable
zPhere::SceneGraph::SceneObject::StandardObjects::Box
«interface» zPhere::Interfaces::IzPherePhysics
zPhere::Models::zPhereStaticModel
zPhere::Models::zPhereModel
«interface» zPhere::Interfaces::IzPhereModel
«interface» zPhere::Interfaces::IzPhereObjectBoundings
«interface» zPhere::Interfaces::IzPhereAcceptInput
zPhere::Models::zPhereInputModel
hier natürlich alle Methoden und Attribute von zPhereStaticModel
«interface» zPhere::Interfaces::IzPhereAnimationModel
zPhere::Models::zPhereAnimationModel

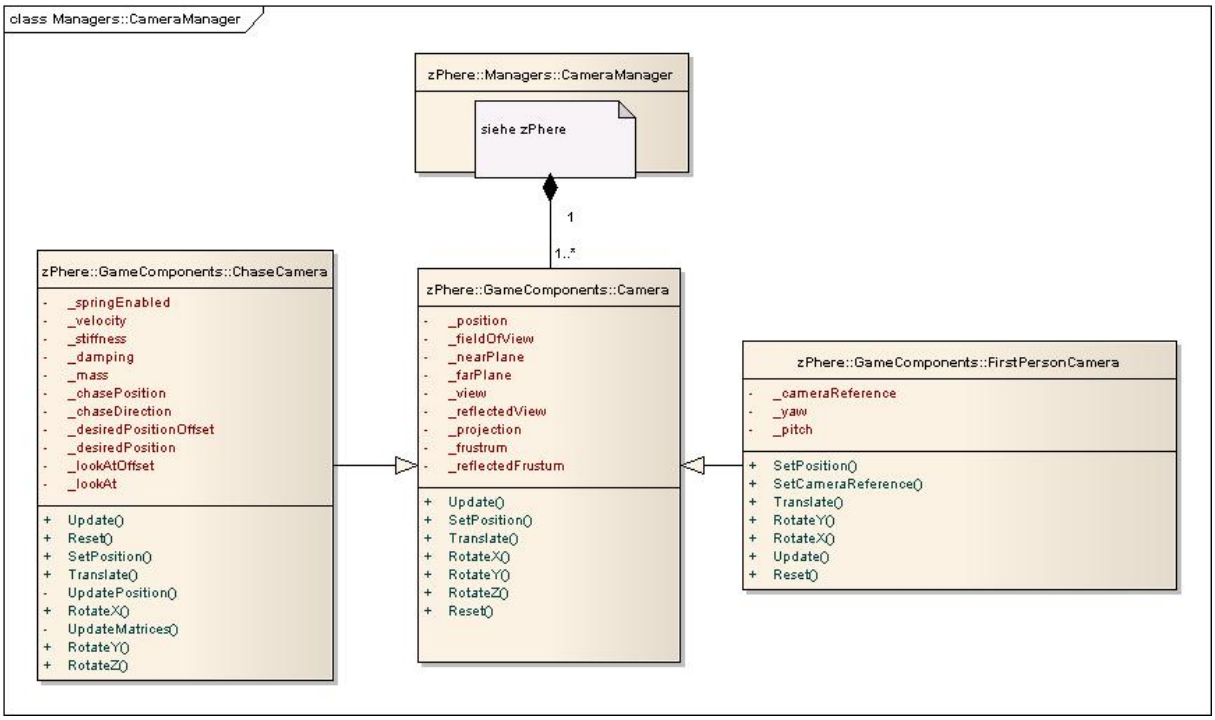
2.1.2 ModelManager



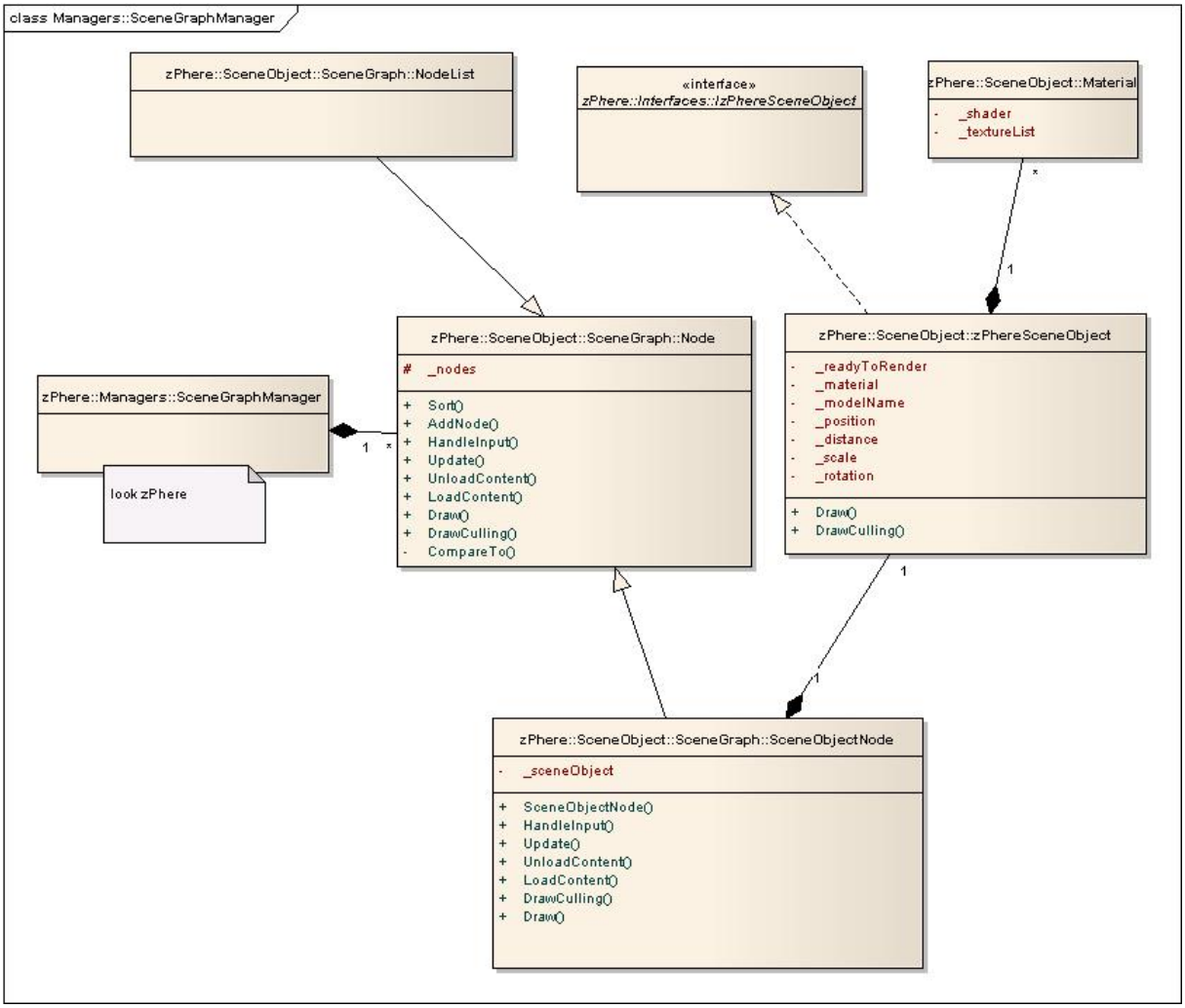
2.1.3 LogFileManager



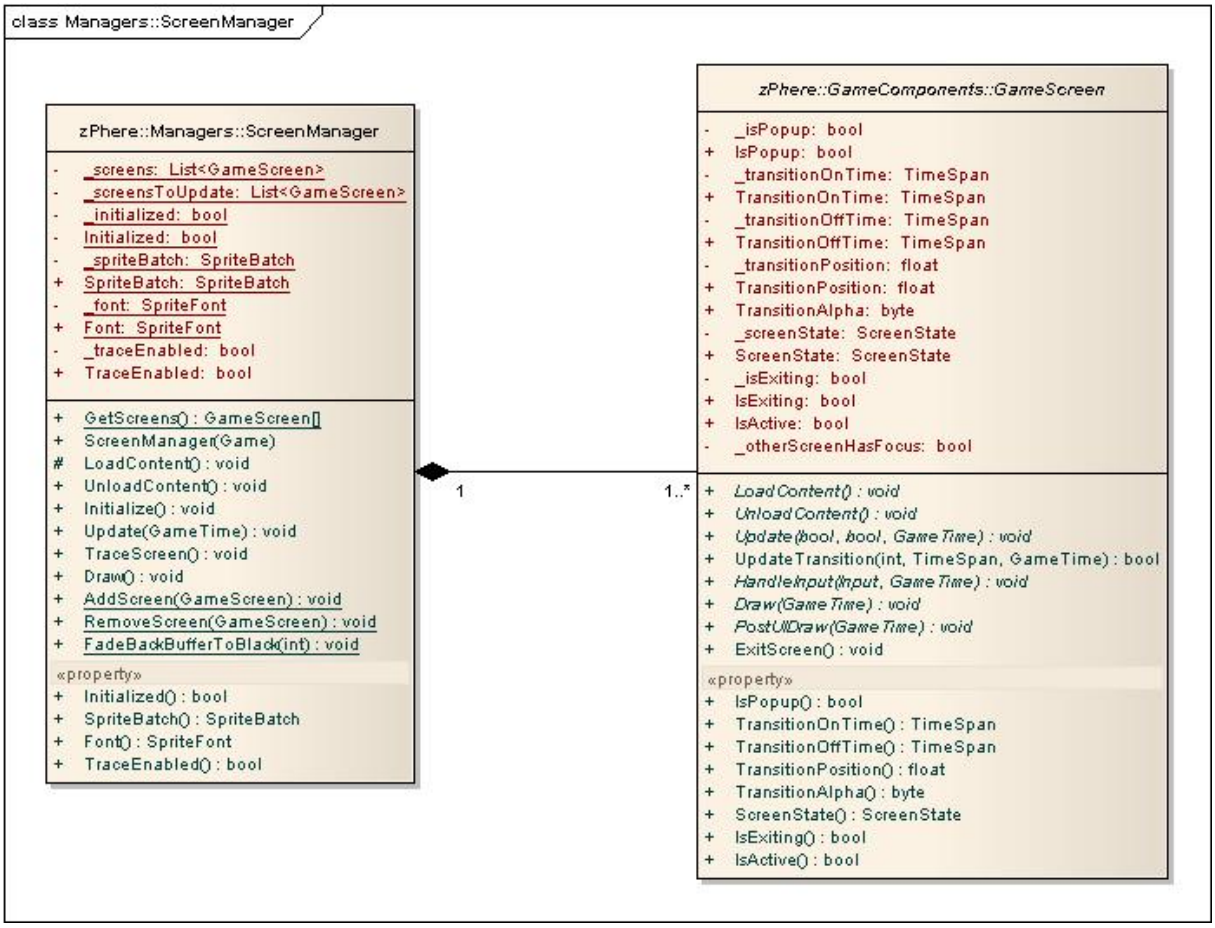
2.1.4 CameraManager



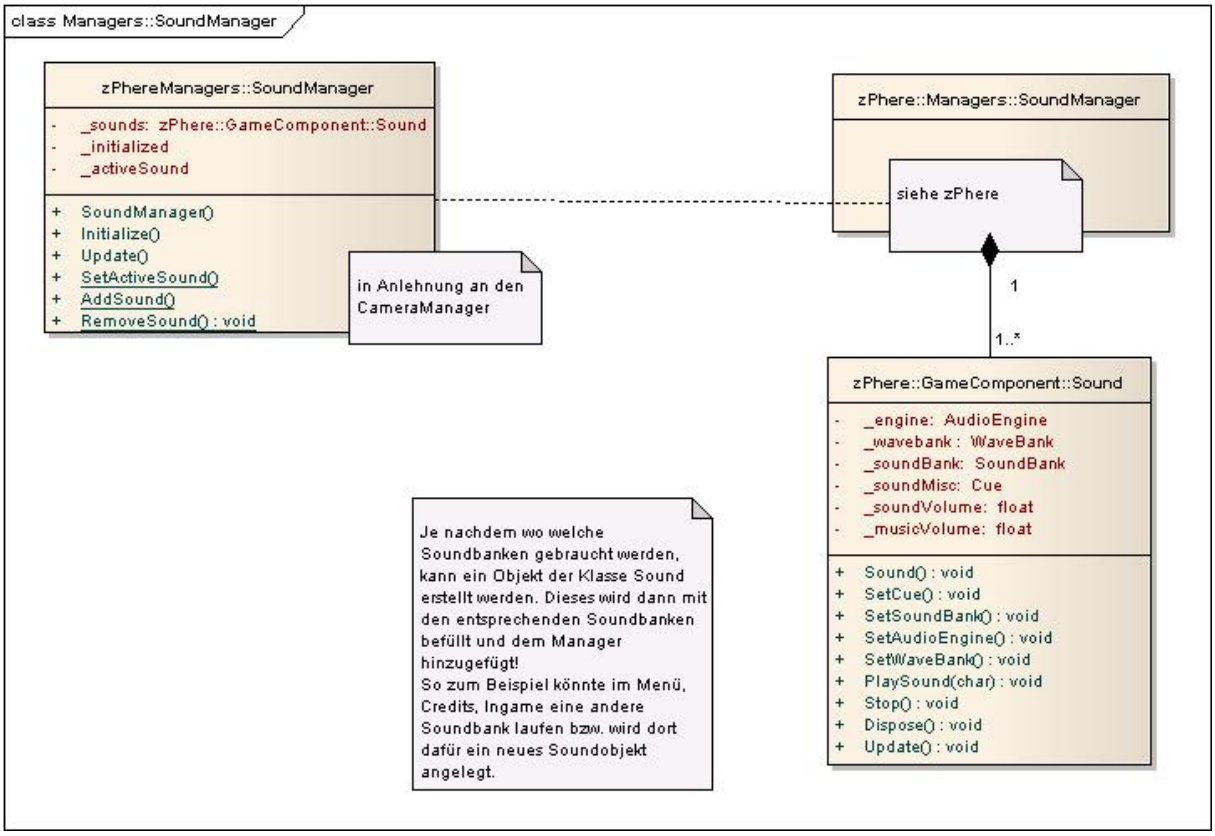
2.1.5 SceneGraphManager



2.1.6 ScreenManager



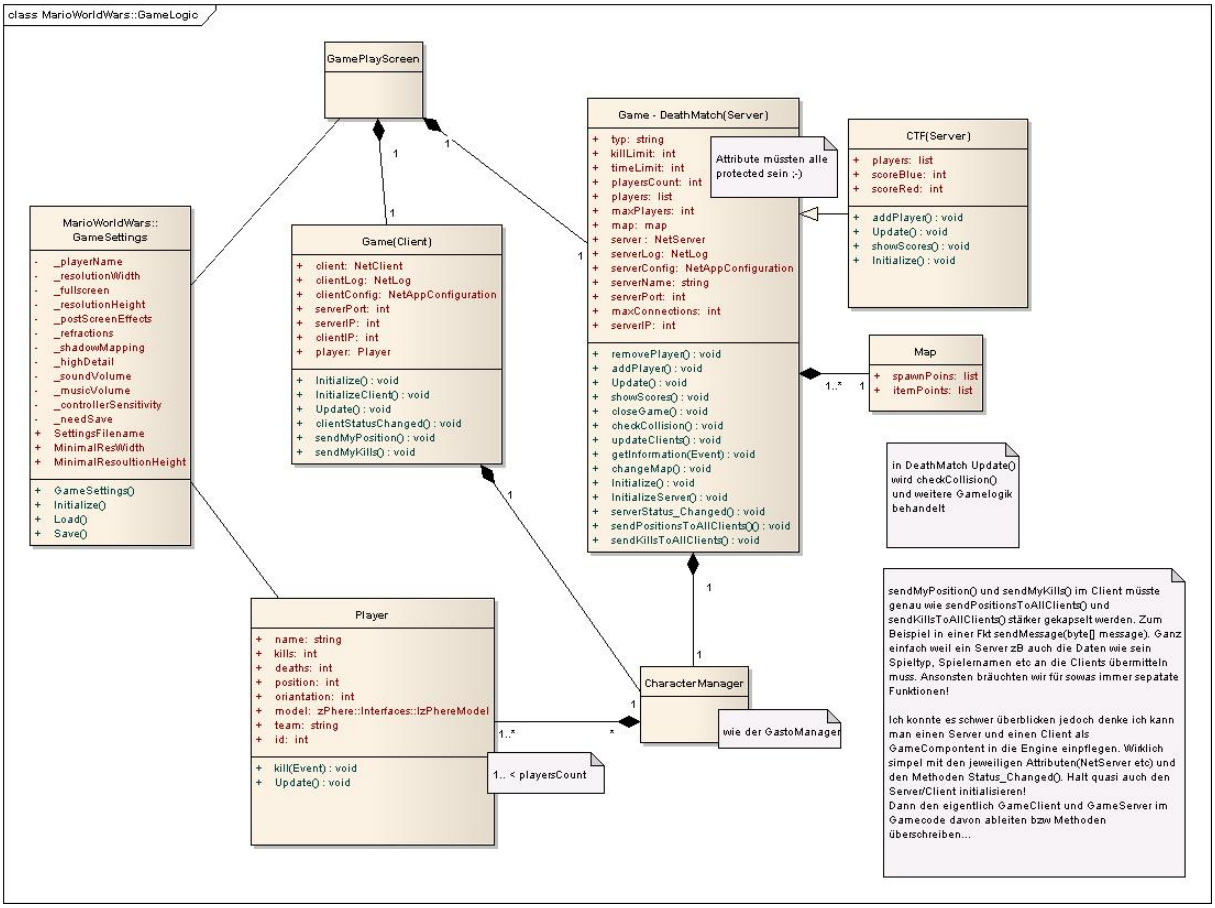
2.1.7 SoundManager



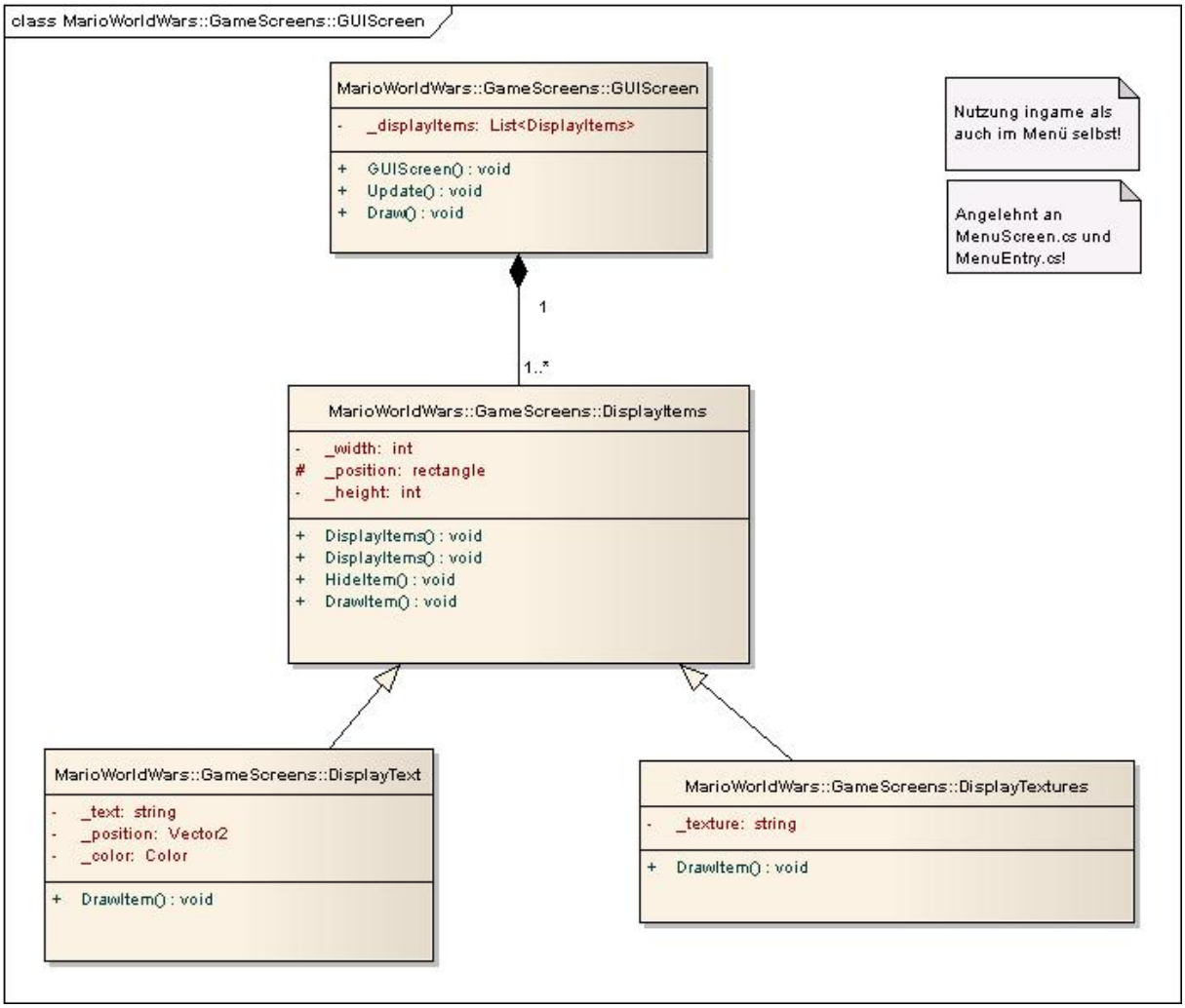
2.2 Das Spiel

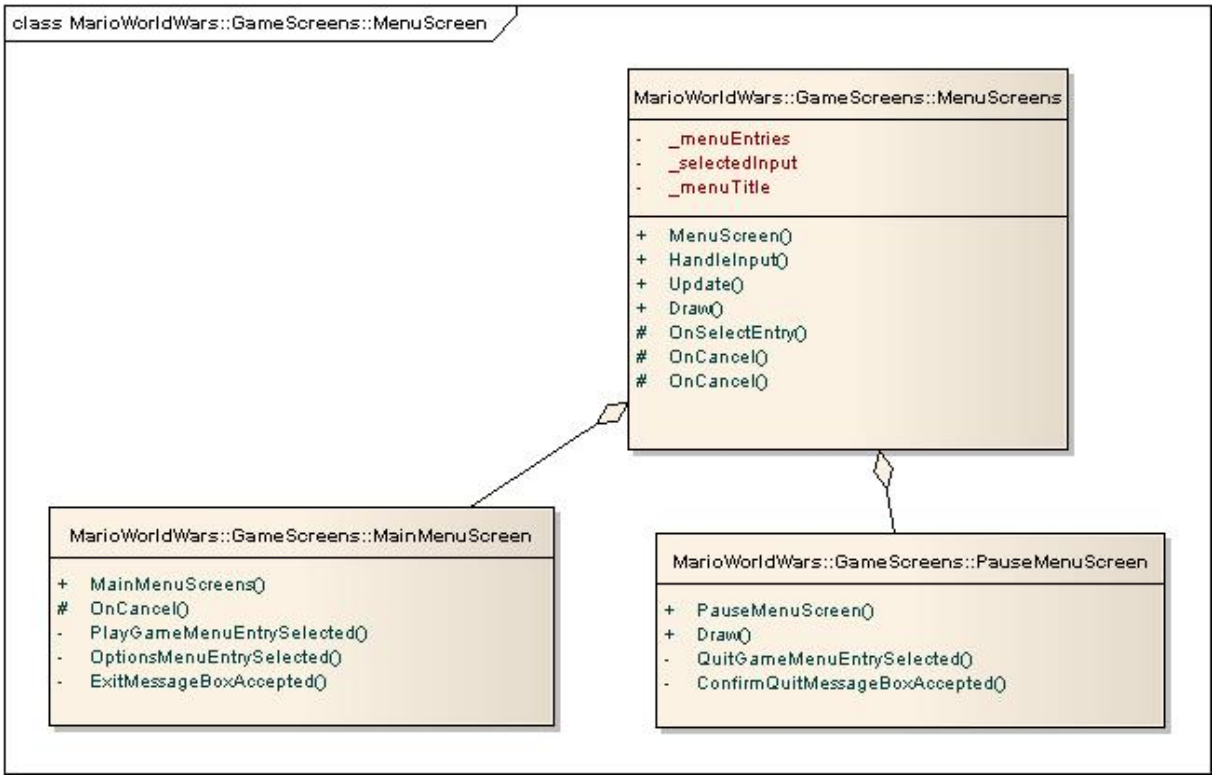
Hier können wir Ihnen ebenfalls nur einen sehr groben Umriss der Spielimplementierung geben, da sich auch hier vieles noch in Entwicklung befinden und auch erstmal stark von der Engineimplementierung abhängig ist.

2.2.1 Spielimplementierung



2.2.2 Die GameScreens

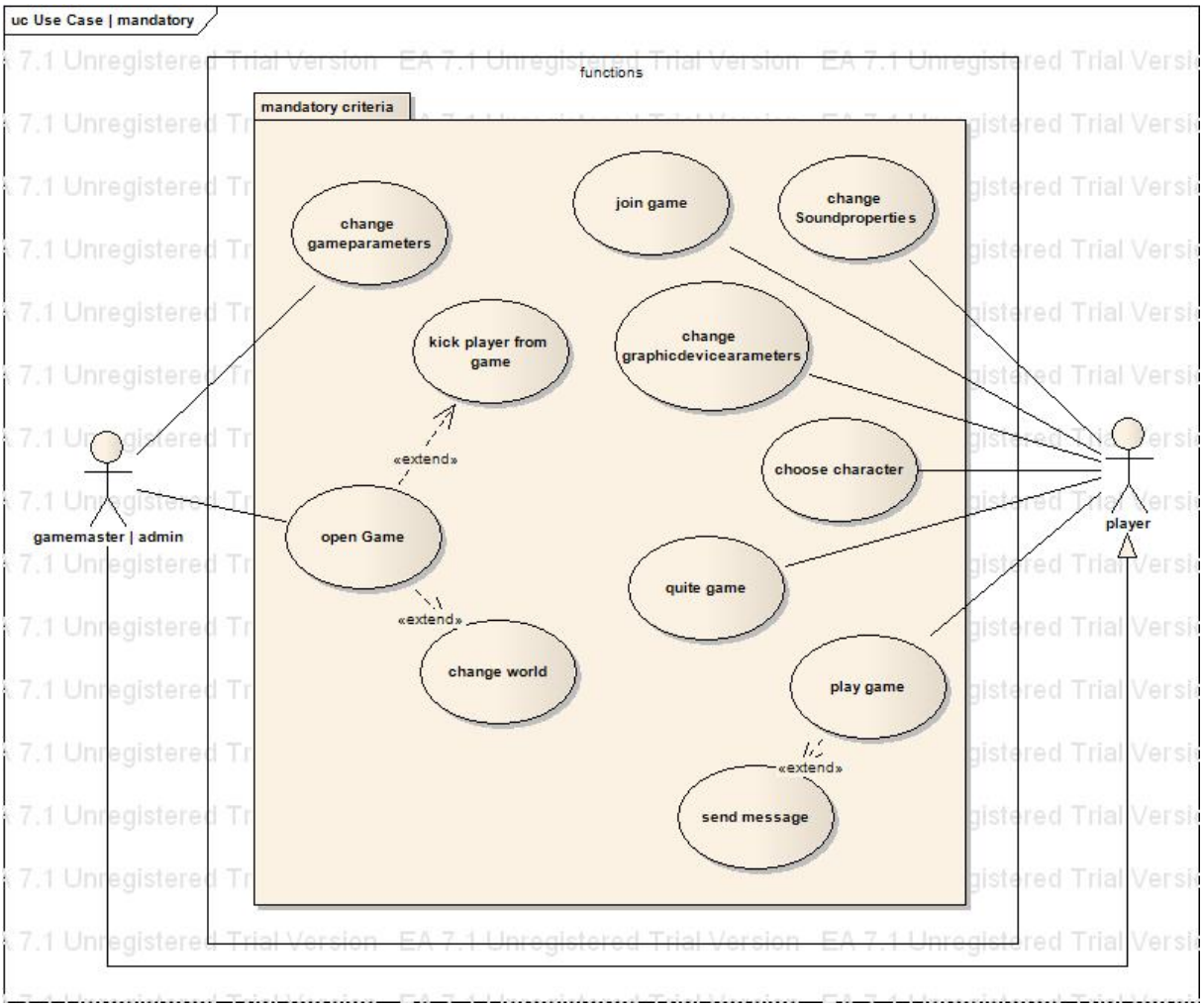

class MarioWorldWars::GameScreens::GUIScreen
MarioWorldWars::GameScreens::GUIScreen
- _displayItems: List<DisplayItems>
+ GUIScreen() : void
+ Update() : void
+ Draw() : void
Nutzung ingame als auch im Menü selbst!
Angelehnt an MenuScreen.cs und MenuEntry.cs!
1
1..*
MarioWorldWars::GameScreens::DisplayItems
- _width: int
_position: rectangle
- _height: int
+ DisplayItems() : void
+ DisplayItems() : void
+ HideItem() : void
+ DrawItem() : void
MarioWorldWars::GameScreens::DisplayText
- _text: string
- _position: Vector2
- _color: Color
+ DrawItem() : void
MarioWorldWars::GameScreens::DisplayTextures
- _texture: string
+ DrawItem() : void


class MarioWorldWars::GameScreens::MenuScreen
MarioWorldWars::GameScreens::MenuScreens
- _menuEntries
- _selectedInput
- _menuTitle
+ MenuScreen()
+ HandleInput()
+ Update()
+ Draw()
# OnSelectEntry()
# OnCancel()
# OnCancel()
MarioWorldWars::GameScreens::MainMenuScreen
+ MainMenuScreens()
# OnCancel()
- PlayGameMenuEntrySelected()
- OptionsMenuEntrySelected()
- ExitMessageBoxAccepted()
MarioWorldWars::GameScreens::PauseMenuScreen
+ PauseMenuScreen()
+ Draw()
- QuitGameMenuEntrySelected()
- ConfirmQuitMessageBoxAccepted()

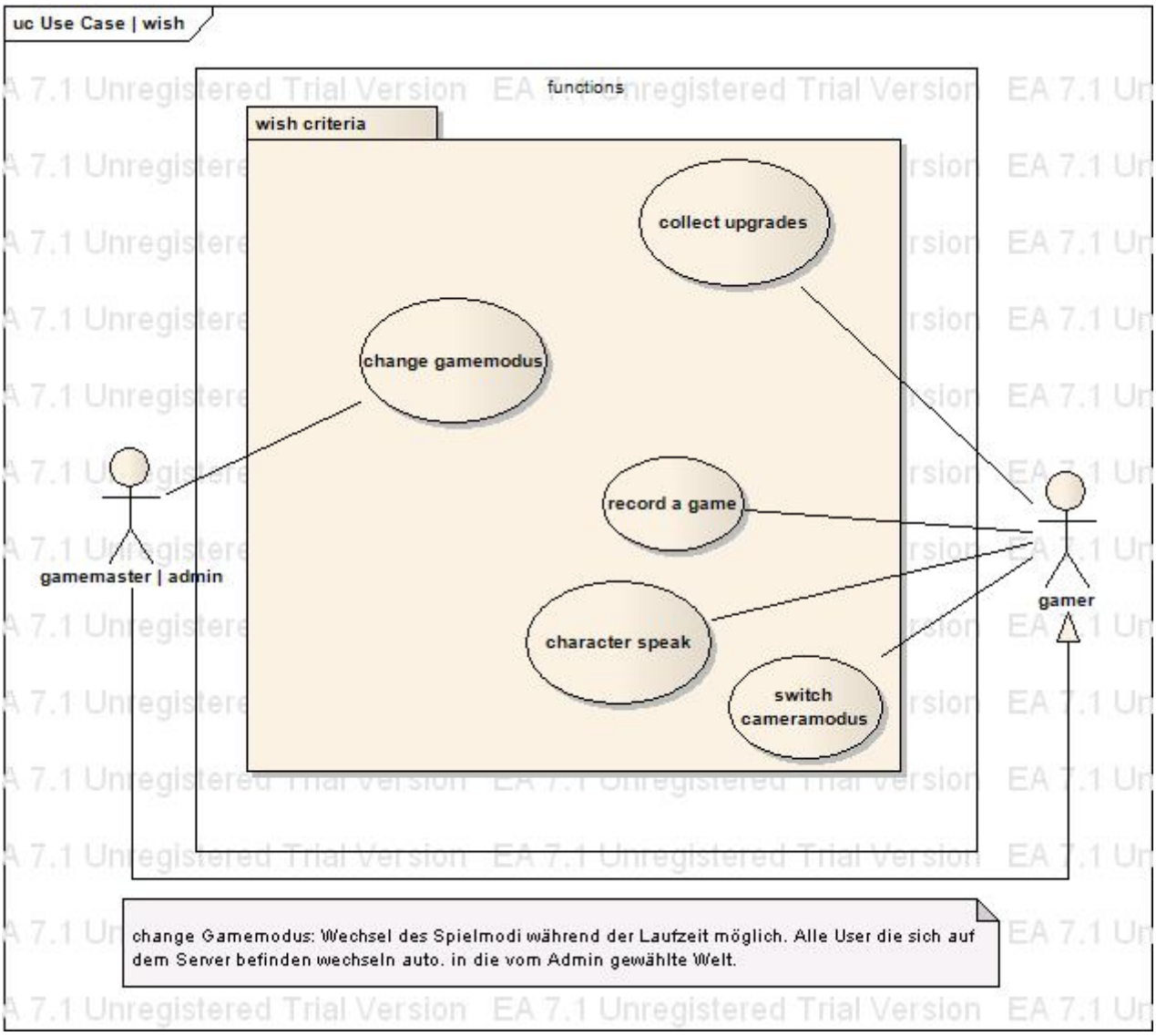
2 UseCase Diagramme

Die Geschäftsprozesse werden in Form von UseCase-Diagrammen dargestellt. Es existieren zwei Benutzer: Admin und Player

2.1 Musskriterien



2.2 Wunschkriterien



3 Sequenzdiagramme

Die zwei Diagramme zeigen den Ablauf was nötig ist um ein einfaches Model auf dem Bildschirm zeichnen zu können. Das erste Diagramm verdeutlicht die Vorgänge der Engineinitialisierung während das zweite Diagramm zeigt wie die initialisierten Objekte zum eigentlichen zeichnen verwendet werden.

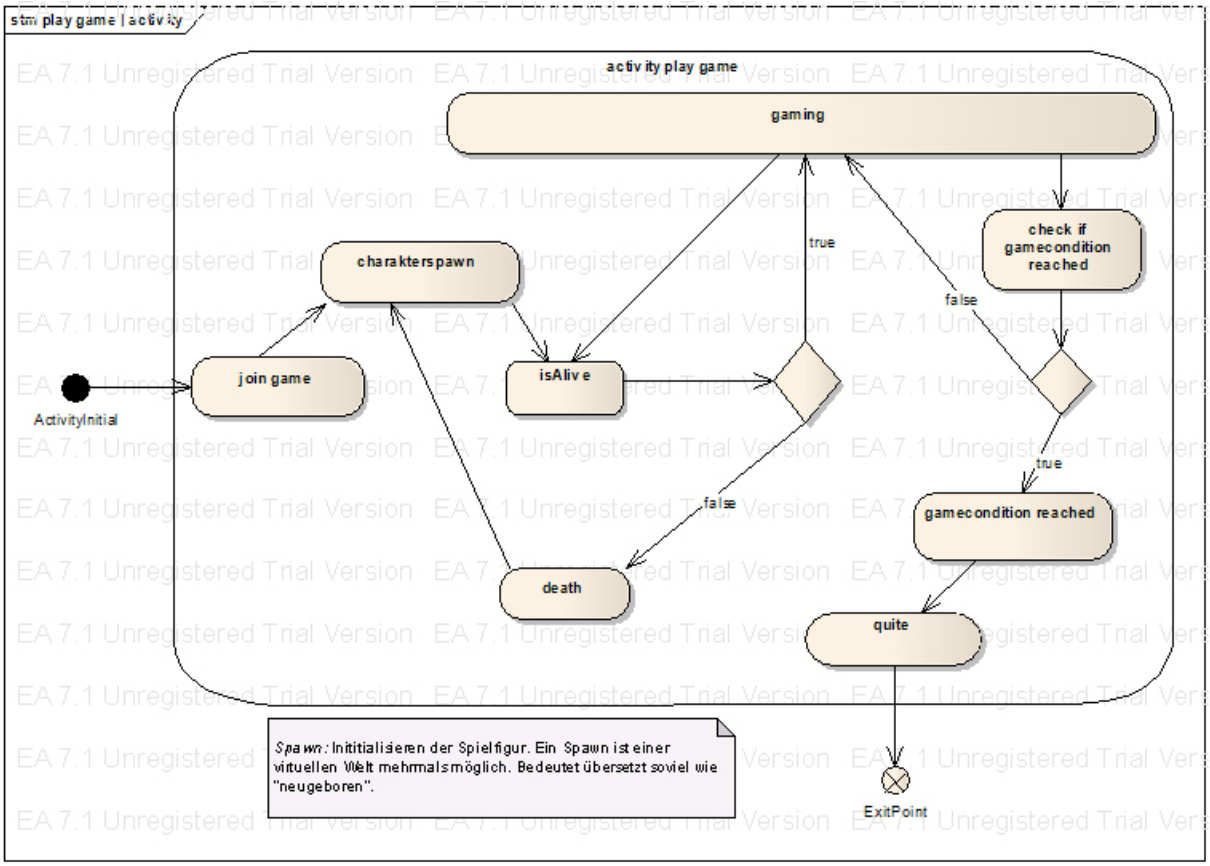
Da die Sequenzdiagramme sehr groß sind, liefern wir diese als Bilddateien mit. Das erste Diagramm ist als 'Engineinitialize' und das zweite als 'renderModel' bezeichnet.

4 Aktivitätsdiagramme

4.1 Spiel spielen

Dieses Aktivitätsdiagramm beschreibt den Lebenszyklus innerhalb der Anwendung.

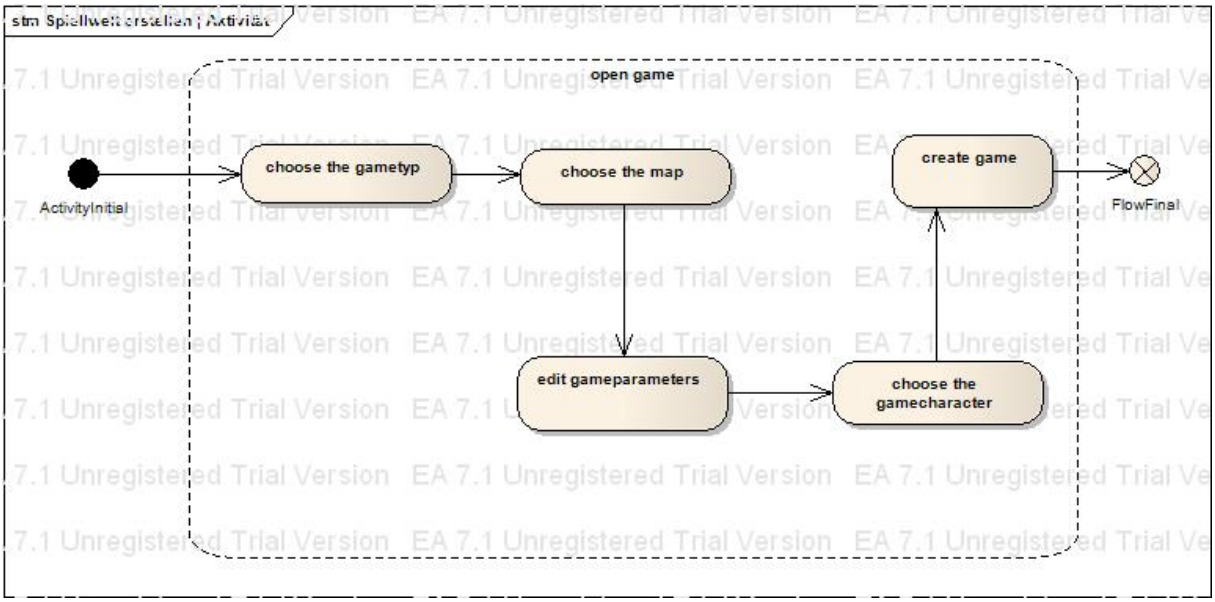
Der Benutzer tritt einem offenen Spiel bei. Die Spielfigur *spawnt* in der Welt. Der *gamecycle* beschreibt die andauernde Abfrage ob die Spielfigur des Benutzers noch am Leben ist. Neben läufig findet dauerhaft eine Abfrage statt ob das vorher festgelegte Spielziel erreicht wurde. Die Aktivität 'gaming' umfasst alle Möglichkeiten(springen, laufen, Gegner besiegen) während des Spiels. Die Aktivität 'is Alive' stellt das Leben in der virtuellen Welt dar. Wenn ein anderer Spieler den „virtuellen Benutzer" tötet wird 'isAlive' auf false gesetzt und der Benutzer landet in der Aktivität 'death'. Nach dieser Aktivität wird der Charakter neu initialisiert(*characterspawn*) und der *lifecycle* läuft wieder ab.



4.2 Spielwelt erstellen

Das Aktivitätsdiagramm stellt die einzelnen Aktivitäten für die Eröffnung eines Spieles dar.

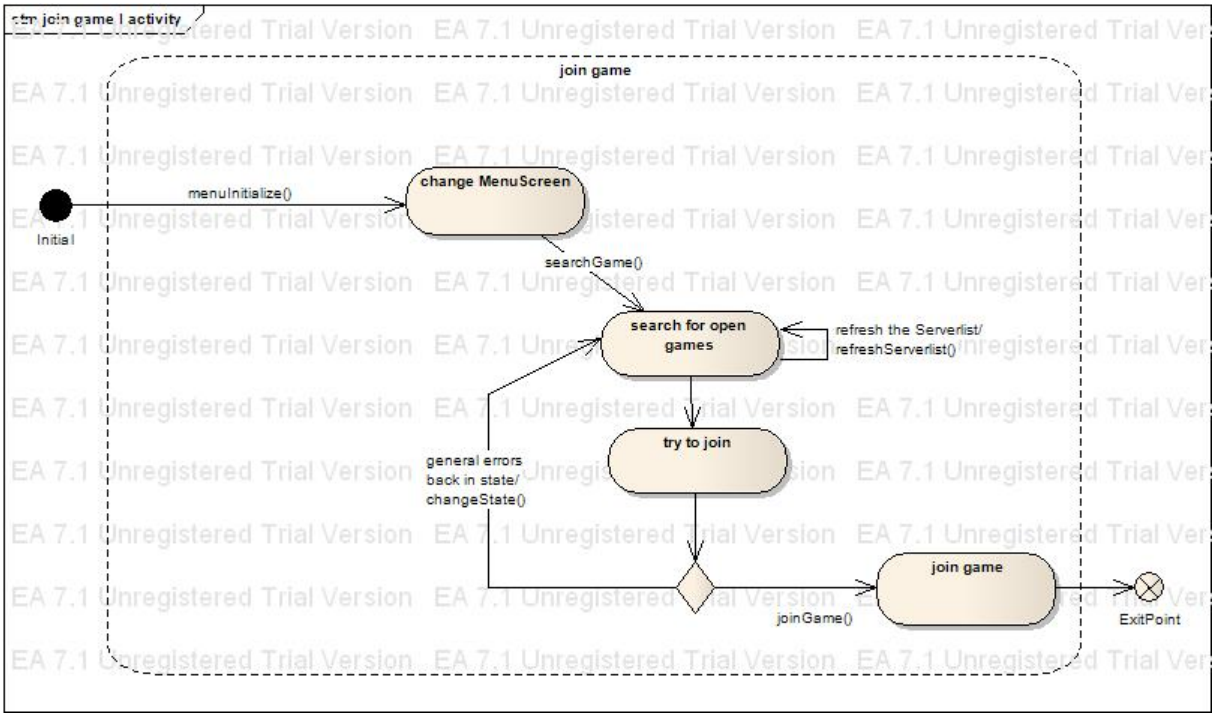
Bei der Erstellung einer Spielsession muss der Benutzer mehrere Parameter auswählen. Es muss ein Spieltyp festgelegt werden, eine Spielwelt und Spielparameter(Spielzeit, Ziel). Nachdem die Parameter für die „Spielwelt" festgelegt wurden, wählt der Benutzer seinen eigenen Avatar aus und erstellt daraufhin das Spiel.



4.3 Spiel beitreten

Dieses Aktivitätsdiagramm stellt das beitreten eines Spiels dar.

Wenn der Benutzer nach einem bereits offenen Spiel sucht kann er jederzeit die Serverliste aktualisieren. Angenommen er findet ein offenes Spiel probiert er dieser Partie beizutreten. Entweder hat die Partie die maximale Spieleranzahl bereits erreicht oder ein reibungsloser Verbindungsaufbau findet statt und er befindet sich in der gewünschten Partie.

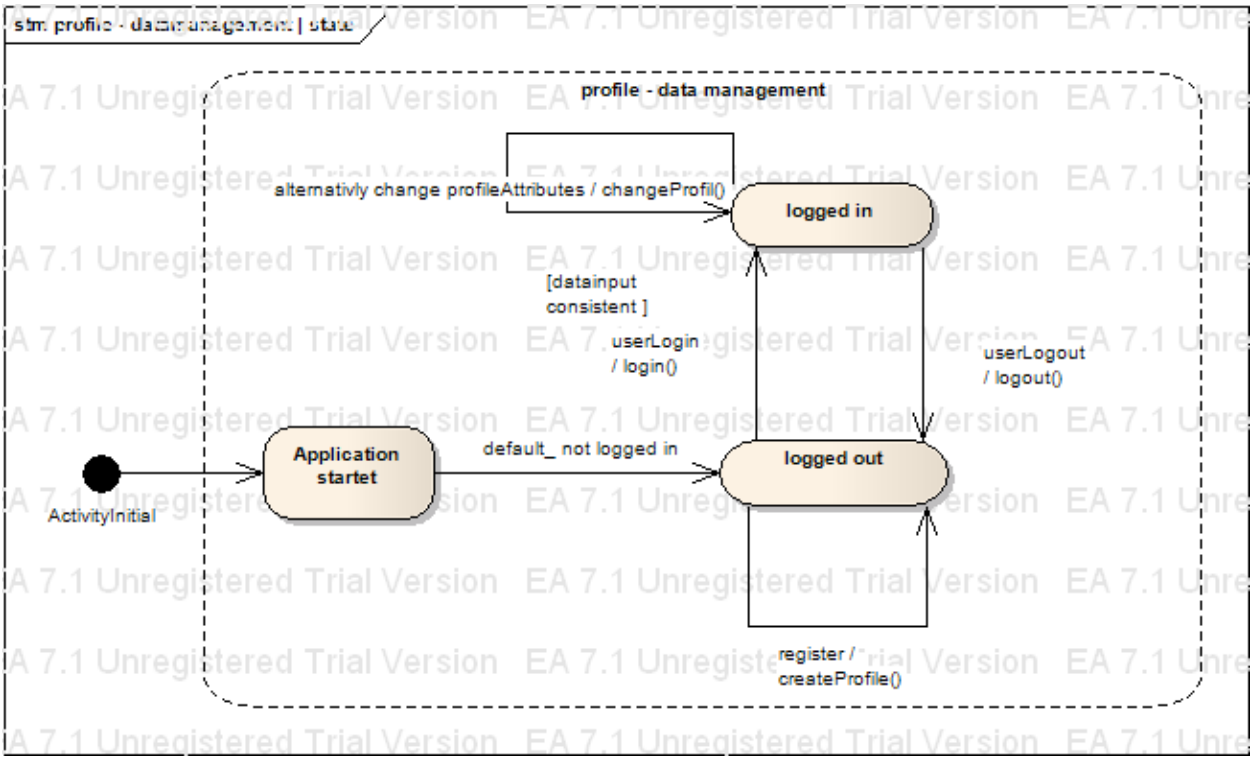


5 Zustandsdiagramm

5.1 Datenhaltung

Dieses Zustandsdiagramm beschreibt den Registrierungs- sowie den Anmeldevorgang.

Nach dem Start der Applikation ist der Benutzter standardmäßig nicht eingeloggt. Wenn der Benutzer noch keinen Account besitzt muss er sich am System registrieren um an einem Spiel teilnehmen zu können. Nachdem sich der Benutzer eingeloggt hat kann er sein Profil jederzeit aktualisieren.



6 Konfigurationsmanagement

Als Versionsverwaltungstool benutzen wir 'Subversion(SVN)'. Zu finden ist dies unter http://subversion.tigris.org/ .

Zur Sourecodedokumentation und zur Dokumentation von Änderungen greifen wir auf das Tool 'Doxygen' zurück. Dieses finden sie unter http://www.stack.nl/~dimitri/doxygen/ .

Des weiteren führen wir in jedem Sourcefile einen Header ein, der das Datum der letzten Änderung festhält und welche Änderungen genau stattgefunden haben.

Die Autoren der einzelnen Abschnitte:

Klassendiagramme : Florian Oeser und Michael Palmer

Sequenzdiagramme: Michael Palmer und Florian Oeser

Use Case Diagramme: Thomas Vogel

Aktivitätsdiagramme: Michael Palmer und Florian Oeser

Zustandsdiagramm: Thomas Vogel