### Code Review

#### Änderungsgeschichte

|  |  |  |  |
| --- | --- | --- | --- |
| Datum | Version | Änderung | Autor |
| 19.04.2012 | 1.0 | Erste Version des Dokuments, erstes Code Review | LE |
| 07.05.2012 | 1.1 | Dokumentation Code Review 03.05.2012 | DT |
| 07.06.2012 | 1.2 | Dokumentation Code Review 05.06.2012 | LE |
| 09.06.2012 | 1.3 | Review | CH |
| 12.06.2012 | 1.4 | Review | DT |
| 13.06.2012 | 1.5 | Review | LE |

#### Übersicht

Ziel dieses Dokuments ist die Vorbereitung und Dokumentation von Code Reviews. Zu Beginn werden die für das Code Review festgelegten Kriterien aufgezeigt, danach folgen die einzeln dokumentierten Code Reviews mit den jeweiligen Kommentaren.

#### Kriterien

Die Kriterienliste wurde am 19.04.12 durch das Team erstellt.

##### Code Style Analyse

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Die Code Richtlinien von HSR Videowall wurden eingehalten |  |  |
| Der Code wirkt durch seine Anordnung & Verschachtelung übersichtlich |  |  |
| Die HSR Videowall Headers sind in allen nicht generierten Sourcen vorhanden |  |  |
| Die XML-Kommentar Kompilation wurde in den Projekten aktiviert |  |  |
| Alle Public / Protected Members sind ausreichend Dokumentiert |  |  |
| Die auskommentierten Programm-Stücke sind ausreichend erklärt |  |  |
| Die Projekte enthalten keine toten Programm-Klassen |  |  |
| Die fehlenden Programmstücke sind mittels TODO-Kommentar beschrieben |  |  |
| Der Code übersetzt ohne Compiler Warnings, die nicht dokumentiert sind und sich nicht in den Test Projekten befinden |  |  |
| Es gibt keine Bad Smells im Code |  |  |

##### Exception Handling

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Der Code enthält keine abgefangenen und ignorierten Ausnahmen |  |  |
| Fehler in asynchronen Prozessen werden mittels Event weitergeleiltet |  |  |
| Das Logging erfasst alle Fehler aus allen Funktionalitätsschichten |  |  |
| Das Before/After Pattern wird, wo möglich, mittels using() { } angewendet |  |  |
| IDisposable.Dispose() Methoden werden in jedem Fall aufgerufen |  |  |

##### Flow Control

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Es existieren keine Schleifen ohne Abbruchkriterien |  |  |
| Es existieren keine nicht dokumentierte, tote Programmstücke (z.B. if(false) / while (false) / …) |  |  |
| Rekursive Calls haben immer eine Verankerung und Abbruchbedingung |  |  |

##### Naming

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Die Namen der Klassen / Variabeln sind selbstbeschreibend |  |  |
| Verwirrende oder falsche Namen sind nicht vorhanden |  |  |
| Interface-Klassen beginnen immer I (z.B. IDisposable) |  |  |
| Klassen / Properties / Methoden werden mit *PascalCasing* geschrieben |  |  |
| Lokale Variablen / Argumente werden mit *camelCasing* geschrieben |  |  |
| Konstanten werden in *PascalCasing* geschrieben |  |  |
| Felder in Klassen werden mit *\_camelCasing* geschrieben |  |  |
| Der Code enthält keine Magic Numbers |  |  |

##### Tools

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Die Warnings von Resharper 4.XXX werden, wo sinnvoll, behoben |  |  |
| Die Errors von FxCop werden wo sinnvoll behoben |  |  |

#### Code Review vom 19.04.2012

Das erste Code Review wurde am 19.04.2012 durchgeführt. Anwesend waren Lukas Elmer, Christina Heidt und Delia Treichler. Es wurde zuerst im Speziellen eine Klasse untersucht, die für die Umrechnung der Maus Position zwischen der Kinect Hand Position und dem Fenster zuständig ist. Danach wurden die Kriterien bewertet.

Das Review zieht einige Verbesserungen nach sich, die sich aus den negativ bewerteten Kriterien ergeben. Im Speziellen sind das:

|  |  |  |
| --- | --- | --- |
| Beschreibung | Bereinigung geprüft am | Kürzel |
| XML Dokumentation im Code, Headers | 05.06.2012 | LE |
| Genaue Untersuchung, ob die IDisposable.Dispose() Methoden immer aufgerufen werden | 05.06.2012 | LE |
| Es wurden „Magic Numbers“ gefunden, die dokumentiert und werden müssen ausgelagert (in statisches Attribut oder in Konfiguration). | 05.06.2012 | LE |

Tabelle 1 - Annotationen und Kommentare Code Review 19.04.2012

##### Bewertung der Kriterien

###### Code Style Analyse

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Die Code Richtlinien von HSR Videowall wurden eingehalten | X |  |
| Der Code wirkt durch seine Anordnung & Verschachtelung übersichtlich | X |  |
| Die HSR Videowall Headers sind in allen nicht generierten Sourcen vorhanden |  | X |
| Die XML-Kommentar Kompilation wurde in den Projekten aktiviert |  | X |
| Alle Public / Protected Members sind ausreichend Dokumentiert |  | X |
| Die auskommentierten Programm-Stücke sind ausreichend erklärt | X |  |
| Die Projekte enthalten keine toten Programm-Klassen | X |  |
| Die fehlenden Programmstücke sind mittels TODO-Kommentar beschrieben | X |  |
| Der Code übersetzt ohne Compiler Warnings, die nicht dokumentiert sind und sich nicht in den Test Projekten befinden | X |  |
| Es gibt keine Bad Smells im Code | X |  |

###### Exception Handling

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Der Code enthält keine abgefangenen und ignorierten Ausnahmen | ? |  |
| Fehler in asynchronen Prozessen werden mittels Event weitergeleiltet | ? |  |
| Das Logging erfasst alle Fehler aus allen Funktionalitätsschichten | ? |  |
| Das Before/After Pattern wird, wo möglich, mittels using() { } angewendet | X |  |
| IDisposable.Dispose() Methoden werden in jedem Fall aufgerufen |  | ? |

###### Flow Control

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Es existieren keine Schleifen ohne Abbruchkriterien | X |  |
| Es existieren keine nicht dokumentierte, tote Programmstücke (z.B. if(false) / while (false) / …) | X |  |
| Rekursive Calls haben immer eine Verankerung und Abbruchbedingung | X |  |

###### Naming

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Die Namen der Klassen / Variabeln sind selbstbeschreibend | X |  |
| Verwirrende oder falsche Namen sind nicht vorhanden | X |  |
| Interface-Klassen beginnen immer I (z.B. IDisposable) | X |  |
| Klassen / Properties / Methoden werden mit *PascalCasing* geschrieben | X |  |
| Lokale Variablen / Argumente werden mit *camelCasing* geschrieben | X |  |
| Konstanten werden in *PascalCasing* geschrieben | X |  |
| Felder in Klassen werden mit *\_camelCasing* geschrieben | X |  |
| Der Code enthält keine Magic Numbers |  | X |

###### Tools

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Die Warnings von Resharper 4.XXX werden, wo sinnvoll, behoben | X |  |
| Die Errors von FxCop werden wo sinnvoll behoben | X |  |

#### Code Review vom 03.05.2012

Das zweite Code Review wurde am 03.05.2012 durchgeführt. Silvan Gehrig und Michael Gfeller vom IFS gingen zusammen den Code durch und machten Notizen. Bei deren Besprechung waren auch Lukas Elmer, Christina Heidt und Delia Treichler anwesend.

Die Annotationen und Kommentare zum Code sind in der Tabelle 2 - Annotationen und Kommentare Code Review 03.05.2012 festgehalten.

|  |  |  |
| --- | --- | --- |
| Beschreibung | Bereinigung geprüft am | Kürzel |
| FileNotFoundException bei Startup, separates Resourcen-Projekt eliminiert Pfade (AutoPlayFileSkeletonReader.cs und LunchMenuReader.cs) | 05.06.2012 | LE |
| Organisation der Namespaces, Converters gehören in View (BoolToVisibilityConverter.cs) | 05.06.2012 | LE |
| Bild-Anzeige, mit RenderOptions.BitmapScalingMode="HighQuality" wird Image wesentlich besser dargestellt. | 05.06.2012 | LE |
| Code ist an einigen Stellen noch nicht kommentiert (z.B. ExtendedVisualTreeHelper.cs) | 05.06.2012 | LE |
| Internal Klasse mit public Methoden in ExtendedVisualTreeHelper.cs | - | - |
| Utils in eigenes Package evt. Common GUI-Library (ExtendedVisualTreeHelper.cs) | - | - |
| Verletzung des Information Experts in ImageExtension.cs. Initialisierungsmethode LunchMenu.CreateFrom (string fileName) nutzen | 05.06.2012 | LE |
| UI Elemente im ViewModel sind unschön und zerstört die Testbarkeit -> Verschiebung in View (z.B. HitStateArgs.cs) | - | - |
| Console.WriteLine() in Code ist sehr unschön, Logger verwenden (z.B. HitTestHelper.cs) | 05.06.2012 | LE |
| OutOfMemoryProblem Exception, geladene Posters benötigen extrem viel Memory | Nicht reproduzierbar |  |
| Filestream wird nicht geschlossen (ImageExtension.cs) | 03.05.2012 | LE |
| Kein ServiceModel vorhanden (für Domain Objekte) | 05.06.2012 | LE |
| Dispose() Pattern nicht überall vollständig implementiert (z.B. KinectSkeletonReader.cs) | 05.06.2012, LE  Dispose Pattern weggelassen, da während der Applikation keine Resourcen durch dieses Pattern freigegeben werden müssen. | |
| Anstelle Loop HitTest-Methode verwenden (Methode OnModelChanged  in HitTestHelper) | - | - |
| ViewModel-Verschachtelung: MenuViewModel weiss mehr/mächtiger als MainWindowViewModel | 05.06.2012, LE  MainWindowViewModel kennt das MenuViewModel, nicht aber umgekehrt. So wie es jetzt gelöst ist sind die Informationen nicht redundant. Evt. Könnte MenuViewModel in AppsViewModel umbenannt werden. | |
| PropertyChanged wirklich nur für Änderung am Property verwenden. (z.B. in LunchMenuService.cs: auf LunchMenuReader nicht PropertyChanged aufrufen) | 05.06.2012 | LE |
| DispatcherTimer bietet Funktionen, welche Thread based Timer macht (HitTestHelper.cs) | 05.06.2012 | LE |
| Assert in Methode RaiseEventOfUIElement in MainWindow.xmal.cs prüft Funktionalität des Frameworks | 05.06.2012 | LE |
| Regionen erstellen | 05.06.2012 | LE |
| Reihenfolge und Strukturierung von Properties / Methoden / Konstruktoren ist inkonsistent | 05.06.2012 | LE |
| Code-Guidelines (vorhanden?) einhalten | 05.06.2012 | LE |
| PreConditions einsetzen | 05.06.2012 | LE |

Tabelle 2 - Annotationen und Kommentare Code Review 03.05.2012

Die original notierten Kommentare und Annotationen von Silvan Gehrig und Michael Gfeller sind im Anhang (TODO: Code Review File in Anhang und verlinken) zu finden.

##### Bewertung der Kriterien

Die Kriterienbewertung wurde von Silvan Gehrig und Michael Gfeller am 04.05.2012 vorgenommen.

###### Code Style Analyse

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Die Code Richtlinien von HSR Videowall wurden eingehalten |  | X |
| Der Code wirkt durch seine Anordnung & Verschachtelung übersichtlich |  | X |
| Die HSR Videowall Headers sind in allen nicht generierten Sourcen vorhanden |  | X |
| Die XML-Kommentar Kompilation wurde in den Projekten aktiviert | X |  |
| Alle Public / Protected Members sind ausreichend Dokumentiert |  | X |
| Die auskommentierten Programm-Stücke sind ausreichend erklärt | X |  |
| Die Projekte enthalten keine toten Programm-Klassen |  |  |
| Die fehlenden Programmstücke sind mittels TODO-Kommentar beschrieben | X |  |
| Der Code übersetzt ohne Compiler Warnings, die nicht dokumentiert sind und sich nicht in den Test Projekten befinden |  | X |
| Es gibt keine Bad Smells im Code |  | X |

###### Exception Handling

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Der Code enthält keine abgefangenen und ignorierten Ausnahmen | X |  |
| Fehler in asynchronen Prozessen werden mittels Event weitergeleiltet | X |  |
| Das Logging erfasst alle Fehler aus allen Funktionalitätsschichten |  | X |
| Das Before/After Pattern wird, wo möglich, mittels using() { } angewendet |  | X |
| IDisposable.Dispose() Methoden werden in jedem Fall aufgerufen |  | X |

###### Flow Control

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Es existieren keine Schleifen ohne Abbruchkriterien | X |  |
| Es existieren keine nicht dokumentierte, tote Programmstücke (z.B. if(false) / while (false) / …) | X |  |
| Rekursive Calls haben immer eine Verankerung und Abbruchbedingung | X |  |

###### Naming

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Die Namen der Klassen / Variabeln sind selbstbeschreibend | X |  |
| Verwirrende oder falsche Namen sind nicht vorhanden |  | X |
| Interface-Klassen beginnen immer I (z.B. IDisposable) | X |  |
| Klassen / Properties / Methoden werden mit *PascalCasing* geschrieben | X |  |
| Lokale Variablen / Argumente werden mit *camelCasing* geschrieben | X |  |
| Konstanten werden in *PascalCasing* geschrieben | X |  |
| Felder in Klassen werden mit *\_camelCasing* geschrieben | X |  |
| Der Code enthält keine Magic Numbers |  | X |

###### Tools

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Die Warnings von Resharper 4.XXX werden, wo sinnvoll, behoben | ? |  |
| Die Errors von FxCop werden wo sinnvoll behoben | ? |  |

#### Code Review vom 05.06.2012

Das letzte Code Review wurde am 05.06.2012 durchgeführt. Anwesend waren Michael Gfeller, Silvan Gehrig und Lukas Elmer. Zuerst wurde der Code allgemein von Michael Gfeller untersucht und die gefundenen Probleme wurden besprochen.

Danach wurden die Bewertungen des Code Reviews von Michael Gfeller am 12.06.2012 eingetragen.

Das Review zog einige Verbesserungen nach sich, die sich aus den negativ bewerteten Kriterien ergaben. Im Speziellen waren das:

|  |  |  |
| --- | --- | --- |
| Beschreibung | Bereinigung geprüft am | Kürzel |
| Exceptions, die applikationsbedingt sind, durch eine spezifische VideoWallException ersetzen. | 07.06.2012 | LE |
| Exception, die keine VideoWallException ist, nicht anzeigen. Nur anzeigen, dass ein Fehler aufgetreten ist ohne spezifisch auf den Fehler einzugehen (zu viele Technische Informationen für den Benutzer). | 07.06.2012 | LE |
| Das Canvas des Mousecursors könnte in UserControl ausgelagert werden (MainWindow der VideoWall). | - | - |
| Wenn Applikation beendet wird sicherstellen, dass sie wirklich beendet wird mittels Process.CurrentProcess.Kill(). | 07.06.2012 | LE |
| Mensa Menu vertikal zentrieren. | 07.06.2012 | LE |
| Vollbild über mehrere Bildschirme sicherstellen: Manuell programmieren mithilfe der Screen Klasse. | 07.06.2012 | LE |
| State Machine Demo Modus: Timer könnte ausgelagert werden und die Methode Tick() könnte public gemacht werden. | - | - |
| State Machine Demo Modus: Switch Statement könnte mihilfe eines State Patterns umgesetzt werden. Es wurde aber besprochen, dass dies nicht mehr gemacht werden soll, da das Switch Statement eine gute Übersicht bietet und die aktuelle Lösung pragmatisch ist. | 07.06.2012, LE  Wie besprochen, ist nicht nötig | |
| Für die PreOrPostCondition.Assert[…] gibt es evt. ein Debug Attribut, damit der Debugger nicht in diese Klasse springt sondern in der Klasse bleibt, wo die Condition tatsächlich fehlgeschlagen hat. | 07.06.2012, LE  Debug Attribut nicht gefunden | |
| Tests: Test Directory kann über Konfiguration hinzugefügt werden. Damit können Dateien und Ordner in den Tests verwendet werden. | 07.06.2012 | LE |
| Um zu zeigen, dass die Applikation im Betrieb keine (gravierenden) Memory Leaks beinhaltet, soll die Applikation über 24h gelaufen lassen werden. Danach soll analysiert werden, ob der Memory Verbrauch etwa gleichmässig ist. | 07.06.2012 | LE |
| Das IDisposable Interface wird nicht von allen Klassen implementiert. Dies könnte beim Herunterfahren der Applikation im schlimmsten Fall zu einem Absturz führen. Da dies für die Videowall bis jetzt nicht kritisch ist, muss dies aber nicht noch implementiert werden. | 13.06.2012, LE Da dieses Kriterium bisher nicht auf der Kriterienliste des Code Reviews war, wurde dies noch hinzugefügt:  „Alle Elemente, die (beim Herunterfahren der Applikation) aufgeräumt werden müssen, implementieren das Interface IDisposable“ | |

Tabelle 3 - Annotationen und Kommentare Code Review 05.06.2012

##### Bewertung der Kriterien

Die roten Kommentare wurden von Michael Gfeller eingefügt oder gemäss den Mails zwischen Michael Gfeller und Lukas Elmer vom 12. Juli (TODO: Anhang, ist schon im Anhang Ordner) ergänzt.

###### Code Style Analyse

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Die Code Richtlinien von HSR Video Wall wurden eingehalten | X |  |
| Der Code wirkt durch seine Anordnung & Verschachtelung übersichtlich | X |  |
| Die HSR Video Wall Headers sind in allen nicht generierten Sourcen vorhanden | X |  |
| Die XML-Kommentar Kompilation wurde in den Projekten aktiviert Problem nur im DiagnosticApp, das nicht zum Code Review gehört | X |  |
| Alle Public / Protected Members sind ausreichend Dokumentiert  Problem nur im DiagnosticApp, das nicht zum Code Review gehört | X |  |
| Die auskommentierten Programm-Stücke sind ausreichend erklärt | X |  |
| Die Projekte enthalten keine toten Programm-Klassen | X |  |
| Die fehlenden Programmstücke sind mittels TODO-Kommentar beschrieben | X |  |
| Der Code übersetzt ohne Compiler Warnings, die nicht dokumentiert sind und sich nicht in den Test Projekten befinden | X |  |
| Es gibt keine Bad Smells im Code Diverse Kleine sind noch vorhanden. | X |  |

###### Exception Handling

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Der Code enthält keine abgefangenen und ignorierten Ausnahmen | X |  |
| Fehler in asynchronen Prozessen werden mittels Event weitergeleiltet | X |  |
| Das Logging erfasst alle Fehler aus allen Funktionalitätsschichten | X |  |
| Das Before/After Pattern wird, wo möglich, mittels using() { } angewendet | X |  |
| IDisposable.Dispose() Methoden werden in jedem Fall aufgerufen | X |  |
| Alle Elemente, die (beim Herunterfahren der Applikation) aufgeräumt werden müssen, implementieren das Interface IDisposable  Wie besprochen; Noch nicht komplett implementiert |  | X |

###### Flow Control

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Es existieren keine Schleifen ohne Abbruchkriterien | X |  |
| Es existieren keine nicht dokumentierte, tote Programmstücke (z.B. if(false) / while (false) / …) | X |  |
| Rekursive Calls haben immer eine Verankerung und Abbruchbedingung | X |  |

###### Naming

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Die Namen der Klassen / Variabeln sind selbstbeschreibend | X |  |
| Verwirrende oder falsche Namen sind nicht vorhanden | X |  |
| Interface-Klassen beginnen immer I (z.B. IDisposable) | X |  |
| Klassen / Properties / Methoden werden mit *PascalCasing* geschrieben | X |  |
| Lokale Variablen / Argumente werden mit *camelCasing* geschrieben | X |  |
| Konstanten werden in *PascalCasing* geschrieben | X |  |
| Felder in Klassen werden mit *\_camelCasing* geschrieben | X |  |
| Der Code enthält keine Magic Numbers | X |  |

###### Tools

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Die Warnings von Resharper 4.XXX werden, wo sinnvoll, behoben  Die „disabled Warnings“ vom Reshaper stören den Lesefluss im Programmcode erheblich. Zusätzlich besteht die Gefahr, dass bei neuen Anpassungen gerade weitere Unschönheiten durch die Bestehenden „disabling“-Anweisung ausgeblendet werden.  Daher: Lieber nicht disablen und im Source (z.B. als Remark / …) beschreiben, warum dies so sein muss als zukünftige Implementationen erschweren.  Ansonsten ist dies ok. | X |  |
| Die Errors von FxCop werden wo sinnvoll behoben | X |  |