## 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 |

### Übersicht

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

### Kriterien

Die Kriterienliste wurde am 19.04.12 vom 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 |  |  |
| 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 | Bereinigungsdatum | Kürzel |
| XML Dokumentation im Code, Headers |  |  |
| Genaue Untersuchung, ob die IDisposable.Dispose() Methoden immer aufgerufen werden |  |  |
| Es wurden „Magic Numbers“ gefunden, die dokumentiert und werden müssen ausgelagert (in statisches Attribut oder in Konfiguration). |  |  |

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 | 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 - Issues Code Review 03.05.2012 festgehalten.

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

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 |  | 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 | ? |  |