CommunityMirror - Grundarchitektur und Wording

CommunityMirror - Informationsstrahler

Ein CommunityMirror ist ein **„interaktiver Informationsstrahler"**

(funktional, Szenario)

- Ziel: Informationen / Signale aussenden, die bei (menschlichen) Empfängern eine Verbesserung der Handlungsfähigkeit auslösen / den handlungsrelevanten Informationsstand bei den Empfängern erhöhen - d.h. proaktiv, wartet nicht auf Anfrage

- Empfänger suchen diese Signale nicht explizit, sondern nehmen sie erst mal peripher und dann vielleicht bewusst war und interagieren vielleicht am Ende noch mit dem Gerät (ohne aber etwas bestimmtes zu suchen) - d.h. Entdecken anstelle von Suchen, kein Kiosk, d.h. Aufmerksamkeit muss erregt werden, Walk-Up-And-Use, Joy-of-Use, ...
- Es ist schon möglich, dass Empfänger den Informationsstrahler gezielt aufsuchen - aber nicht, dass sie auf / mit dem Informationsstrahler vorher festgelegte Fragen beantworten

- die Nutzung ist normalerweise nur kurz
- gleichzeitige Nutzung durch mehrere Nutzer möglich / wahrscheinlich; dabei verschiedene Typen der Nutzung in verschiedenen Interaktionszonen (Entfernungen vom Informationsstrahler)

Schaffung von Gewahrsein / Awareness - zu Personen und was die machen (INFMirror), zu Aktivitätsmöglichkeiten (UrbanLife+), ...

Grundstruktur

- Wandbildschirm

- interaktiv (siehe unten)
- groß O(1m) -> dadurch Multi-User-fähig (in verschiedenen Interaktionszonen aber auch in derselben)

Darstellung der Signale / Information

- hauptsächlich visuelle Ausgabe (auf dem Wandbildschirm)
- akustische Signale eventuell zur Ergänzung (Ton von Videos, …)

Interaktionsmöglichkeit

- nicht um gezielt etwas zu suchen, sondern um
  - noch mehr Ungesuchtes zu entdecken
  - um entdeckte Information zu vertiefen
- primär über Touch-Interaktion
- eventuell auch Embodied Interaction

Einordnung der **Anwendungsklasse "interaktiver Informationsstrahler / Community-Mirror"** zwischen

- Digital-Signage (immer nur ein Signal, versucht Aktivität zu erzielen - nicht Wissen zu generieren) und
- Kiosk (aktive, bewusste, zielgerichtete Beschäftigung)

CommunityMirror - Architektur

Ein CommunityMirror besteht technisch aus

- UI-Hardware (großer Wandbildschirm mit Touch-Erkennung) - siehe Auflistung aktuell verfügbarer Hardware in Große Wandbildschirme
- Rechner-Hardware: ein Rechner zum Laufenlassen der Software - **TBD: diesen Teil noch auf eigener Seite ausführen ...**
- Software: Eine Instanz des CommunityMirrorFrameworks (d.h. CMF mit speziellen Konfigurationen, Templates)
- Service: Eine Instanz des CommunityMashup auf einem Server um die Daten einzusammeln und aufzubereiten

Wichtigstes Konzept ist dabei die Trennung der CommunityMirror-App (Instanz des CommunityMirrorFrameworks) und der CommunityMashup-Instanz, von der die Daten bezogen werden (und auf die verwiesen werden kann für eine Referenz / URL auf die Daten).

Für die CommunityMirror-App gibt es weitere Architekturüberlegungen:

- das CMF nutzt JavaFX um Objekte zu zeichnen und zu animieren
- das CMF stützt sich auf Funktionen einer Abstraktionsschicht namens CommunityInteractionFramework um Touch-Events zu empfangen
    - diese Abstraktionsschicht stützt sich aktuell dann entweder auf JavaFX oder auf GestureWorks
    - Aktuelle Probleme: GestureWorks läuft nicht stabil; JavaFX unterstützt folgende Features nicht: Inertia (bei Drag), Multi-User (Drag)

Im CMF kann folgendermaßen konfiguriert / gethemed werden:

- TBD

TBD Aktuellen Stand dokumentieren

Verschiedene Dokumente zur Architektur (teilweise veraltet)

- **CommunityMirror - Grundarchitektur und Wording**
- CommunityMirror - Grundarchitektur und Wording

CommunityMirrorFramework / CM-Anwendungen/Anwendungsklassen

CommunityMirrorFramework (CMF) = Modularer Baukasten für das schnelle und einfache Erstellen verschiedener einsatzszenariospezifischer ***CommunityMirror-Anwendungen***

CommunityMirror-Anwendung (CMA) = Einsatzszenariospezifische Kombination aus

- Einsatzfeld (z.B. Konferenz, Empfangshalle, Kaffee-Ecke, etc.)
- Datenquelle(n) (Mashup Endpoint, ggf. gefiltert),
- Views (jetzt ggf. eher "Filter / Einstiegspunkte, da View ja immer identisch),
- Kontext, d.h. Anzahl Screens & Aufstellungsort(e), Nutzergruppe(n), etc.,
- Plugins (z.B. Fingerprint, RFID, iBeacon) und
- Zusatzanwendungen (z.B. Mobile-Apps)

In der Vergangenheit haben wir von folgenden konkreten Anwendungen (Anwendungsklassen) gesprochen:

- ***MeetingMirror***: CMA für die Konferenzunterstützung (Personenfokus, wissenschaftliche Tagungen)
- ***IdeaMirror***: CMA für die Unterstützung des Innovationsmanagements (Inhaltsfokus, Innovationsmanagementsysteme im Unternehmenskontext, gab es im Prinzip schon als IdeaNetMirror für Hyve oder als SapiensMirror für SAP)
- ***LibraryMirror***: CMA zur Unterstützung der Awareness in Bibliotheken (Awaress-Fokus / Schlagworte, Ausleihungen in Bibliotheken)
- ***ExhibitMirror***: CMA zur Präsentation von redaktionell aufbereiteten Projekten / Forschungsergebenissen (Inhaltsfokus, Vernetzung von Personen und Content, z.B. MS Wissenschaft, Tag der offenen Tür UniBwM)
- ***SocialNetworkingMirror***: CMA zur VisualiIsierung von sozialen Netzwerken (Personenfokus, Erweiterung des ursprünglichen MeetingMirror-Konzepts, z.B. auf der Webinale eingesetzt)
- ***CollabMirror***: CMA zur Visualisierung von Aktivitätsströmen (Aktivitätsfokus / Awareness, Kombination der verschiedenen bisherigen Ansätze, Erste Anwendungen bwsp. SkiBaserlMirror oder WikiBwMirror, später 3M Community)

In Veröffentlichungen sollte möglichst immer zuerst von "CommunityMirror" gesprochen werden - und nur wenn es um spezifische Eigenschaften einer Anwendungsklasse geht von dieser Anwendungsklasse.

CommunityMirrorFramework - Komponenten

Im CommunityMirrorFramework wird unterschieden zwischen

- Gesamtlayout/animation - also den ganzen Bildschirm betreffend - die potentiell mehrere Informationsobjekte aus dem Mashup zeigt -> View
- Layout/Animation eines einzelnen Informationsobjektes (aus dem Mashup) -> Info Particle

Hier eine Liste der verschiedenen Elemente in der UI: **TBD: Überarbeiten und Aktualisieren**

- **Info Particle:** Visuelle Repräsentanz eines Informationsobjekts (aus dem Mashup). Hat mehrere Zustände, z.B. MicroView, Preview, DetailView (die ggf. noch anders benannt werden).
- **View**: Globales Layout (Hintergrund) + Visualisierung der Informationsobjekte und deren Verhalten zueinander (optional, beispielsweise Fisch), NICHT Interaktionskonzept mit den Informationsobjekten (das ist bei allen Views gleich), aber insbesondere "Erschein- und Verwindelogik", reagieren ggf. in 2.0 auf Kontext (beispielsweise über iBeacon für Benutzererkennung):
    - FischView
    - FlowView: Standardpartikelfluss. Hier v.a. auch Logik für Partikel-"Verdränung" erforderlich, damit **Workspace**-Inhalte noch gut erkennbar sind - auch für andere Views relevant.
    - MicroPostView (z.B. klasische TwitterWall oder ggf. auch Lavalampen-Metapher)
    - NavigationView (Terminal an der Wache)
- **Workspace** (=Browser, = Userzone): (Ggf. visualisierter) Bereich, der benutzerspezifisch mehrfach auf dem Bildschirm auftauchen kann und primär der Exploration dient. Im Workspace wird ein **Graph** angezeigt.Sofern Benutzer identifizierbar ist, wird der Workspace einem Benutzer direkt zugeordnet und kann in 2.0 ggf. personalisiert werden.

- **Graph**: Enthält Verbindungen der Partikel untereinander sowie zusätzliche dynamische **Entrypoints** je nach Inhalte des Informationsobjekts. Graph sollte sinnvollerweise in irgendeiner Art "offen" bleiben, um Benutzern den Bezug zu geben, wo sie herkommen. Ggf. können Automatismen sinnvoll sein (2.0?), um zu große Graphen vom Ende her wieder zu "schließen" bzw. zu verkleinern.
- **Entrypoint**: Mit spezifischen Daten vorbelegter **Selector** zur einfachen Auswahl einer Teilmenge von Informationsobjekten. Unterschieden wird zwischen
    - Dynamischen Entrypoints: Tauchen direkt als verbundene Nodes im Graph innerhalb der Workspace auf und erlauben beispielsweise das schnelle "Browsen" zu Teilmengen wie "Personen gleicher Institution", "Inhalte vom gleichen Ort", etc.
    - Anwendungsspezifische Entrypoints: Stellen je nach Anwendugskontext (z.B. MeetingMirror auf wissenschaftlichen Konferenzen) potenziell interessante Entrypoints zur Verfügung (z.B. "Vorträge innerhalb der nächsten 2 Stunden", etc.)
- **Selector**(s): Verschiedene Klassen von Visualisierungen und Interaktionsmöglichkeiten für Entrypoints. Erlauben die Vorbelegung mit verschiedenen "Mashup-Suchkriterien", ermöglichen die (weitere) Einschränkung der Informationsobjekte und zeigen die aktuelle Auswahl dynamisch in einer **SelectorPreview** an. In 2.0 sind Selektoren ggf. kombinierbar, so dass beispielsweise Vorträge aus Bayern am nächsten Tag ausgewählt werden könnten. In irgendeiner Form sollte eine Art "History" bzw. zumindest ein "Weg zum aktuellen Selektor" (z.B. Pfleil, der am aktuellen Hauptpartikel des Graph hängt). Beispiele für Selektoren sind:
    - MapSelector
    - TagcloudSelector
    - AlphabeticListSelector
    - TextSearchSelector - hier v.a. Touch-Tastatur erforderlich
    - TimelineSelector
    - ...
- **SelectorPreview**: Vorschau der aktuellen Entrypoint- bzw. Selektorauswahl. Sinnvollerweise als "Konstante", die dem Benutzer bei Selektionen immer einen direkten "Effekt" darstellt. Verwendet im Optimalfall existierende Renderer (z.B. MicroView) und zeigt parallel insbesondere die Anzahl der aktuell ausgewählten Elemente an.
- **Toolbox**: Enthält anwendungsspezifischen (und dynamische?) Entrypoints in einer ggf. nach bestimmten Kriterien gruppierten / sortieren Übersicht. Kann ggf. auch ohne Partikel "aufgerufen" werden (z.B. Tap & Hold).

Partikelsystem

Generell: Soweit möglich, sollte alles über die Partikel-Metapher abgebildet werden, d.h. statt großer INTRApartikelinhalte (beispielsweise Darstellung von Autor / Organisation zu einem Content) sollte die Information besser über INTERpartikel-Beziehungen (beispielsweise das "Dranhängen" eines Autoren- bzw. Organisationspartikels an einen Content) abgebildet werden.

Partikeldetaillierungsgrade und -zustände

TBD: Wie kann zwischen Detailierungsgraden (hier brauchen wir noch ein besseres Wort!) gewechselt werden (Tap zu mehr Details ist einfach, wie kommt man aber "zurück"), ggf. per Pinch? Kommt man von inDetail wieder zu inFlow? Wird inFlow bei Tap gecloned oder "verwendet".

- (inNano): ggf. nicht erforderlich, reine Andeutung weiterer Partikel als Dot, Box, etc. für komplexe Graphen
- inMicro: Kleinste informationstragende Darstellung eines IPs, primär im Graph für komplexere Layouts erforderlich
- inPreview: "normale" Darstellung eines IPs im Graph, hierbei v.a. Schwerpunkt auf Bildmaterial für "User Attraction" und das Wecken von ausreichendem Interesse (kein "grau in grau")
- inDetail: Detailanzeige, einzige Darstellungsform bei der wirklich alle Inhalte angezeigt werden, ggf. auch Anhänge (z.B. PDFs von Papern = Content) zum "Durchblättern".
- inFlow: Im Prinzip ähnlich wie inPreview, allerdings sollten die Größen der verschiedenen inFlow-Partikel randomisiert werden, um das Lesen aus verschiedenen Entfernungen (vgl. Paper von Domhardt&Schmidt, 2013: Ergonomischer_Schriftgrad.pdf) zu ermöglichen. Außerdem ggf. zusätzliche Animationen, um Flow interessanter zu machen, was bei inPreview (im Graph) ggf. eher störend sein könnte. Partikel inFlow benötigen ggf. eine andere "Sprache", d.h. hier geht es eher um Aktivities (Peter hält Vortrag am XX um XX in Raum XX zu Paper XX) als um statischen Content ("nur" Paper von Peter").
- inSpotlight: quasi die "Digital Signage" Version eines Partikels, ggf. reduziert bis auf eine (riesige) Headline + ggf. Bild in Sonderanschnitt, um bessere Aufmerksamkeit / Bewegung und Informationsaufnahme im "Vorübergehen" zu ermöglichen.

Für alle diese Detailierungsgrade sollte es grundsätzlich die Möglichkeit geben, folgende Zustände als Addons (z.B. über entsprechende "visuelle Anhängsel" wie Badges, Farbänderungen, Rahmen, etc.) zu definieren (2.0):

- new: gerade (Zeitfenster definierbar) neu hinzugekommener Partikel
- updated: gerade (Zeitfenster definierbar) aktualisierter Partikel
- featured: redaktionell hervorgehobener Partikel
- highlighted: in einer Visualisierung (z.B. Graph) hervorgehobener Partikel (scheint auf Basis der Erfahrungen mit Prefuse ein SEHR sinnvoller Zusatzzustand)
- focussed: aktuell (von einem der Nutzer) fokussierter Partikel