**Erstmal quasi ungefiltert**

Formatierung: Spiele sollten kursiv sein, andere Patterns dann irgendwie anders formatieren. Quellenangaben wohl sinnvoll.

Roland: „ich würde versuchen, basierend auf Deinem Literaturergebnis Bestandteile des Frameworks auszuwählen und zusammenzustellen, die speziell sind für AR oder spezifische Probleme von AR lösen. Dabei kannst Du bewusst eine Einschränkung vornehmen auf Hololens (und vergleichbare Geräte) als Plattform. Das führt Dich z.B. zur Diskussion von folgenden Aspekten:

- Game design (erstellen von Spielen für unbekannte Räume, Spiele mit/ohne Bezug zum Raum)

- Interaktion (Steuerung, Navigation, Auswahl)

- Visualisierung (HUD, Navigationshilfen, virtuelle Objekte und ihre Platzierung, NPCs)

- optional: Multiplayer-Fragen? (z.B. sharing von Hologrammen? Was wäre mit remote-multiplayer?)

Daraus lassen sich dann Vorschläge/Konzepte ableiten, von denen ausgewählte in Deinem Prototypen landen“  
**-> Das noch machen (Zu Visualisierung und Interaktion sind ja schon Sachen da)**

* **GD Patterns unten**

**Wichtig: Game-Beispiele da Game-Framework -> Infos zu HL-Spielen raussuchen!**

**Wekit-Quelle:**

* Haptic Feedback
* Directed Focus/Gaze
* Movement Instruction (Stand here)
* Autopause/-play
* Gaze to select/Cursor („activate spot by staring“)
* Shared pointer (Multi-user)
* Voice commands

**Andere Quellen**:

* OIV und was damit zusammenhängt
* Information filtering, je nach Bedürfnissen
* POI
* Gesture-Based Interaction
* ~~Fiducial~~ (Macht keinen Sinn für HoloLens, ist auch nicht in Dunleavy-Quelle)
* Audio Augmentation (Ternier)?

**Name**: Auto-Play

**Forces/Problem**: How can events in Augmented Reality be triggered without requiring or despite varying kinds of user input?

**Feature/Solution**: Some events may work best if started, paused and/or stopped automatically. Automatic here refers to a lack of directed efforts to create input, for example based on location or gaze. It is possible to automatize only parts of any event. Results of Directed Focus or Directed Movement are not considered Auto-Play as the user is clearly instructed to perform certain actions.

**Examples**: Applications using Points of Interest can use varying levels of Auto-play: They either immediately make the content accessible (as with the audio recordings in Ternier, De Vries et al.’s field trip application), they automatically retrieve some content or they simply make the user aware of the existence of such content.

**Effects/Consequences**: Content presented automatically can allow the user to avoid tunnel vision and make using the application more pleasant as less effort is required. Automatic warnings can prevent injury or incorrect use. If too much information is communicated too suddenly, Auto-Play could also confuse or disorient the user or cause screen clutter. If there is no apparent way to close unwanted content, the user may become frustrated.

**Sensors**: What sensors are necessary for this is entirely dependent on the application. Automatic content may for example be based on position, gaze/orientation, audio input, or fiducials.

**Name**: Directed Focus

**Forces/Problem**: How do you direct a user’s attention to something when they have full control over their view?

**Feature/Solution**: Use an icon to indicate something the user should direct their attention towards; affix the icon to the screen, pointing towards the object of interest if it is not currently visible. Remove the icon after the user has looked at the focus point.

**Examples**: *Fragments* uses Directed Focus at some points to direct the player towards important information.

**Effects/Consequences**: Directed Focus icons can make for a more structured AR experience and avoid user confusion. However, they may obstruct other, possibly important elements and cause screen clutter. If multiple focus points exist at a time, Information Filtering may prove necessary to avoid this. To communicate to the user where the system currently assumes their gaze is directed, use a Gaze Cursor.

**Sensors**: The system always needs to be aware of the position of the focus point relative to the user or the user’s gaze. How this works is dependent on the AR device, but generally, IMUs will be most useful for this.

**Name**: Directed Movement

**Forces/Problem**: Some applications may require the user to move to certain locations. How do you communicate these?

**Feature/Solution**: Display an icon at the target location; affix the icon to the screen, pointing towards the object of interest if it is not currently visible. Remove the icon after the user has moved to the target location.

**Examples**: On a macro-scale, a Point of Interest can serve as a form of Directed Movement if permanently displayed on a map and/or in direction.

**Effects/Consequences**: Directed Movement icons can make for a more structured AR experience and avoid user confusion. However, they may obstruct other, possibly important elements and cause screen clutter. If multiple focus points exist at a time, Information Filtering may prove necessary to avoid this.

**Sensors**: The system always needs to be aware of the position of the focus point relative to the user’s position and possibly gaze. How this works is dependent on the AR device; vision-based systems can use IMUs, while location-based systems can also utilize e.g. GPS sensors.

**Name**: Environment-Adaption

**Forces/Problem**: How do you design a game that can be run anywhere while still taking into account the characteristics of that specific environment?

**Feature/Solution**: Make the game automatically adapt to the characteristics of the environment.

**Examples**: The levels and character interactions in *Young Conker* are based on local characteristics, e.g. the player character teeters when near ledges. *Fragments* automatically finds surfaces with specific characteristics – for instance height and surface size – for characters to sit on.

**Effects/Consequences**: The game will be able to run anywhere (unrelated issues notwithstanding) while making use of the affordances of Augmented Reality. However, they cannot be planned as stringently and different users may have entirely different experiences.

**Sensors**: Depend on the degree of interaction and specifics of the application. For general purposes, depth sensors may be sufficient.

**Name**: Environment-Independence

**Forces/Problem**: How can you develop AR games for unknown environments?

**Feature/Solution**: Make the game not interact with the environment.

**Examples**: Universal Windows Platform games not optimized for the HoloLens are presented in windows which can be placed on surfaces while the games themselves do not account for the environment. Location-based games like *Pokémon GO* may take into account map data but will ignore current local conditions.

**Effects/Consequences**: Environment-Independentgames are simpler in their construction and will function more reliably and predictably. However, their classification as Augmented Reality may be questioned and users may not get as much enjoyment out of them as more interactive applications.

**Sensors**: Aside from the sensors needed for the system itself, Environment-Independentgames are intended to function without extra sensors.

**Name**: Environment Requirements

**Forces/Problem**: How can you design a game which interacts with environments without changing the game itself?

**Feature/Solution**: Make the game require certain environmental features.

**Examples**: Games based on fiducial markers require a prepared environment. Some AR applications such as the *Touring Machine* (Feiner et al.) are designed for use in one specific location. The HoloLens allows for different kinds of environmental queries which could be used to judge whether the space is appropriate.

**Effects/Consequences**: Having Environment Requirementsensures that a game works as intended (other issues notwithstanding) while still making use of the affordances of Augmented Reality. However, having certain requirements means that some users will be unable or unwilling (refusing to make changes to their environment) to play the game.

**Sensors**: Application-dependent.

**Name**: Gaze Cursor

**Forces/Problem**: In an environment with multiple interactive objects, how do you select which ones to apply actions to?

**Feature/Solution**: Base actions on the user’s gaze or orientation and communicate this process graphically as with a computer’s cursor. The Gaze Cursor can be used either for only selecting objects or for completing actions, for example based on duration of gaze.

**Examples**: Various HoloLens applications utilize Gaze Cursors, such as *RoboRaid* for targeting enemies and *Young Conker* for controlling the player character. In a broader sense, the entirety of the scene taken in by the camera in a marker-based AR system can be considered to be under the Gaze Cursor.

**Effects/Consequences**: Using a Gaze Cursor should make interactions more clear to the user if the cursor is accurate and consistent. Inaccurate, inconsistent, or lagging cursors may however make interaction frustrating and unreliable. A graphical representation of the user’s gaze will take up screen space – one should ensure that the object of the user’s interest is not obscured by the cursor. There might also be a danger of tunnel vision.

**Sensors**: The system must be aware of the user’s gaze, which can be achieved by tracking their head movements. Eye tracking can provide a higher degree of accuracy if available.

**Name**: Gesture-based Interaction

**Forces/Problem**: How can users interact intuitively with AR environments?

**Feature/Solution**: Allow the manipulation of objects through gestures

**Examples**: Gestures are one of the three forms of interaction for HoloLens applications. Games have previously used gestures to control games for the Microsoft Kinect. Touch displays are controlled through gestures on the screen.

**Effects/Consequences**: Without additional input devices, AR systems may be more immersive and intuitive to use. However, users may accidentally make inputs. If inputs are not intuitive or require too much effort, the users may become frustrated.

**Sensors**: Gestures can be used as input with specialized devices such as the Leap Motion, Myo armband or Microsoft Kinect. Simple gesture recognition could also be implemented with only a camera and computer vision algorithms.

**Name**: Haptic Feedback

**Forces/Problem**: User may miss visual or audio feedback; user does not receive appropriate feedback when “touching” augmented objects

**Feature/Solution**: Give feedback via the haptic sense, either accurately or generally

**Examples**: Vibrotactile feedback, a form of Haptic Feedback, is a standard feature of game controllers. The Myo Armband can give such feedback while keeping the user’s hands free. The NormalTouch and TextureTouch devices (Benko et al.) are able to give somewhat accurate haptic representations of virtual objects.

**Effects/Consequences**: Users may be more responsive to Haptic Feedback, especially if their attention is not on the augmentation. The AR device has to implement haptic technology, or other devices have to be added – if the user has to hold the feedback device in one or more hands, their freedom of movement will be decreased. (Inappropriate) Haptic Feedback may break immersion.

**Sensors**: If accurate feedback corresponding to body movements is to be given, hand or full-body tracking must be used (e.g. with Leap Motion or Kinect)

**Name**: Information Filtering

**Forces/Problem**: Too much information available at a time will clutter the screen and make the application more difficult to use. Too little information will make the application less useful.

**Feature/Solution**: Filter information e.g. by distance and angle of gaze.

**Examples**: Julier et al. utilized Information Filtering in their sniper avoidance system. In *Fragments* new evidence only materializes upon approaching it.

**Effects/Consequences**: Information Filtering makes systems easier to use. Designers need to ensure that the filter parameters reflect the intention of the application to avoid presenting too little or too much information.

**Sensors**: Sensors depend on the exact filter mechanism. If distance and viewing angle are to be used, the system needs access to this data e.g. through IMUs and geolocation systems.

**Name**: Obscured Information Visualization

**Forces/Problem**: Some applications may want the users to keep track of content that is not visible at a given time.

**Feature/Solution**: Make it visible through other objects (various approaches).

**Examples**: Furmanski et al. explored multiple design approaches to Obscured Information Visualization and compared their effectiveness, though they note a number of factors which might have influenced the results. The game *Little Conker* has the rim of the main character visible when obscured by real objects. Many non-AR games allow the player to see other players through walls.

**Effects/Consequences**: Obscured Information Visualization can afford the user a better understanding of their (augmented) environment. However, some approaches may cause depth perception issues.

**Sensors**: The system needs to be aware of the positions and rotations of both real objects and the augmented content, therefore a completely passive marker-based approach will not work; active RFID chips could function but the most reliable approaches visualize static or completely virtual objects in a known environment through head tracking, e.g. with IMUs.

**Name**: Point of Interest

**Forces/Problem**: How can information be provided to users in a location-based system? How can information be bound to locations that are not reasonably accessible to developers?

**Feature/Solution**: Bind information to location data, automatically making it available to the user upon getting within a predefined range and allowing you to direct users to such points.

**Examples**: Several Augmented Reality browsers have implemented Point of Interest approaches. Ternier, De Vries et al. utilized it to guide students on a field trip. *Alien Contact!* (Dunleavy, Dede & Mitchell) is based on Points of Interest represented by real objects.

**Effects/Consequences**: Points of Interest either cannot overlap or the system needs a method to handle such overlap. Constantly gathering information about user location and comparing it to Points of Interest may consume a lot of energy, which could negatively affect the user. If the user is directed towards Points of Interest, care must be taken to avoid screen clutter and information overload. A degree of precision is lost if only location data is used.

**Sensors**: Location technology such as GPS sensors; IMUs in vision-based systems with a local coordinate system.

**Name**: Shared Pointer

**Forces/Problem**: How can multiple people in an Augmented Reality environment communicate efficiently?

**Feature/Solution**: Use (gaze) cursors which are also visible to the other players or allow users to leave markers at set points.

**Examples**: Nilsson, Johansson & Jönsson used Shared Pointers in the form of settable icons in all versions of their cross-organizational collaboration application and a form of cursor in the first version.

**Effects/Consequences**: Shared Pointers can allow users to communicate more clearly. This approach requires either a Gaze Cursor or some other kind of directed user input. For consequences of users setting markers, see Directed Focus and Directed Movement (though depending on your application it may not be necessary for the markers to be pointed towards when off-screen).

**Sensors**: If a Gaze Cursor is used, the sensors related to it are needed. If only virtual objects can be selected, special sensors may not be necessary. If real objects can also be targeted, the relative positions and orientations of the users must be taken into account, derived from e.g. IMU data.

**Name**: Voice Commands

**Forces/Problem**: How do you allow user input while keeping the user’s hands free?

**Feature/Solution**: Have the user perform actions by speaking appropriate phrases.

**Examples**: Voice Commands are a standard feature of HoloLens applications and can be seen in games such as *RoboRaid* where a voice command is used to activate a special ability. *Fragments* gives you a choice between Gesture-Based Interaction and Voice Commands, e.g. for examining objects.

**Effects/Consequences**: If implemented well, Voice Commands can greatly enhance user experience. However, if phrase recognition does not work well or the phrases are too complicated, the user may become frustrated. If the application is to be used in shared spaces, loudly talking and recording audio without consent may face social acceptance issues; the system could also pick up other people’s voices and act accordingly, against the user’s wishes.

**Sensors**: One or more microphones.