Experiences with Handheld Augmented Reality

Dieter Schmalstieg[1] Daniel Wagner[2]

Graz University of Technology

Figure 1: The Augmented Reality game "Expedition Schatzsuche" played in the "Landesmuseum Kärnten" in Carinthia, Austria. Left and right: Participants of the test run; Middle: Playing the silent piano game.

ABSTRACT

In this paper, we present Studierstube ES, a framework for the development of handheld Augmented Reality. The applications run self-contained on handheld computers and smartphones with Windows CE. A detailed description of the performance critical tracking and rendering components are given. We also report on the implementation of a client-server architecture for multi-user applications, and a game engine for location based museum games that has been built on top of this infrastructure. Details on two games that were created, permanently deployed and evaluated in two Austrian museums illustrate the practical value of the framework and lessons learned from using it.

Categories: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems - Artificial, augmented, and virtual realities, K.8.0 [Personal Computing]: Games – General, K.3.1 [Computing Milieux]: Computers and Education - Computer Uses in Education

Keywords: Mobile augmented reality, wearable computing, cultural heritage, augmented reality games

1 INTRODUCTION

The work reported here was sparked in 2002 when handheld computing experienced a sudden significant increase in processing power. Personal digital assistant (PDAs), which previously had been designed to run simple organizer software, were now marketed as handheld multimedia appliances and game consoles. As part oft this trend, built-in cameras were quickly adopted as a standard feature for cellphones and other handheld devices.

Even before this new kind of hardware became available, some researchers working on mobile Augmented Reality (AR) had started replacing the cumbersome backpack plus head-mounted display setups with Tablet PCs. An AR application can leverage the tablet form factor to display a video stream from a camera attached to the Tablet PC with real-time augmentations. This "through-the-lens" approach towards AR is cheaper and more convenient for a mobile user than a backpack solution. Tablet PCs offer amble processing power, but compared to the PDA/smartphone class they are not really designed for free-handed operation and still too expensive for casual deployment.

Compared to Tablet PCs, PDAs and smartphones are aiming for a different market. Price, weight and battery life are designed for a large consumer base and mobile – rather than merely portable – operation. Unmodified consumer devices are also surprisingly robust and foolproof given their fragile appearance. However, these desirable properties come at the price of restricted computing capabilities compared to the PC platform. Achieving sufficient performance for AR applications requires careful choice of algorithms and optimized code.

In this paper we describe our work on *Studierstube ES*[1], a software framework for handheld AR developed over the past five years specifically for ultra-lightweight devices. Earlier versions of this framework have been described in [30] and [31]. This paper consists of two main parts: The first part presents previously undocumented details concerning the technical solution, while the second part describes the design rationale, development process and evaluation of a recently built large-scale application: A multi-user treasure hunt game was developed and permanently deployed in two museums in Austria. To our knowledge this application is the largest application involving handheld AR to date.

The paper is organized as follows: In section 2 we discuss related work and compare it to the solution presented in this paper. Section 3 gives an overview of the software architecture of *Studierstube ES*. Specific technical components are described in the following sections: Section 4 gives details on the fiduciual tracking, while section 5 presents the rendering engine. Section 6 gives an overview of the game design and game engine, followed by details on the museum applications in section 7. Section 8 presents evaluation results, and section 9 draws conclusion and gives an outlook to future work.

[1] email: schmalstieg@icg.tugraz.at
[2] email: wagner@icg.tugraz.at

[1] Embedded System

2 Related Work

The Augmented Reality application presented in this paper builds upon a large body of previous research. The two main topics, handheld AR and AR games, are described in the following.

2.1. Handheld Augmented Reality

Handheld devices seem to be a superior alternative for AR - especially for untrained users in unconstrained and non-supervised environments. They are more robust than HMDs and due to the advent of mobile phones and PDAs users are comfortable operating them. Even before the success of the smartphones as mass-marketed items, pioneering projects started using small displays for custom see-through devices. Amselem's work [1] and Fitzmaurice's Chameleon [8] used small tethered LCD displays for location based information. Rekimoto's NaviCam [25] used color-coded stickers to track objects in the environment. Due to the tethered trackers in these early works, the degree of mobility was rather limited. mPARD [24] is a variant using analogue wireless video transmission to replace tethers.

The Transvision [26] project by Sony CSL introduced handheld AR devices for a shared space. Researchers at HITLab later improved this concept [23] with a better user interface and an optical tracking solution re-using the camera needed for video see-through.

From 2000 on, PDAs with wireless networking were considered suitable for thin-client solutions outsourcing computationally intensive tasks such as rendering, tracking and application to a nearby workstation. The Batportal [13] used non-mixed 3D graphics using VNC, while the AR-PDA project [10] used digital image streaming from and to an application server. Shibata's work [28] aims at load balancing between client and server - the weaker the client, the more tasks are outsourced to a server. ULTRA uses PDA for augmenting "snapshot" still images [21].

In 2003 we ported ARToolKit [14] to Windows CE and consequently developed the first fully self-contained PDA AR application [31]. This platform was used in a peer to peer game in [25]. Meanwhile Möhring et al. targeted a Symbian smartphone for mobile AR [22]. The scarce processing power of the target platform allowed only a very coarse estimation of the object's pose on the screen. Later Henrysson ported ARToolKit to the Symbian platform and created a two-player AR game [12] on current-generation smartphones. Several of these projects involve collaborative applications, but not for larger users group.

2.2. Pervasive Games and Augmented Reality Games

With availability of mobile devices to a generation that grew up with video games, pervasive games are receiving increasing attention. In contrast to mobile games, which can be played independent of location, pervasive games are location based services (LBS) that leverage the real world as part of the gameplay. For example, coarse positioning technologies such as WiFi cell id can be used as tactical features in gameplay.

MIT's Mystery at the Museum [15] invites players to play a detective investigating a crime scene. The iPERG project [17] uses a pervasive computing environment including mobile phones, PDAs, backpack setups and stationary PCs. Some of these interfaces incorporate AR. Lancaster University's Real Tournament [34] is a fast multi-player LBS game that uses PDAs to provide every player with live-updated contextual information. Similar technology is used by mobile tour guides such as CyberGuide [18], Lighthouse [6], and Phone Guide [9], which are more educational. Storytelling for education in museums is becoming more commonplace. Some museum interfaces already rely on AR edutainment, such as Stapleton's work [29] or Sweet Auburn [19].

There are several multi user AR games reported in literature, such as Shared Space [4] or AquaGauntlet [2], but most of them are limited to a tabletop environment. Pervasive AR games incorporating more users and multiple locations are more difficult to design and implement, requiring some consideration about authoring tools to create, test and manage the game content in a reasonable way [11][16]. One of the most sophisticated tools to address these needs is DART [20], but this tool requires Macromedia Director and does not run on handheld devices. The work presented in this paper goes one step further by describing design and evaluation of a complete toolset for the creation of location based multi-user AR experiences.

3 Software Architecture

As a foundation for Handheld Augmented Reality, a software framework called *Studierstube ES* was developed since 2003. The framework was originally intended to be compatible with the PC-based *Studierstube* software framework[2], but this requirement was eventually relaxed in favor of optimized performance and memory footprint for handheld devices. The *Studierstube ES* framework (see Figure 2) is available for Windows CE and Windows XP, targeting small form factor devices ranging from smartphones to ultra-mobile PCs (see Figure 3). Experimental versions also exist for Symbian and Linux.

3.1. Handheld device framework

All processing is done natively on the handheld device, so that applications can run independently of any infrastructure and scale to an arbitrary number of simultaneous users. Typical frame rates on smartphones are in the order of 5-20 fps, depending on the content and device. A server component running on a PC was developed specifically for multi-user communication and content management. Clients maintain a wireless networking connection to the server if available, but it can run fully stand-alone too.

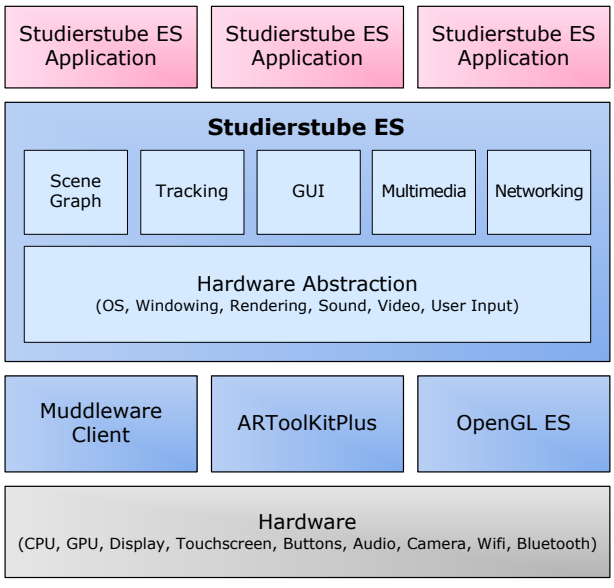

Figure 2: *Studierstube ES* component based framework design

2 http://www.studierstube.org

Figure 3: Smartphone, PDA and Tablet PC are used in the Handheld AR project

The software framework is based on a component design, and allows customizing the runtime environment to the needs of the application and the capability of the handheld device. In particular, memory footprint can be optimized to as little as 500K for a basic system. However, an extensive set of components is available. This component library includes not only AR specific components such as the tracking and rendering toolkits described in sections 4 and 5, but also components for networking, multimedia playback (2D user interface widgets, Flash, 2D and 3D animation, audio, video), networking and scripting (mostly through XML dialects).

Application code is managed through dynamically linked libraries (DLLs), which can be loaded and unloaded at any time. This allows to optimize memory usage and also simplifies code maintenance in a distributed system by downloading code and data from the server at startup. Only a small bootloader, which can be provided on a memory card, must remain on the device.

3.2. Server

One of the characteristics of mobile computing is that connectivity can change over time [3]. Consequently we have designed a server platform that mediates communication among clients and also stores transient as well as persistent client data. A blackboard architecture was chosen for the server implementation [33]. The server, called Muddleware, consists of three main components (Figure 4): a memory mapped database, a networking component communicating with clients and a controller component that can be scripted with a finite state machine (FSM) to enforce rules on the database. The memory mapped database is organized hierarchically with an XML document object model, and the XPath standard is used as a lightweight database query and update language.

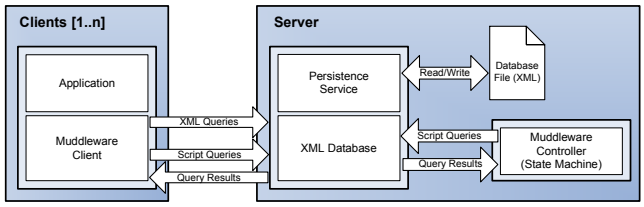

Figure 4: Muddleware components

4 FIDUCIAL TRACKING

As a foundation for registering real and virtual objects, the purpose of fiducial tracking is to find one or multiple markers in the camera image and estimate the corresponding poses. The black-on-white square fiducial markers detected by the Studierstube Tracker are not fundamentally different from other marker trackers developed for the PC like ARToolKit [1] or ARTag [7]. However, the implementation of our tracker has been tuned to provide the maximum tracking quality and performance using the limited resources of handheld devices. While the tracker started out as a modified version of the original ARToolKit code, called "ARToolKitPlus" in [32], the current version constitutes a complete re-implementation from scratch with more efficient methods. The tracking pipeline consists of five basic steps: thresholding, fiducial detection, rectangle fitting, pattern checking, and pose estimation.

4.1. Thresholding

The first step in the image processing pipeline is to convert the input image into a binary image to reliably detect the black and white portions of the fiducial markers. In practice, inexpensive cellphone cameras and variations in image brightness cause severe problems for constant threshold values. Global thresholding is too computationally expensive.

Instead, a heuristic that has proven effective is to use the median of all marker pixels extracted in the last frame as a threshold for current image. If this heuristic seems to fails as no marker is found, the threshold is randomized for every new frame until a new marker is detected. Empirical tests show that after a marker gets lost it takes only a few frames to find a new, working threshold.

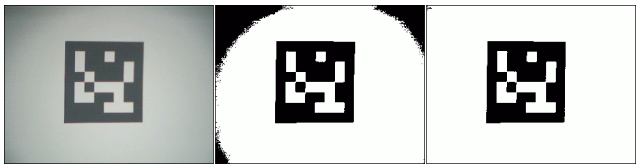

Figure 5: Vignetting. Left: original camera image. Middle: constant thresholding. Right: thresholding with vignetting compensation.

An additional problem in thresholding is that cellphone camera often exhibit strong vignetting, i. e., a noticeable radial brightness falloff (see Figure 5). Thresholding such images with an image-wide constant value yields unusable results, as a marker close to the border will be effectively hidden in the dark area. To prevent this, vignetting compensation incorporates the radial brightness into the per-pixel threshold value. The radial fall-off is determined during camera calibration and is specified numerically rather than using an image mask in order to minimize memory bandwidth usage. Thus, vignetting compensation adds only a minimal performance penalty during thresholding.

4.2. Fiducial Detection

The tracker detection system to find fiducial markers (see Figure 6) is based on simple edge following: As a first step all scanlines are searched left to right for edges. A sequence of white followed by black pixels is considered a candidate for a marker's border. The software then follows this edge until either a loop back to the start pixel is closed or until the border of the image is reached. In case of a closed loop, the contour is stored as a polygon and considered for further processing. In case the border is reached, the edge is discarded. All pixels that have been visited are marked as processed in order to prevent following edges more than once.

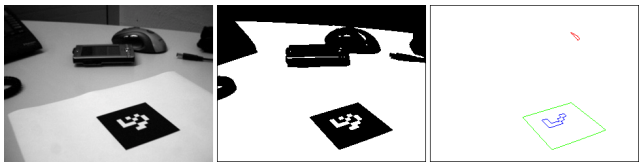

Figure 6: Left: Source image; Middle: Threshold image; Right: Three closed polygons as candidates for rectangle fitting.

4.3. Rectangle fitting

In this step, rectangles (more precisely, quadrilaterals after perspective projection) need to be identified among the previously identified polygons. Method based on edgel sorting, such as used by ARTag, are robust against partial occlusion, but are too computationally expensive for current handheld devices. Instead, the rectangle fitting in the tracker relies on finding polygons with exactly 4 corner points and mostly straight lines between corners.

Since a physical marker might not be perfectly flat and because the radial distortion can warp the camera image considerably, it is necessary to use a relaxed method instead of searching for perfectly straight lines. Conventional line fitting is not suitable since the polygon contour would have to be segmented into corner points and candidate lines first.

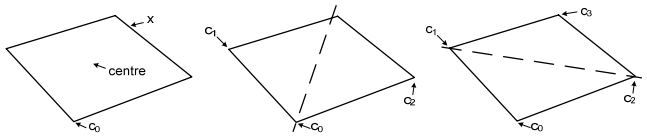

Figure 7: Example for fitting a rectangle to a polygon.

Instead an iterative process searches for corners until the whole contour has been searched or more than four corner points are detected. Only if exactly four corners are detected the contour can be a rectangular marker and is stored for further processing.

A first corner point c_0 is selected as the contour point that lies at the maximum distance to an arbitrary starting point x of the contour (Figure 7, left). For a rectangle this operation always detects a corner point and is robust against distortions. The center of the rectangle is estimated as the centre of mass of all contour points. A line is formed from the first corner point and the center (Figure 7, middle). Further corner points c_1, c_2 are found on each side of the line by searching for those points that have the largest distance to this line. These three corners c_0, c_1, c_2 are used to construct more lines and recursively search for additional corners. In the example (Figure 7, right), only one more point c_3 is found. The length of a line is used to determine the threshold for the minimum distance of a new candidate corner point from the line. Points closer than the threshold are not considered. The algorithm converges if all contour segments have been searched or more than 4 corner points are found. The corners of rectangles are sorted in clockwise order. Finally candidate rectangles with an area that falls below a threshold are suppressed.

4.4. Pattern Checking

After detecting quadrilateral polygons, these polygons need to be checked to contain valid marker patterns. First a marker's interior region is resampled into a normalized arrays of pixels. For perspectively correct unprojection, the homography matrix is computed from the markers' corner points, which are assumed to form a rectangle. In contrast to the simple L2-norm template matching of ARToolKit, which is too computationally expensive, we use 2D bar code techniques with forward error correction - either BCH (Bose, Chaudhuri, Hocquenghem) encoding or DataMatrix. BCH codes rely on cyclic redundancy checks for forward error correction. A 6x6 pattern in the markers yields 36 bits of raw information, 12bit (1/3 of the raw information) of which are used to distinguish 4^{12}=4096 different markers, while the remaining 2/3 of the raw information are used as redundancy to correct for partly incorrectly observed patterns. Bar codes are directly generated from the id, and no training phase like for template matching is necessary.

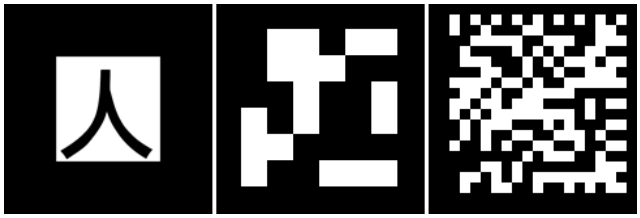

Figure 8: Marker types. Left: Template markers; Middle: BCH-markers; Right: DataMatrix markers.

DataMatrix[3] codes are an ISO standard for 2D bar codes. Code patterns can be up to 144x144 pixels and store up to 1558 bytes, allowing to encode for example a complete URL or a small binary data set. Such large patterns require higher image resolutions and more time to decode successfully. However, after a successful initial marker decoding, an application may choose to continue tracking the marker rectangle at real-time frame rates without decoding the marker pattern again. This is particularly useful as BCH and DataMatrix markers (see Figure 8) can be mixed in the same image.

4.5. Pose estimation

The final step in the marker tracking pipeline is the estimation of the camera pose from the corner points. For a stable estimation, the position of the corner points must be as accurate as possible. First, sub-pixel accurate corner point estimation is performed using a Harris corner detector, initialized with the original corners from the rectangle fitting, which are only accurate to one pixel. The resulting corners are unwarped using the intrinsic camera parameters (calibrated offline using the Matlab calibration toolbox) to compensate for radial distortion.

The tracker allows to choose from various pose estimation algorithms. An initial approximation of the camera pose is derived from the homography estimation, and used to initialize a non-linear optimization. The first approach, direct optimization for mimimal reprojection error is the fastest method, while the Robust Planar Pose (RPP) estimator [27] systemtically avoids local minima, but is more computationally expensive.

Besides tracking of independent markers, multi-marker sets attached to a rigid body are supported. Multi-markers yield improved stability of the pose estimation when multiple markers are visible. Multi-marker tracking works by first estimating the poses of all visible markers independently, then computing a combined pose estimate by iteratively minimizing reprojection error of all observed marker corners.

4.6. Performance considerations on embedded devices

Fixed point The lack of hardware floating point units is probably the single most important performance issue for numerically intensive algorithms on embedded devices. Floating point emulation is not feasible for real-time applications, so all computation needs to execute using fixed point arithmetic. The limited precision implied by the use of fixed point implies to choose only numerically well-behaved algorithms and carefully optimize the computations to avoid exceeding the precision bounds. This tuning was done through code instrumentation, which enabled us to choose the required precision and range for every variable and code fragment individually if necessary. Moreover, many mathematical functions, such as polynomial functions for lens undistortion, trigonometric functions and

3 http://datamatrix.kaywa.com

perspective divisions, were replaced by lookup tables and interpolation.

Pixel formats Bandwidth on embedded devices is so scarce that image format conversations incur a significant performance penalty even for relatively low resolution images, such as VGA (640x480). Fortunately, the most common format on embedded devices is YUV12, which contains an 8bit luminance component directly suitable for image processing. Other common format such as RGB565 are converted on the fly using lookup tables (assuming only a subset of the pixels must ever be examined), to minimize the impact on performance.

4.7. Performance measurements

To test the tracker' performance for practical applicability, benchmarks on several handheld devices were performed. These tests compare tracking performance with different numbers of visible markers, which is the only criterion affecting tracking speed in practice. Contrary to expectation, we found that the size of the marker does not influence the tracking speed, since the edge following adds only very little to the overall calculation time.

The tracking computation is primarily CPU bound and not influenced much by the operating system. A number of Windows CE based devices were chosen, and also compared to a 2GHz PC. Table 1 lists the devices used in the test. All devices except the PC employ an ARM9 compatible CPU.

Brand name	OEM name	CPU	MHz
i-mate SP5	HTC Tornado	TI OMAP850	200
HTC MTeor	HTC Breeze	Samsung S3C2442	300
HTC TyTN	HTC Hermes	Samsung S3C2442	400
Gizmondo	-	Samsung S3C2440	400
T-Mobile MDA Pro	HTC Universal	Intel XScale PXA270	520
Dell Axim X51v	-	Intel XScale PXA270	624
PC	-	Intel Core Duo	2000

Table 1: Performance characteristics of the devices used in the tracking comparison

The following performance figures were obtained by using either the Microsoft or Intel ARM compiler with full optimization, whichever performed better. On the PC, native floating point support was used. Table 2 compares average tracking times for scenarios with one marker (M1) and multi-marker arrangements with 4 markers (M4) and 10 markers (M10).

Device	M1	M4	M10
i-mate SP5	13.3	66.4	234.1
HTC MTeoR	10.2	44.6	153.3
Gizmondo	8.5	34.5	122.7
HTC TyTN	8.3	34.9	128.1
MDA Pro	6.0	24.1	83.4
Dell X51v	5.1	20.7	69.8
PC	0.43	2.77	8.3

Table 2: Benchmarks performed on images with one, four and ten markers (all measurements in ms). M4 and M10 were tracked with a multi-marker set of 12 markers of which four and ten markers were visible respectively.

As expected the tracking performance increases linearly with the CPU clock. The tested ARM-powered devices process 31.12±1.76 frames per second at a normalized speed of 100 MHz.

5 RENDERING

5.1. Immediate mode rendering

Besides tracking, 3D graphics rendering is the second critical enabling technology required to build a working AR system on a handheld device. Unfortunately, dedicated graphics processing units (GPUs) are only slowing becoming available on handheld devices.

This can be attributed to increased cost and power consumption, to which the handheld device market is particularly sensitive. Those GPUs that are available on handhelds today are tuned for highest performance and smallest transistor count, but not for a rich feature set, full programmability or convenient developer interface. Besides, the market demands that rendering must also be possible using only the CPU.

This situation is reminiscent of the beginning of GPU hardware on personal computers about a decade ago. The imposed challenge is that we would still like to leverage modern graphics algorithms and techniques rather than duplicating outdated approaches from the past.

Graphics standards are a prerequisite towards this goal. The most widely accepted 3D graphics standard on embedded devices is OpenGL ES (embedded subset), followed by Microsoft's Direct3D Mobile (D3DM). Both standards target the same embedded GPU hardware, but unlike D3DM, OpenGL ES is available for multiple operating systems and highly optimized software implementations exist.

While OpenGL ES constitutes a solid foundation for graphics programming, it is significantly more restricted than OpenGL. Existing OpenGL software cannot be trivially ported to OpenGL ES because of the restricted feature set. Moreover, to support software implementations on embedded CPUs, OpenGL ES uses a 16 bit fixed point format for all numeric operands. This requires developers to take care in the modeling process as large models can create numeric overflows.

The typical bottleneck of the graphics pipeline depends on the level of GPU support. Current embedded GPUs such as the Intel 2700G or the nVidia GoForce 4500 accelerate only the pixel stage, and the vertex stage, which must be computed in the driver on the CPU, quickly becomes saturated. Consequently, it is difficult to balance rendering in order to make the best use of both a pixel-limited software renderer and a vertex-limited hardware renderer. Moreover, texture upload to an embedded GPU is notoriously slow due to the scarce bandwidth, which has a noticeable performance impacts for video texturing used in AR applications.

	Pure SW	Mixed	Pure HW
Vertex Stage	Software	Software	Hardware
Pixel Stage	Software	Hardware	Hardware
Typical Limits	Pixels	Vertices	-
Framebuffer Access	Yes	No	No
Fast Texturing	No	Yes	Yes

Table 3: Comparison of embedded graphics implementations, based on their support of vertex and pixel stage

An efficient graphics program therefore must implement multiple render paths for making optimal use of the strengths of each kind of renderer. 2D graphics, such as drawing video background or user interface widgets, must be drawn as textures when using hardware 3D – such operations are actually faster when copying directly to the memory-mapped frame buffer of a

software renderer. Developers therefore face the challenge of balancing the load imposed by their graphics content so that both hardware and software rendering is possible.

5.2. Scene graph

Scene-graphs are the most common architectural pattern for high level rendering engines. Unfortunately, there is a limited selection of rendering engines for handheld devices, and no general-purpose scene graph was available when the work on this project started. We therefore investigated the porting of an existing scene-graph for the PC platform, Coin3D, to Windows CE. After several significant modifications, including the stripping of unneeded features and introducing custom memory management, a working prototype of Coin3D on Windows CE was completed and tested. Unfortunately, while the library was fully functional it did not meet our expectations. The minimal version still had a memory footprint of over 10MB, which exceeds what embedded devices can comfortably handle. Performance was also unsatisfactory due to many built-in convenience features such double-dispatch graph traversal or automatic event propagation on scene graph updates.

As a remedy a new custom scene graph was developed which borrows many design ideas from Coin3D, but does not share its overhead. This scene graph contains only the features that are relevant for rendering on embedded devices. It retains many important features such as reflection, fields and field connections, but has a very small memory footprint, and the performance overhead of traversal is negligible.

The scene graph library provides a wrapper for both OpenGL ES and D3DM. It can therefore target all currently available embedded graphics platforms, while offering important graphics programming concepts that are painfully missing from the underlying toolkits. For example, neither of the low-level graphics libraries support stacks or queries for graphics state variables such blending, depth sorting or transformation matrices. Stacks and a corresponding separator group node are implemented in the scene graph, which makes efficient modeling possible. Another important abstraction is the geometry buffer node which passes geometrical data to the underlying toolkit. It lets the scene graph decide whether to store vertex data in system or video memory, and thus hides the differences between specifying vertex buffers in OpenGL ES and the rather inconvenient "Flexible Vertex Format" in D3DM. In the absence of programmable shading hardware, many special rendering effects can be scripted from within the scene graph using multi-pass techniques.

Models are stored using either a custom XML based or a compact binary file format. Efficient content creation is ensured by providing a suitable conversion of models from Maya and from the VRML file format. Skeleton animations from content creation tools are converted to per-vertex keyframing for CPU-friendly playback.

5.3. Performance evaluation

Native performance of *Studierstube ES* was recently evaluated with a selection of smartphones. The main components competing for computation time are tracking and rendering, but relative performance is difficult to measure if a GPU is available and can run concurrently with a CPU.

The devices in Table 4 were used in the benchmarks.

Device	CPU	MHz	GPU
HTC Tornado	TI OMAP850	200	none
HTC Excalbur	TI OMAP850	200	none
Palm Treo700W	Intel XScale	312	none
Motorola Q	Intel XScale	312	none
Motorola Q9	TI OMAP2420	330	on CPU

Table 4: Devices used in the system benchmark

Figure 10: Test models rendered for benchmarking. Left: Textured cube; Middle: Venus of Willendorf; Right: Model of a car.

The Treo was used with 312MHz and also overclocked with 520MHz to assess the influence of CPU speed. The Q9 was used with and without the GPU activated to assess the influence of the GPU. On all devices, capturing the video from the camera runs in a separate thread that competes for processing time with the main thread. To benchmark realistic application performance, we adopted a strategy of adding increasing load to the application:

- Empty frame: no video, no tracking, no rendering
- Just video capture and conversion to RGB565
- Video rendering added
- Tracking of a single marker added
- Rendering of a textured cube added
- Rendering of a medium scale model (textured Venus statue, 2625 triangles)
- Rendering of a large scale model (Toyota car, 25652 triangles)

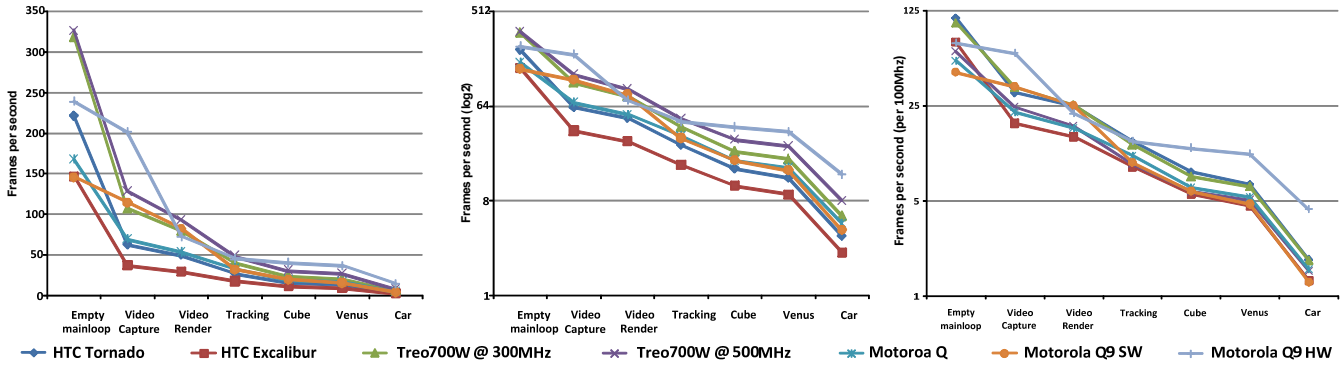

Figure 9: Performance results. Left: Linear scale; Middle: Logarithmic scale; Right: Linear, normalized to 100Mhz.

Figure 9 shows the performance results. Most devices are severely hurt by the task of video capturing and video rendering. This shows the scarcity of memory bandwidth and also the effect of video format conversion and texture upload if a GPU is used. The differences induced by tracking again turn out to be proportional to clock rate.

Rendering the cube and Venus (left and middle images in Figure 10) model perform similarly, apparently dominated by pixel fill rate. In contrast, the car model (right image in Figure 10) demonstrates the effect of a GPU on the vertex state of the pipeline – Excalibur and Tornado can no longer deliver interactive speeds, while the Q9 with its vertex-acceleration delivers almost the camera rate of 15fps.

6 GAME DESIGN AND AUTHORING

6.1. Physical affordances of handheld AR

The magic lens metaphor afforded by handhelds imposes very specific constraints to interaction design. The device must be held at a distance of about 50cm, with the camera normally tilted downwards, to allow for prolonged use without significant fatigue and also to let the user focus on the screen. The field of view defined by the screen is therefore very limited. This implies that in order to observe a physically large environment, the device needs to be frequently moved or rotated. Ergonomic constraints and the necessity to keep a line of sight to the display effectively limit the type and amount of possible movements of the handheld. While rotation and movement with the supporting arm are quick, moving the device through physical walking is more disrupting since it is often difficult to keep the screen in view while physically navigating the environment.

A game design should therefore aim to minimize such physical movements. Using a touchscreen or built-in mini-joystick allows holding a device still while interacting. However, we have observed that the enjoyment of physically navigating the environment is key to the appeal of handheld AR games. In that respect, fiducial markers are not only necessary evil but also create affordances allowing a user to identify locations in a larger environment, where AR interaction is possible. AR interaction in the strict sense takes place by interacting with an augmented video stream. However, since the handheld is so small, a lesser form of AR interaction is also possible by presenting non-registered content on the handheld after identifying a marker.

6.2. Game design

As a real-world application for the multi-user handheld AR infrastructure described in sections 3-5, a location-based treasure hunt game in a museum was developed. Museums are aiming to make their exhibitions more attractive to high school students. Students are typically visiting in scheduled groups, which allows to manage the logistics of running the game. Working with a museum allows researchers to recruit museum staff for assistance with the game content, which requires knowledge about the museum exhibits.

The overall goal was to link selected exhibits into an engaging story-driven game. The objective is to solve a sequence of puzzles and other tasks associated with the exhibits. Tasks range from spatial interaction using handheld AR to more conventional multiple choice questions.

We wanted to use the multi-user aspect of the game to raise the motivation. Students are therefore divided into multiple teams. Within a team, students work cooperatively to solve the tasks, while every team competes with other teams for the highest score.

Moreover, creating unique content for each player increases the work required for content creation enormously. A game of 10 players and 20 minutes active game play equals 200 minutes of unique content. The size and number of teams can be varied to make the creation of content more economical, by re-using content across teams. This team organization together with the game engine tools and interfaces to standard content creation packages, described in the next section, made it tractable to create a sizeable game with constrained resources.

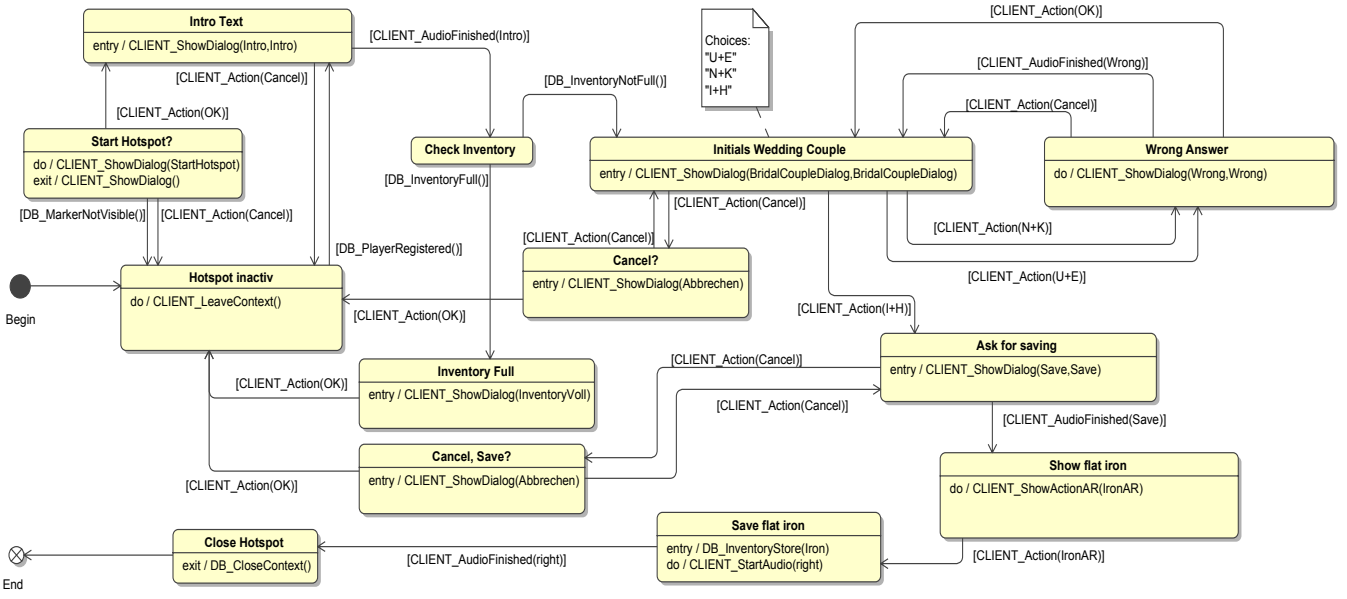

Figure 11 : Games usually start in an idle state. An approaching player is introduced via a dialog box and audio message. Here the user has to answer a multiple-choice question (state “Initials Wedding Couple”). If the player answers correctly she is rewarded with an item. In this case, a 3D model of a flat iron flies from the exhibit to the handheld and is placed in the player inventory.

6.3. Game engine

A game engine for treasure hunt games was devised, which plugs into the *Studierstube ES* infrastructure. We designed the treasure hunt games to resemble classic adventure games which require a player to solve puzzles and find and apply items. These game mechanics are rather simple and rely on the appeal of the real world environment, which in case of a museum is full of noteworthy and unusual items that can be leveraged for the game play. The engine itself is set up to interpret script code which is easy to write and debug even by non-experts, and special code developed in C++ is only needed for very specific games.

The game engine consists of software components added to the server and client part of the *Studierstube ES* architecture. The server part is an extension of the Muddleware controller, capable of executing multiple threads with one finite state machine, one for each team and exhibit. Each FSM listens to specific elements on the XML server, waiting for client activity at the exhibit. When an exhibit is visited, the FSM starts sending actions to the client by putting commands or state updates into the XML database and waiting for reactions from the client. The game engine plug-in contains pre-defined structures in the XML database, such as an inventory of items the player has collected, and corresponding convenience functions to manipulate the database from within the FSM.

The game client runs as a loadable application on top of *Studierstube ES*. At startup it retrieves a list of available exhibits from the XML database. When the user approaches an exhibit, the user interface and code module for the associated tasks is pushed to the client. The game engine currently supports four different types of tasks: simple 2D GUI elements, Flash animations, scripted AR content (for example, use the magnifying lens to discover something) or custom AR content programmed in C++.

Game logic and game state is normally kept in the XML database controlled by the FSM. This makes the game robust against client failure, since the server operation cannot be permanently disrupted by client malfunction or battery failure. However, we found it necessary to allow executing the FSM driving the game logic on the client in cases where wireless connectivity to the server is not possible. In these cases only the score for a completed exhibit is synchronized with the server-side database as soon as the client becomes online again.

6.4. Authoring tools

FSMs are imported in the XMI format, which is a common format for describing state machines in commercial visual programming editors, such as MagicDraw UML[4]. The runtime system can parse the XMI file (an XML dialect), and can operate directly on the resulting document object model. Figure 11 shows a state chart developed for the game described in section 7. During several months of content creation on this game, the combination of various tools, most importantly MagicDraw and Maya for 3D modeling proved to be very efficient despite the fact that our time budget did not permit the development of a custom game editor.

Figure 12 shows the server inspector. This debugging tool uses XPath to inspect and manipulate the XML database. Besides browsing the game state, an operator can manipulate the current states of ongoing games, restart individual exhibits or give items to players. This is useful not only as a debugging aid but also for a game master to control the game and assist player in case of unexpected circumstances, which always must be expected in a live game.

4 MagicDraw UML: http://www.magicdraw.com

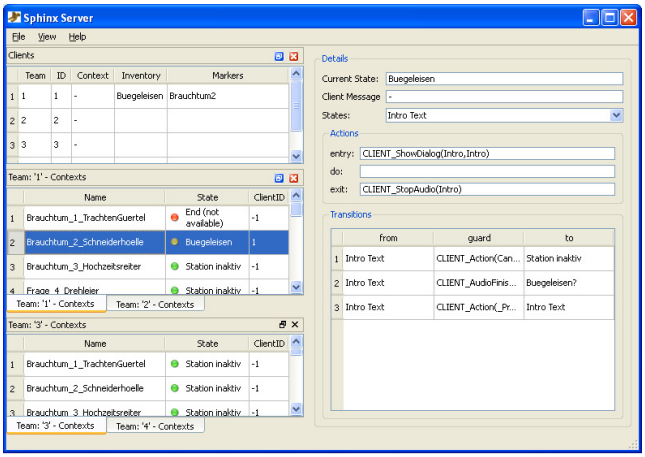

Figure 12: Server user interface (team specific log history of state transitions and events are hidden)

7 Museum Applications

7.1. Case study "medien.welten"

In 2006 a first field test for handheld AR in a real museum was developed. It was designed for a target group of high school students aged 12-16. The game was set in the exhibition *medien.welten* situated at Technisches Museum Wien in Vienna, Austria. This exhibition illustrates how different media were preserved and transmitted since antiquity.

The game prototype focuses on three exhibits linked through an espionage story set in World War II. The exhibits have to be visited by the players in a certain order to achieve the game objectives. Players first receive a briefing at a checkpoint terminal, where the objectives are explained and the handhelds (Dell Axim X51v) are handed out (Figure 13, left). A group of players receiving one handheld usually consists of two or three students. The checkpoint terminal and handhelds show a map of the exhibition, highlighting the relevant task locations. The map also indicates the current position of the players and lists already solved and remaining tasks.

The first task is a live size radio direction finder used to detect and record radio messages from mobile transceivers in the field. The operator had to manually turn the antenna to home in on the signal and then follow it to record it. A special electronic guide was aiding this task by producing characteristic sounds when turning the antennas near the exact signal direction.

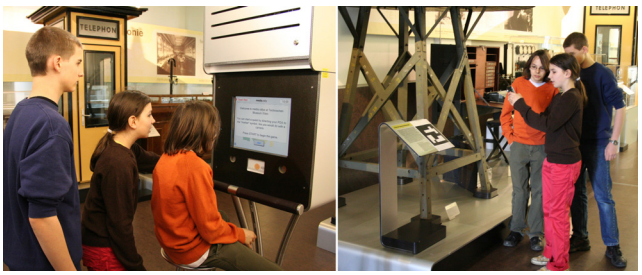

Figure 13: Left: The game starts at the checkpoint terminal with an introductory message. Right: Players try to orient their handheld device to tune in on the incoming message at the radio finder

The radio finder game (Figure 13, right) uses AR to demonstrate the homing task. The handheld has to be physically

moved around the exhibit to find and hold the exact signal direction. For that aim, a virtual compass is superimposed onto the lower platform of the exhibit, and sound indicates the deviation from the exact signal direction. The sound depends on the direction and the angle of deviation – intervals between beeps get shorter when closer to the signal. When the players are close enough to the exact signal direction, the characteristic beeps of a Morse message are played. Once the players have found the right position (which is slightly drifting over time), the handheld must be kept in this position until a progress bar indicates that the entire message has been received.

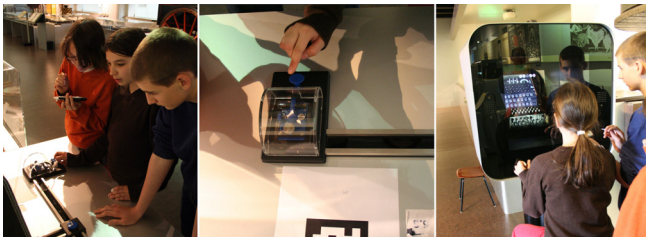

Figure 14: Left: Translating the Morse code at the Morse exhibit; Middle: The Tangible interface of the Morse exhibit, an old-style push button; Right: The virtual show case containing the Enigma

The next task presented to the players is that the received Morse message must be decoded. To solve this task, the players must visit the Morse exhibit, a specially designed hands-on exhibit consisting of a computer terminal with a physical Morse pushbutton (Figure 14, left and middle). Using the pushbutton, the players must input the Morse code received earlier. The exhibit translates correctly input Morse code into the corresponding text on the terminal screen.

During the game, the handheld plays back the previously recorded Morse code. Players can listen several times to the sound sequence corresponding to a single character. Once they have identified the combination of short and long beeps, one player types them with the pushbutton and then reads the translated letter from the display. The player operating the handheld uses the virtual keyboard to enter the letter, and then the next Morse code is presented. This process is intended to encourage the collaboration between the players.

When the players have completed this task, they learn that the translated message makes no sense, but looks like random letters. This is because the message was encrypted with the Enigma machine, and needs decoding. The players have to move to the Enigma exhibit (Figure 14, right) to decrypt the message. The exhibit shows a real Enigma embedded in a Virtual Showcase [5], a mixed reality display combining real artifacts with projected imagery through mirror optics. A trackball enables the visitors to operate the virtually overlaid Enigma without touching the real one.

For the game, the Enigma exhibit is switched to decryption mode. The players have to begin by setting the day key on the Enigma as instructed by their handheld. Then they decrypt the message letter by letter. One player operates the Enigma while another records the plain text on the handheld. The players learn that this collaborative process resembles the way an Enigma was actually operated in the field.

After solving the last task, the players return to the checkpoint, where the results of their performance are displayed. The checkpoint terminal shows which mistakes occurred and what percentage of the message was revealed.

7.2. Integration of hands-on exhibits

The hands-on exhibits featured in *medien.welten* use tangible interfaces to explain certain technologies. While they were originally designed for stand-alone operation, they can also be set to present a certain task when approached by a player. In that way, the environment is responsive to the player beyond the through-the-lens experience with the handheld, heightening the illusion of a mixed reality environment. Figure 15 shows the integration of hands-on exhibits and the checkpoint terminal into the game infrastructure. All 2D interfaces on the exhibits, terminal and handheld were implemented in Flash.

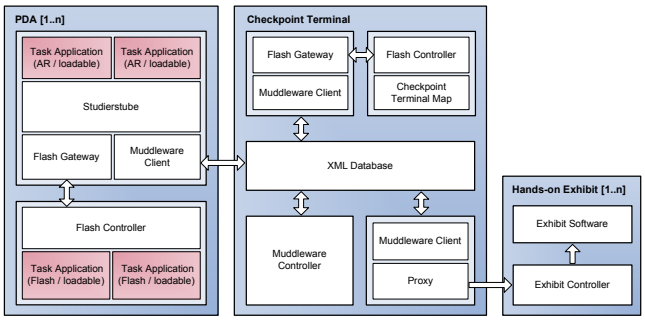

Figure 15: Integration of terminal PCs and hands-on exhibits into the MARQ prototype developed for the Technisches Museum Wien

Depending on the state of the game, *Studierstube ES* on the handheld shows a 3D scene or delegates the interface control to the Flash controller. The latter uses ActionScript to communicate user input to the Muddleware server. The checkpoint terminal is a PC hidden in a console, which usually runs the museum's online information system for visitors. For the game, the terminal runs the server and a Flash application showing the team's overview map and game state.

Hands-on exhibits are also controlled by hidden PCs linked to the museum network. The control software of these exhibits was modified to allow externally selecting of the mode of operation through the server. The hands-on exhibit presents the game interface when a player approaches, while presenting a standard interface for other visitors.

7.3. Case study "Expedition Schatzsuche"

A second, larger game, *Expedition Schatzsuche* was developed in early 2007 for Landesmuseum Kärnten (Carinthia State Museum) in Klagenfurt, Carinthia, Austria. *Expedition Schatzsuche* is now permanently available there for booking by high school visitors. Landesmuseum Kärnten focuses on traditional Carinthian folklore and historical artifacts.

The game encompasses a selection of 16 exhibits and links them into multiple story arcs. Like *medien.welten*, the objective of the game is to solve a quest composed of puzzles and other tasks associated with the exhibits. At the end of each story arc, consisting of 3-4 exhibits, the players receive a piece of a treasure map. Players are divided into teams to compete for the highest number of tasks solved in a given time.

For Landesmuseum Kärnten, the Gizmondo (see middle image in Figure 1) was selected as the handheld device. The Gizmondo is a Windows CE based game console with a 400MHz ARM CPU, 64MB RAM, GoForce 4500 GPU, a screen size of 320x240 pixels and Bluetooth. Its robustness and form factor make the Gizmondo an ideal device for public deployment. A Bluetooth

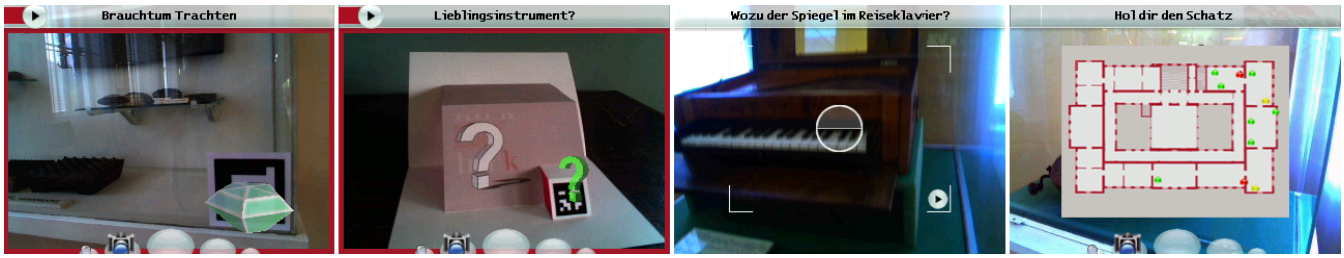

Figure 16: Expedition Schatzsuche; Left: Virtual diamonds telling the state of a hotspot (green: free, yellow: in use, red: already solved); Second: virtual questions marks use the same color codes as the diamonds; Third: Solving a task by taking a picture of the wanted item; Right: map of the museum showing all hotspots and their availability.

network was installed in the museum to support networking of the mobile devices.

The exhibits were selected by museum staff, who also wrote explanatory texts and designed the tasks that the players have to master. We found that players are generally unwilling to read on-screen text, so all explanations were recorded with a professional actor. This measure significantly improved the coherence of the story arcs and the appeal of the game.

Since many of the historical artifacts, such as traditional household appliances, are no longer familiar to the students, the puzzles presented to the players are linked to the original purpose of a particular exhibit. Several of the items look rather unspectacular, so they are likely to be missed when exploring the museum without a guided tour. In such cases an animation overlaid on top of the item using AR inspires the understanding of the students. Gameplay requires the students to answer questions concerning exhibits, applying specific items they have "collected" elsewhere or taking a picture of a specific item. A few tasks even involve demonstrating dexterity or coordination, although these tasks must be kept simple to avoid frustration.

All story arcs start at a checkpoint in the entrance area of the museum, where a big flat screen shows a map of the museum and the progress of the game. After the last piece of the map has been delivered to the checkpoint, the handheld displays a map that reveals the location of the secret treasure in the museum. In the following we illustrate the game by describing two of the story arcs.

Brauchtum (folklore) This story arc about traditions in Carinthia incorporates three exhibits and focuses on wedding customs. The first puzzle (Figure 17, left) involves deciphering the initials of a bridal couple on a traditional belt. For a correct answer, the player receives flat iron, which must be brought to the second exhibit, a tailor's workshop (Figure 17, middle). The tailors happen to miss a flat iron, so the player can help out. Gratefully the tailor rewards the player with a wedding suit. In the next room the player finds a picture of a wedding rider (Figure 17, right). The story continues by explaining the wedding rider's duties as a messenger inviting the wedding guests. The player can assist by providing the wedding suit required by the rider, and is rewarded with a piece of the treasure map.

Music This story arc contains four exhibits (see Figure 18) of which two – the silent piano and the organ – are implemented using custom AR interactions implemented in C++. The story starts with a silent piano (Figure 18, left) that was used by novice pianists to practice without disturbing others. The handheld device invites the player to learn a short piece of music. First only one note is played by highlighting it on the real keyboard, which the user has to repeat. In the following steps more and more keys have to be pressed, until the user can play a complete musical sequence. As a reward the player receives a sheet of music. At the Mandora (a bass lute, Figure 18, 2nd image) exhibit, the player is asked to present some notes to hear a composition played on the instrument. In return the user receives a "Stimmbogen" (crook), which is used to modify the resonance characteristic of a horn. At the "Wiener Horn" (Figure 18, 3rd image) the player learns that the horn can play a large range of notes when applying different crooks. After applying the crook the horn plays a song and the user receives a pair of bellows. The bellows belong to an organ shown in the same room. To play the organ (Figure 18, right image), an assistant must pump the bellows to provide the necessary air supply. The user is cast into the role of the assistant and must keep the virtual bellows moving, or the music will stop. This is done by selecting the bellows in turn and pressing an action key on the handheld to pump. After providing air pressure for a short piece of music, the player receives a part of the treasure map.

Figure 17: Hotspots of sequence "Brauchtum. Left: the player retrieves the flat iron for finding out the initials of the wedding couple; Middle: the player puts the iron into the tailors' oven; Right: the player learns about the wedding rider and finishes the sequence.

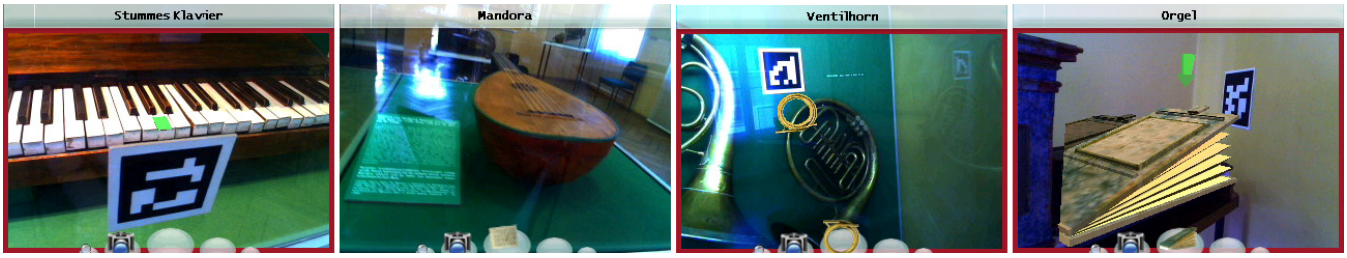

Figure 18: Hotspots of sequence "Music": Left: playing the silent piano; Second: listening to the Mandora; Third applying the "Stimmbogen" to the horn; Right: keeping the organ playing by pumping up the bellows.

8 Evaluations

Both case studies reported in section 7 were evaluated through supervised test runs with high school students. The evaluation consisted of an upfront interview, to find out about users' interests and expectations, observation during the gameplay, logging data from the game and another semi-structured interview conducted after playing, concerning the acceptance of the game and personal experiences of the players. The *medien.welten* game was tested by eight groups from different schools (encompassing 19 students in total, aged 12-15). The results were used in several iterations for design improvements. The evaluation of *Expedition Schatzsuche* was set up similarly. We observed two runs of six simultaneous players each (overall six male and six female, all at age 12). Each of the subjects had been given their own handheld to play the game. Generally subjects had little or no prior experience with handhelds. However, all of them were familiar with mobile phones and some with portable game consoles.

Expedition Schatzsuche already incorporates design changes based on lessons learned from the earlier *medien.welten*. In the following we will discuss general observations as well as discuss differences between the two game settings where it is relevant.

8.1. Gameplay and interface

The *medien.welten* game was limited to 15 minutes overall and 5 minutes per task. The average task duration is given in Table 5. The time limit was not exceeded except for one group, which had severe difficulties in navigating the exhibition space without help. However, for most students the on-device overview map was sufficient and was used extensively for localization of the exhibits, *"look, we are here – we have to go there."* However, from observation we had the impression that the total time permitted was rather too short, because of the requirement to find the exhibits and comprehend the task. Consequently, no time limit per exhibit was enforced in *Expedition Schatzsuche*, which removed problems related to personal pace.

As Table 5 indicates, the task incorporating the Morse exhibit seems to be more difficult to solve correctly than the other tasks. This is emphasized by the high average time spent on this exhibit and the higher error rate. Interestingly, the Morse task was found to be most enjoyable. The two main reasons given by the subjects were using the tangible interface in combination with the handheld, and the sounds of real Morse signals.

The survey data suggests that the overall acceptance of the game was fairly good, and it was perceived better by male students than female students, probably due to their acquaintance with portable game consoles. In contrast to the technical affinity of male students, female students left the impression that they had faster insight in understanding the tasks.

Quest Application	Avg. duration	Average Score
Radio Detection Finder	1:44 min	9 of 9
Morse Telegraph	3:01 min	7 of 9
Enigma	3:33 min	9 of 9
Mean	2:44 min	-

Table 5: Result of statistical analysis from the logging data

The fact that typically only a teacher without AR experience is available for supervision puts extraordinary requirements on the robustness and ease of use of the handheld devices. We aimed at making the user interface as intuitive as possible, and provide detailed task descriptions at any time. Still it turned out that some tasks need to be explained in more detail to be fully understood – especially those employ unusual interaction mechanisms, in particular AR.

One of the main results of the observation of *medien.welten* was that the users' motivation is high, if they are at ease with the interface technology. However, frustration can quickly arise, if the user feels uncomfortable or experience malfunctions.

8.2. Tracking and augmentation

An important lesson learned from *medien.welten* was that users do not have experience with marker tracking and needed a certain time for getting acquainted with the interface technology. For *Expedition Schatzsuche*, players were more carefully instructed about the tracking technology, and visual tracking feedback was added in the form of a red frame that appears on the screen when the tracking system detects a marker. These measures together with the switch to the Gizmondo device which can be held steady with two hands while pointing the device's camera to an instrumented exhibit noticeably improved the performance of the students.

In contrast to the tracking itself, the interaction with the game interface and the comprehension of the game tasks posed no problems for the test subjects. The mechanism of activating an exhibit by aiming at a marker and pressing a button was intuitively accepted.

In the interviews the users rated playing the AR games in *Expedition Schatzsuche* requiring dexterity as more difficult than the multiple choice questions, which is backed up by our observations. Players stated that they had no problem with playing the "silent piano" after finding out "how to" and "what to do". However, operating the organ's bellows demanded their full attention. Consequently some players discontinued aiming the camera at a marker while playing (which had no negative effect except for intermittent augmentation).

Using the currently available marker tracking technology is not trivial in a museum. Placement of markers is restricted, since they cannot be directly attached to historical artifacts, yet they need to

be of reasonable size and in view of the camera. More tracking problems are caused by the dim light and reflections from windows and showcases. Many exhibits are very old and may only be presented under restricted lighting conditions. Some interesting areas in the exhibition could not be integrated due to improper illumination or the inability of proper marker placement. As computer vision on handheld and embedded devices matures, pre-trained natural feature tracking that exploits the static scene afforded by an exhibit should become feasible even on handheld devices.

8.3. Enjoyment

The overall reactions to AR were very positive. Table 6 shows scores on questions concerning enjoyment collected from the *medien.welten* trial. The interviews revealed that subjects demanded to have "more handhelds", "more action", "longer messages to decode" and "more adventure" in the game. Only in one case students were not convinced that the game should be extended to the whole exhibition. As their main reason they mentioned the complexity of the tasks, which turned out to be caused by insufficient instructions.

Interview Question	Assessment [1..5]
Experience with handhelds	1,29
Motivation by use of handhelds	4,29
Mark for the game	4,29
Extend game to whole exhibition	4,2

Table 6: Average values of interviews rating from 1 to 5(=best)

Likewise, AR animations were experienced as exciting and interesting in *Expedition Schatzsuche,* leading to statements such as "*the exhibit with the flat iron was cool.*" In particular the users rated spatial interaction as highly motivating: "*Not only answering questions, but also playing … like the piano and the organ*" and "*to learn and to play*" [at the same time]; "*…although one is learning*" and "*I haven't known that the music instruments are so old.*" [referring to the oldest conserved Mandora in the world]. The concept of story arcs was also found very motivating, "*the combination thing was very good*" and [I liked] "*…that you have to move around and search…like a detective.*"

Solving tasks by taking pictures of specific items also was perceived as intuitive and easy to handle. When the players found the right answers, audio explanations were given, which was found to be highly satisfactory.

8.4. Collaboration

One aspect that worked unexpectedly well in *medien.welten* was the cooperative use of a single handheld device by a group of 2-3 students. The players collaborated efficiently rather than competing for the device as we had feared. While the device sharing in *medien.welten* was a result of a short supply of devices, in *Expedition Schatzsuche* we were able to provide a personal device for every student. Even with this increased device availability, users were collaborating voluntarily. They spontaneously formed groups playing together, then split up again and played individually. In one case, two users shared a device, after another device had crashed due to hardware problems.

One problem we found was that when players formed groups they often operated multiple devices simultaneously, which resulted in playing multiple audio clips at the same time. Especially voices were hard to understand in such a case. This problem is specific to audio rendering and could possibly be addressed using earphones.

9 Conclusions and Future Work

In this paper we presented a complete handheld AR framework and a multi-player game engine for treasure hunt games built on top of it. The framework runs efficiently on low cost, off-the-shelf devices with Windows CE. The presented system is currently used in a real life project with a museum and a commercial partner, who is planning to reuse the developed technology for other museum games. Other applications for public venues such as large conferences are currently under commercial development.

While the current framework is suitable for several real applications, marker tracking is not suitable for other potential areas where AR could be deployed. A next major step will therefore be to develop natural feature tracking for handheld devices. The low processing capabilities of mobile devices make this a challenging endeavor. However, programmable GPU features that are expected for the next generation of smartphones could provide the additional computational resources that are necessary for real-time natural feature tracking.

Another topic for future research are massive multi-user AR applications. Massive multi-user online games for PCs are extremely successful, and multi-user AR games could also be an exciting opportunity. Yet a lot of research on innovative user interface concepts for such scenarios will be required.

Acknowledgements

This project was funded in by part by Austrian Science Fund FWF under contract No. L32-N04 and Y193. Further we want to thank Mag. Erich Wappis, Mag. Nina Mayer and Mag. Ines Kuttnig of Landesmuseum Kärnten for designing the content of "Expedition Schatzsuche" and organizational assistance.

References

[1] Amselem, D., A window on shared virtual environments. Presence, Vol. 4, No. 2, pp. 130-145, 1995

[2] Aqua Gauntlet, Mixed Reality Lab Japan, ACM SIGGRAPH 2000 Emerging Technologies demo, http://www.mr-system.co.jp/project/aquagauntlet

[3] Benford S., A. Crabtree, M. Flintham, A. Drozd, R. Anastasi, M. Paxton, N. Tandavanitj, M. Adams, J. Row-Farr: Can you see me now? ACM Trans. Comput.-Hum. Interact. 13(1):100-133, 2006.

[4] Billinghurst, M., Poupyrev, I., Kato, H., May, R., Mixing Realities in Shared Space: An Augmented Reality Interface for Collaborative Computing, Proceedings of ICME 2000, pp. 1641-1644, NY, USA, 2000

[5] Bimber O., B. Fröhlich, D. Schmalstieg, L. M. Encarnação: The Virtual Showcase. IEEE Computer Graphics and Applications 21(6):48-55, 2001.

[6] Brown, B., MacColl, I., Chalmers, M., Galani, A., Randell, C., Steed, A., Lessons from the lighthouse: collaboration in a shared mixed reality system, Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 577-584, Florida, USA, 2003

[7] Fiala M.: ARTag, a Fiducial Marker System Using Digital Techniques, In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), 590-596, 2005.

[8] Fitzmaurice, G. W. Situated Information Spaces and Spatially Aware Palmtop Computers. Communications of the ACM, Vol.36, Nr.7, pp 38-49, 1993

[9] Föckler, P., Zeidler, T., Brombach, B., Bruns, E., Bimber, O., PhoneGuide: Museum Guidance Supported by On-Device Object Recognition on Mobile Phones, Proceedings of International Conference on Mobile and Ubiquitous Computing (MUM'05), pp. 3-10, 2005

[10] Gausemeier, J., Fruend, J., Matysczok, C., Bruederlin, B., Beier, D., Development of a real time image based object recognition method for mobile AR-devices, Proceedings of 2nd international conference on Computer graphics, Virtual Reality, visualisation and interaction in Africa, pp. 133-139, 2003

[11] Grimm P., Haller M., Paelke V., Reinhold S., Reimann C., Zauner J., AMIRE - Authoring Mixed Reality, First IEEE International Augmented Reality Toolkit Workshop, Darmstadt, Germany, 2002

[12] Henrysson, A., Billinghurst, M., Ollila, M. Face to Face Collaborative AR on Mobile Phones. International Symposium on Augmented and Mixed Reality (ISMAR'05), pp. 80-89, 2005

[13] Ingram, D., Newman, J., Augmented Reality in a Wide Area Sentient Environment, Proceedings of the 2nd IEEE and ACM International Symposium on Augmented Reality (ISAR 2001), p. 77, New York, USA, 2001

[14] Kato H., M. Billinghurst: Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System. Proc. 2nd IEE International Workshop on Augmented Reality (IWAR 99), 1999.

[15] Klopfer, E., Perry, J., Squire, K., Jan, M., Steinkuehler, C., Mystery at the Museum - A Collaborative Game for Museum Education, CSCL (Computer Supported Cooperative Learning) pp. 316-320, 2005, Taipei, Japan, 2005

[16] Ledermann, F., Schmalstieg, D., APRIL: a high-level framework for creating augmented reality presentations, Proceedings of Virtual Reality, 2005 (VR2005), pp. 187-194, 187-194, Germany, 2005

[17] Lindt, I., Ohlenburg, J., Pankoke-Babatz, u., Prinz W., Combining Multiple Gaming Interfaces in Epidemic Menace, Experience Report at International Conference for Human-Computer Interaction (CHI'2006), 213-218, Montréal, Canada, 2006

[18] Long, S., Aust, D., Abowd, G. D., Atkeson, C., Cyberguide: Prototyping Context-Aware Mobile Applications. Proceedings of the CHI '96, pp. 293-294, NY, USA, 1996

[19] MacIntyre, B., Bolter, J.D., Moreno, E., Hannigan, B., Augmented Reality as a New Media Experience, International Symposium on Augmented Reality (ISAR 2001), p. 197, New York, NY, 2001

[20] MacIntyre, B., Gandy, M., Prototyping applications with DART, the designer's augmented reality toolkit. Proceedings of STARS 2003, pp. 19-22, Tokyo, Japan, 2003

[21] Makri, A., Arsenijevic, D., Weidenhausen, J., Eschler, P., Stricker, D., Machui, O., Fernandes, C., Maria, S., Voss, G., Ioannidis N., ULTRA: An Augmented Reality System for Handheld Platforms, Targeting Industrial Maintenance Applications, Proceedings of 11 th International Conference on Virtual Systems and Multimedia (VSMM'05), Ghent, Belgium, 2005

[22] Moehring, M., Lessig, C. and Bimber, O., Video See-Through AR on Consumer Cell Phones, International Symposium on Augmented and Mixed Reality (ISMAR'04), pp. 252-253, 2004

[23] Mogilev, D., Kiyokawa, K., Billinghurst, M., Pair, J., AR Pad: an interface for face-to-face AR collaboration, CHI '02 extended abstracts on Human factors in computer systems, pp. 654-655, 2002

[24] Regenbrecht, H.T., Specht, R., A Mobile Passive Augmented Reality Device – mPARD. Proceedings o ISAR, pp. 81-84, Munich, Germany, 2000

[25] Rekimoto, J., Nagao, K. The World through the Computer: Computer Augmented Interaction with Real World Environments, User Interface Software and Technology (UIST '95), pp. 29-38, 1995

[26] Rekimoto, J., TransVision: A Hand-held Augmented Reality System for Collaborative Design, Proceedings of Virtual Systems and Multi-Media (VSMM '96), pp. 18-20, Gifu, Japan, 1996

[27] Schweighofer G., A. Pinz: Robust pose estimation from a planar target. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(12):2024–2030, 2006.

[28] Shibata, F., Mobile Computing Laboratory, Department of Computer Science, Ritsumeikan University, Japan, http://www.mclab.ics.ritsumei.ac.jp/research.html

[29] Stapleton, C.B., Hughes, C.E., Moshell, J.M., MIXED FANTASY: Exhibition of Entertainment Research for Mixed Reality, CM International Symposium on Mixed and Augmented Reality (ISMAR 2003), pp. 354-355, Tokyo, Japan, 2003

[30] Wagner, D., Pintaric, T., Ledermann, F., Schmalstieg, D., Towards Massively Multi-User Augmented Reality on Handheld Devices, Proceedings of the 3rd International Conference on Pervasive Computing (PERVASIVE 2005), pp. 208-219 Munich, Germany, 2005

[31] Wagner, D., Schmalstieg, D., First steps towards handheld augmented reality, Proceedings of the 7th International Symposium on Wearable Computers (ISWC'2003), pp. 127-137, NY, USA, 2003

[32] Wagner, D., Schmalstieg D., ARToolKitPlus for Pose Tracking on Mobile Devices, Proceedings of 12th Computer Vision Winter Workshop (CVWW'07), 2007

[33] Wagner, D., Schmalstieg D., Muddleware for Prototyping Mixed Reality Multiuser Games, Proceedings of IEEE Virtual Reality 2007 (VR2007), pp. 235-238, Charlotte, NC, USA, 2007

[34] Wu, M., Mitchell, K., McCaffery, D., Finney, J., Friday, A., Real Tournament – mobile context-aware gaming for the next generation, The Electronic Library, Volume 22, Number 1, pp. 55-64, 2004