Flexible development of location-based mobile augmented reality applications with AREA

Implementation of a serious game shows the flexibility of AREA

Marc Schickler · Manfred Reichert · Philip Geiger · Jens Winkler · Thomas Funk · Micha Weilbach · Rüdiger Pryss

Received: 30 December 2019 / Accepted: 15 April 2020 © The Author(s) 2020

Abstract
Mobile applications have garnered a lot of attention in the last years. The computational capabilities of mobile devices are the mainstay to develop completely new application types. The provision of augmented reality experiences on mobile devices paves one alley in this field. For example, in the automotive domain, augmented reality applications are used to experience, inter alia, the interior of a car by moving a mobile device around. The device’s camera then detects interior parts and shows additional information to the customer within the camera view. Another application type that is increasingly utilized is related to the combination of serious games with mobile augmented reality functions. Although the latter combination is promising for many scenarios, technically, it is a complex endeavor. In the AREA (Augmented Reality Engine Application) project, a kernel was implemented that enables location-based mobile augmented reality applications. Importantly, this kernel provides a flexible architecture that fosters the development of individual location-based mobile augmented reality applications. The work at hand shows the flexibility of AREA based on a developed serious game. Furthermore, the algorithm framework and major features of it are presented. As the conclusion of this paper, it is shown that mobile augmented reality applications require high development efforts. Therefore, flexible frameworks like AREA are crucial to develop respective applications in a reasonable time.

Keywords Mobile augmented reality · Location-based algorithms · Mobile application engineering · Serious game · Mobile augmented reality game

1 Introduction
In everyday life, mobile augmented reality is increasingly utilized (Sánchez-Acevedo et al. 2018). However, the development of respective mobile applications is still complex. To mention only one example, existing mobile operating systems pose challenging differences and are frequently updated. Although the capabilities of modern mobile devices enable powerful mobile applications, the aforementioned aspects make the development of mobile augmented...
realities applications costly and time-consuming (Korinth et al. 2020). Hence, flexible frameworks are required. With the releases of ARKit Apple (2019) and ARCore Google (2020) by Apple and Google, it seems that also large vendors address the demands of mobile application developers by providing flexible frameworks. Interestingly, for other mobile application types, the trend to provide flexible frameworks is also increasing. For example, Apple has recently released a framework called HealthKit Apple (2020b) to develop mobile health applications more flexibly.

In the AREA (Augmented Reality Engine Application) project, a kernel was implemented that enables the robust implementation of location-based mobile augmented reality applications. In the light of the discussed demands, the kernel focuses on the following pillars:

- It shall run on Android and iOS in the same way. More specifically, the development shall be possible in the same way on both mobile operating systems, with the same functionality.
- It shall abstract from the peculiarities of the mobile operating systems in the best possible way. For example, the same usage of the sensor possibilities must be ensured for developers.
- It shall enable developers to easily integrate new features.
- It shall enable developers to easily implement their own application types on top of AREA.

For interested readers, in-depth information to the modular design principles of AREA can be found in Pryss et al. (2016, 2017a). In addition, the works (Schickler et al. 2015; Geiger et al. 2014, 2013; Pryss et al. 2017b) discuss in what way AREA deals with the peculiarities of the different mobile operating systems.

Importantly, this work extends (Schickler 2019). In the latter work, the algorithm framework of AREA was presented in more detail. This manuscript extends (Schickler 2019) by the following aspects:

1. The discussion of related works has been extended.
2. The discussion of algorithms developed in AREA has been extended. In particular, major settings of the algorithms as well as more information to the track optimization algorithm have been added.
3. A discussion of aspects when comparing AREA with ARKit has been added.
4. Most importantly, the development of a serious game based on top of AREA has been added.

The remainder of the work at hand is organized as follows: In Sect. 2, related works are discussed. Sect. 3 discusses the developed AREA algorithms, while Sect. 4 presents aspects of comparing AREA with ARKit from Apple. In Sect. 5, a feature is presented that enables the optimization of tracks and areas using AREA. In Sect. 6, the development of a serious game on top of AREA is presented and discussed. In Sect. 7, the practical use of AREA and its features based on the presented results are summarized. Finally, Sect. 8 concludes the paper with a summary and an outlook.

2 Related work

Previous research that is related to the development of location-based augmented reality applications in non-mobile environments can be found in Kooper and McIntyre (2003). In Kähäri and Murphy (2006), mobile devices are used to develop an augmented reality system. The augmented reality application described in Lee et al. (2009) enables users to interact with media data through augmented reality. However, none of these approaches share insights pertaining to the development of location-based augmented reality on mobile devices as AREA does. Regarding tracks in mobile augmented reality, only little work can be found in literature. For example, the approaches (Vlahakis et al. 2002; Lee and Hollerer 2008; Hollerer 2004) present tracks as a key feature of (mobile) augmented reality applications. Interestingly, the algorithms to implement the tracks are not presented. In addition, no performance issues related to the developed track algorithms are discussed. Moreover, few contributions exist, which deal with the engineering of mobile augmented reality systems in general. For example, (Grubert et al. 2011) validates existing augmented reality browsers in the light of engineering aspects. As two recent examples, in Ren et al. (2019); Liu et al. (2019), edge-driven aspects for mobile augmented reality are presented. These works also show that engineering aspects are crucial for the development of mobile augmented reality applications. Kim et al. (2016), in turn, discusses various types of location-based augmented reality scenarios. More precisely, issues that have to be particularly considered for a specific scenario are discussed in more detail. Again, engineering aspects of mobile applications are not considered. In Yang et al. (2016), an authoring tool for mobile augmented reality applications, which is based on marker detection, is proposed, while (Paucher and Turk 2010) presents an approach for indoor location-based mobile augmented reality applications. Reitmayr and Schmalstieg (2003) gives an overview of various aspects of mobile augmented reality for indoor scenarios. Moreover, another scenario for mobile augmented reality is presented in Lee and Rhee (2015). The authors use mobile augmented reality for image retrieval. To summarize, (Yang et al. 2016; Paucher and Turk 2010; Reitmayr and Schmalstieg 2003;
Lee and Rhee (2015) do not address engineering aspects of location-based mobile applications. In Chung et al. (2016), an approach supporting pedestrians with location-based mobile augmented reality is presented, while (Capece et al. 2016) deals with a client and server framework enabling location-based applications. Presently, new scenarios emerge, in which mobile augmented reality is investigated. Recent related works can be found that provide general overviews (Sánchez-Acevedo et al. 2018). Other related works deal with particular scenarios. For example, in Liou et al. (2018); Leighton and Crompton (2017); Tosun (2017), many examples related to education are discussed and evaluated. Further scenarios constitute tourism (Jung et al. 2018; Koh 2019) and crime management (Liao et al. 2018). A field, which is currently heavily pursued, constitutes mobile augmented reality in gaming scenarios (Rauschnabel et al. 2017; Litts and Lewis 2019; Laine and Suk 2019; Jang and Liu 2019). In medicine, mobile augmented reality becomes increasingly important as well (Ghandorh et al. 2017; Mladenovic 2019). Considerations on psychological factors when using applications in the reality–virtuality continuum are also being addressed more and more (Ibili and Billinghurst 2019; Hoppenstedt et al. 2019). In addition to this, the behavior of mobile users should be generally considered in future for any mobile application (Beierle 2019; Majeed 2019). Finally, in patents, engineering aspects are also covered (Hill et al. 2019). In conclusion, the engineering of mobile augmented reality applications is still a topical subject, for which AREA tries to be a contribution for the interested community.

3 Algorithm framework

Prior to the algorithm details, a brief introduction into the concept of AREA (Geiger et al. 2013; Pryss et al. 2017a) is given. Most importantly, AREA relates users, while holding their mobile device, to the objects detected in the camera view (i.e., Points of Interest (POIs), tracks, areas, 3D objects). In general, AREA is based on five pillars.

1. A virtual 3D world is used to relate the user’s position to the position of the objects.
2. The user is located at the origin of this world.
3. Instead of the physical camera, a virtual 3D camera is used that operates with the created virtual 3D world. The virtual camera is therefore placed at the origin of this world.
4. The different sensor characteristics of the supported mobile operating systems are covered to enable the virtual 3D world.
5. The physical camera of the mobile device is adjusted to the virtual 3D camera, based on the assessment of sensor data.

To enable these five technical pillars, the algorithms presented in Fig. 1 have been developed. In a first version of
A new POI algorithm became necessary (see Fig. 1, POI Algorithm v2). All presentations in this work are based on POI Algorithm v2. Note that the technical aspects of POI Algorithm v2 can be found in Pryss et al. (2016, 2017a). Furthermore, in Fig. 1, all references are shown, in which the respective information of the algorithms (i.e., their listings and backgrounds) can be found. The following list summarizes the important aspects as well as new features shown in this work:

- The calculations to position POIs correctly are based on a multitude of other calculations (i.e., mainly the sensor fusion) and design decisions (i.e., mainly whether or not using external libraries). In this context, the correct positioning of POIs constitutes a major challenge. On the other hand, the provided performance is also very important. When considering preciseness and performance on different mobile operating systems with the goal to have comparable mobile applications, the overall technical endeavor is even more challenging. For example, on Android, for the sensor fusion, an additional 3D rotation matrix algorithm (Pryss et al. 2017a) became necessary to enable the same user experience as on iOS (see Fig. 1, only on Android).

- On top of the POI algorithms, two additional algorithms were implemented. The first algorithm is able to handle clusters. Clusters are overlapping POIs, which are difficult to interact with. How clusters are handled is presented in Pryss et al. (2017a). The second algorithm is able to calculate tracks and areas. The demand for this feature emerged while using AREA in practice. For the development of the algorithm that is able to handle tracks and areas, note that OpenGL libraries were used in addition. The decision was made with respect to the differences of the two supported mobile operating systems Android and iOS. In this context, it was a goal to combine the (1) best of the already proven AREA developments and the (2) existing features of OpenGL. More specifically, the proven sensor fusion and POI algorithms of AREA should be further used as they revealed comparable results on both mobile operating systems. In addition, as the OpenGL libraries are decoupled from the sensor fusion with respect to general functions required for track and area handling, it was efficient to additionally use features of OpenGL for AREA. Therefore, OpenGL libraries were utilized for track and area handling. In-depth information to this algorithm can be found in Pryss et al. (2017b).

- In practice, even when considering the existing powerful mobile device capabilities, the proper handling of tracks and areas is challenging with respect to the resource perspective. When dealing with a huge number of tracks or areas, or a combination of both, present mobile devices reveal their limitations. Therefore, a new algorithm was implemented, which deals with performance issues while displaying many tracks and areas at the same time. How this algorithm operates will be presented in Sect. 5.

- Except for the new algorithm that deals with the performance while displaying many tracks and areas, all other algorithms were evaluated in experiments (i.e., compared to other mobile augmented reality applications). As can be obtained from the works (Pryss et al. 2017a, b), AREA competes well with other mobile applications providing the same or similar features. However, in future experiments, performance will be further evaluated. Moreover, the newly developed algorithm for track and area handling will be evaluated in a separate experiment.

- Since Apple and Google recently released ARKit Apple (2019) and ARCore Google (2020), future developments will consider these libraries as well.

- Based on the developed algorithms shown in this work, the development of a serious game is discussed in Sect. 6, which is denoted with ARGame.

As an additional information of AREA, Listing 1 presents major settings used in the algorithms of the iOS version of AREA. Note that the same settings are managed on Android. Radius, minRadius, maxRadius, maxPOIVisible, cameraFieldOfView, compassBetterOnlyPortrait, radiusPicker, areasModal, and sharedInstance are used by the algorithm that positions POIs correctly. useGoogle, googleAPIKey are used for testing purposes. If POIs cannot (or not being wanted to) be loaded from a remote database, this feature can be used to gather data from Google. Finally, poiClustering, horizontalClusterWidth, and verticalClusterHeight are used to handle POI clusters.
since Apple recently released ARkit Apple (2019), this section shall enable researchers to directly compare the features and function of the iOS version of AREA with ARkit. In ARkit, also many other features like face tracking are provided. Consequently, ARkit aims at a broader perspective on mobile augmented reality applications. However, from the technical perspective, it might be of interest to directly compare the functions of the iOS version of AREA with ARKit. In ARkit, the chain of classes ARSession > ARFrame > ARCamera must be used to enable a location-based mobile augmented reality experience. To be more precise, the class ARSession must be used to handle the sensor fusion, while the classes ARFrame and ARCamera must be used to handle the positioning of POIs. Regarding the ARSession class, compared to AREA, a developer must manually add GPS data to the sensor fusion. With ARkit, compared to AREA, a developer is relieved from directly reading data from the device’s motion sensing hardware. Another interesting comparison is related to the ARCamera class of ARKit. By using the ARCamera class, the correct positioning of POIs can be accomplished. Therefore, the relevant components of the ARCamera class (Apple 2020a) can be compared to the iOS version of AREA as shown in Table 1.

In order to compare the functions directly, the corresponding AREA functions are presented in the following. First, Listing 2 presents the listing that can be compared to the func transform of ARKit.

### Listing 1 AREA Base Settings (iOS version)

```swift
import Foundation

// This class can be used to adjust the behavior of Area.
class AREASettings {
    // The current radius set by the user and stored on the device's storage
    var radius : Double
    // The minimum radius
    var minRadius : Double
    // The maximum radius
    var maxRadius : Double

    // Maximum number of points of interest visible and drawn on the screen
    var maxPOIsVisible : Int

    // The camera's field of view
    var cameraFieldOfView : Double

    // Sets whether overlapping points of interest should be clustered to a new one
    var useClustering : Bool

    // Sets whether Google Places API should be used to download points of interest
    var useGoogle : Bool

    // Sets whether the more precise compass (i.e., only the portrait compass) or
    // the slightly less precise compass (i.e., the portrait and landscape compass)
    // should be used for compassBetterOnlyPortrait : Bool

    // The radiusPicker used to store and retrieve the radius set by the user
    fileprivate var radiusPicker : AREARadiusPicker

    // Sets whether AREA is modal or not. If so, a button for dismissing AREA is
drawn on the screen
    var areaIsModal : Bool

    // The minimal horizontal distance between to points of interest before they
    // get clustered
    var horizontalClusterWidth : Double

    // The minimal vertical distance between to points of interest before they
    // get clustered
    var verticalClusterHeight : Double

    // The Google API Key used for downloading points of interest and images
    var googleApiKey : String

    // The shared instance of AREASettings
    static let sharedInstance = AREASettings()

    // Create the standard settings of Area
    init() {
        radius = 1000
        minRadius = 500.0
        maxRadius = 8000.0
        maxPOIsVisible = 100
        cameraFieldOfView = 60.0
        useClustering = true
        useGoogle = true
        compassBetterOnlyPortrait = true
        radiusPicker = AREARadiusPicker()
        areaIsModal = true
        horizontalClusterWidth = 10.0
        verticalClusterHeight = 8.0
        // insert your own Google API Key for Static Maps and Locations
        googleApiKey = ""
        radius = initializedRadius()
    }
}
```

### Table 1 Comparing ARKit and AREA Functions

| ARKit          | AREA          |
|----------------|---------------|
| func transform | func normalizeRotationMatrix |
| func projectionMatrix | func redrawAreaView |
| func projectPoint | func positioningPOI |

4 ARKit and AREA

Since Apple recently released ARkit Apple (2019), this section shall enable researchers to directly compare the features and function of the iOS version of AREA with ARkit. In general, ARKit was developed by Apple with the goal to provide a multitude of mobile augmented reality applications. Currently, all features developed in AREA pursue the goal to enable location-based mobile augmented reality applications. To be more precise, the location is based on GPS coordinates and therefore AREA mainly aims at outdoor location-based augmented reality applications. In ARkit, also many other features like face tracking are provided. Consequently, ARkit aims at a broader perspective on mobile augmented reality applications. However, from the technical perspective, it might be of interest to directly compare the functions of the iOS version of AREA with ARKit. In ARkit, the chain of classes ARSession > ARFrame > ARCamera must be used to enable a location-based mobile augmented reality experience. To be more precise, the class ARSession must be used to handle the sensor fusion, while the classes ARFrame and ARCamera must be used to handle the positioning of POIs. Regarding the ARSession class, compared to AREA, a developer must manually add GPS data to the sensor fusion. With ARkit, compared to AREA, a developer is relieved from directly reading data from the device’s motion sensing hardware. Another interesting comparison is related to the ARCamera class of ARKit. By using the ARCamera class, the correct positioning of POIs can be accomplished. Therefore, the relevant components of the ARCamera class (Apple 2020a) can be compared to the iOS version of AREA as shown in Table 1.

In order to compare the functions directly, the corresponding AREA functions are presented in the following. First, Listing 2 presents the listing that can be compared to the func transform of ARKit.
Second, Listing 3 presents the listing that can be compared to the `func projectionMatrix` of ARKit.

Listing 3 AREA func redrawAreaView

```swift
Listing 2 AREA func normalizedRotationMatrix
```
Currently, performance experiments are conducted to compare ARKit with AREA. In general, the provision of ARKit emphasizes that mobile augmented reality has become an important mobile application type. In line with ARKit, the application of AREA in practice revealed that features enabling mobile augmented reality applications beyond location-based outdoor scenarios are promising. Therefore, the authors of the work at hand work on new features like, for example, the recognition of objects in AREA. Furthermore, AREA is currently compared with ARCore Google (2020) from Google.

5 Optimization of track and area algorithm

In the real-life projects1 for which AREA is utilized, the display of many tracks and areas with the algorithm shown in Pryss et al. (2017b) revealed performance issues. Therefore, a feature on top of this algorithm was implemented in order to cope with demanding scenarios. It is only shown how the optimization is implemented in Android and for tracks; however, areas and the implementation on iOS are performed using the same principles. In general, a track is displayed by the use of bars. Between each bar, a distance of 1 m ($m = \text{meter}$) is used. Having this in mind, a track of 1km (km=kilometer) requires roughly 501 bars. Each bar, in turn, is represented by two triangles. Each vertex of a triangle has 3 coordinates (x, y, and z), and a RGBA value (i.e., r, g, b, and a components). The three coordinates as well as the 4 RGBA components require 4 bytes. Having these values also in mind, reconsider the track of 1km. This track would require $501 \cdot 2 \cdot 3 \cdot (3 + 4) \cdot 4 = 84168$ bytes to store it. If many tracks or areas shall be displayed, this affects the performance based on the required data to be stored and calculated. To increase the performance in the case that many tracks have to be displayed (or/and areas), the general idea is to manage a detail level for tracks (and areas). Based on this detail level, tracks and areas can be displayed in different resolutions. The notion of the resolution and how it is calculated are shown in the following.

As the first step, the required preliminary calculations are presented. As a first step, consider a list checkpoints containing n vectors (x, y, and z). Each vector n in checkpoints represents one point on a track that shall be displayed. Furthermore, the checkpoints list stores the values in an ordered manner according to a track. In addition, three further lists are managed: degreesY, degreesXZ, and pairs. Each of these lists stores n − 2 values. More specifically, the values of the three lists store the following:

- degreesY: stores the angle of a track point B that lies between points A and C. More precisely, it stores the difference in height between A and C, based on point B.
- degreesXZ: stores the angle of a track point B that lies between points A and C. More precisely, it stores the cardinal points between A and C, based on point B.
- pairs: stores the indexes of points A, B, and C.

To determine degreesY and degreesXZ, the following calculations are applied:

- degreesY: To calculate the angle for a point B, the two points $B'$ and $B''$ are calculated. $B'$ contains $(xb, ya, zb)$ and $B''$ contains $(xb, yc, zb)$. Following this, $B'$ holds the y-value of the point A and $B''$ the y-value of the point C. Based on this, the two rectangular triangles $A - B' - B$ and $B - B'' - C$ can be created. Finally, the sum of triangles between $A - B' - B$ and $B - B'' - C$ results in the single entry degreesY.
- degreesXZ: To calculate all values for degreesXZ, the vectors $AB$ and $AC$ are calculated.

Based on the shown lists, the size of a track can be decreased with respect to so-called detail levels. A detail level reduces tracks (and areas) to x track points. Reduction means that the originally defined amount of track points n is reduced to x. The reduction, in turn, is calculated as follows:

- The lists degreesY, degreesXZ, and pairs are calculated for a track that shall be minimized.
- Then, within a loop, all values in degreesY and degreesXZ are evaluated whether they will be in the list for x. If x points have been identified, the loop will be finished. The next steps show how the evaluation is performed.
- First, a variable steps is initialized with 1 (meaning 1 degree). steps is a threshold that must be exceeded when calculating $|180 - \text{degreesY}[i]| + |180 - \text{degreesXZ}[i]|$ for each entry in the lists degreesY and degreesXZ. If the calculation exceeds the value of steps, the entry will be used for x.
- Visually speaking: The more the triangle between two points approaches 180 degrees, the more it visually approaches a straight line. Consequently, the elimination of such a point can be visually accepted.
- If no value can be found within a loop run that can be eliminated, then steps is increased to 2 (and so forth).
- If a value can be found at index i of the lists degreesY and degreesXZ, then they are recalculated as follows: degreesY[i − 1], degreesY[i + 1], degreesXZ[i − 1], degreesXZ[i + 1], pairs[i − 1], and pairs[i + 1] are newly calculated and the entries for the index i are removed.
- If the initial lists degreesY, degreesXZ, and pairs are decreased to x points, the algorithm is finished.

1 see http://www.liveguard.de for all mobile applications that use AREA
The list of $x$ points is then displayed using the algorithm shown in Pryss et al. (2017b). All other relevant calculations for a further understanding can be found in Geiger et al. (2013); Pryss et al. (2017a).

The implementation of the algorithm to reduce the number of track points is shown in Listing 5. Note that the listing only shows the Android version (the iOS version works accordingly).

**Listing 5 Track Optimization Algorithm (Android version)**

```java
int[] complexityLevels = {50, 40, 30, 20, 10, 5};
int[] distanceLevels = {500, 300, 200, 100, 50};
int[] complexityLevels = new int[7];
abstract (class tol
    void (class tol
        int complexityLevels[] = new int[totalPath.size()));
    for (int i = 0; i < totalPath.size(); i++)
        complexityLevels[i] = i;
}

float[] orientationDegree = new float[totalPath.size()-1];
float[] horizontalDegree = new float[totalPath.size()-1];
int[][] pairs = new int[totalPath.size()-1][1];
    // fill orientationDegree, horizontalDegree, pairs
    // orientationDegree[i] and horizontalDegree stores the degree
    // between points i+1, i and i
    // pairs[i][0] stores point j+1 and i

int actualCount = orientationDegree.length;
int lastAbstraction = 0;
float actualAbstraction = 1;
    for (int i = 0; i < orientationDegree.length; i++)
        if (actualCount <= complexityLevels[1] & complexityLevels[1] == null)
            // fill complexityLevels[1]
    }
    int check = min(abs[100 - orientationDegree[i]], abs[100 - horizontalDegree[i]]);
    if (check < actualAbstraction - orientationDegree[i] || abs[100 + horizontalDegree[i]]
        if (check < actualAbstraction - horizontalDegree[i] || abs[100 + horizontalDegree[i]]
            // calculate pairs[i][1], pairs[i][1] and [i]
            // calculate horizontalDegree[i] and [i]
            lastAbstraction = 1;
            actualCount = -1;
            if (i == orientationDegree.length - 1) i = 0;

// complexityLevels[] - array is managed as follows:
// complexityLevels[0] contains the index to all points i.e., -> 0, 1, 2, ..., totalPath.size()
// complexityLevels[1] contains indices from maximally 50 points,
// complexityLevels[2] contains indices from maximally 40 points,
// complexityLevels[3] contains indices from maximally 30 points,
// ...
// complexityLevels[n] contains indices from maximally 5 points.

// Which complexityLevels then is actually used depends on the distance from the user to
// the nearest point of the bounding box of the track
// the bounding box must be calculated beforehand while creating the list totalPath
// the array distanceToGeo stores the distance to decide which complexityLevels is
// actually used.
```

| Level | Reduction schema |
|-------|------------------|
| 0     | No reduction     |
| 1     | Tracks within 50 m of a user’s position (i.e., using 50 track points) |
| 2     | Tracks within 100 m of a user’s position (i.e., using 40 track points) |
| 3     | Tracks within 200 m of a user’s position (i.e., using 30 track points) |
| 4     | Tracks within 300 m of a user’s position (i.e., using 20 track points) |
| 5     | Tracks within 400 m of a user’s position (i.e., using 10 track points) |
| 6     | Tracks beyond 400 m of a user’s position (i.e., using 5 track points) |

Which level is actually used is determined during run time based on the position changes of a user. In Figs. 2 and 3, the algorithm is shown in practice (the screenshots are from one of the real-life applications that can be found in CMCityMedia (2020)). Notably, trails that are displayed by the use of bars constitute detail Level 0. The other displayed trails constitute detail Level 4. In practice, the feature was highly welcome as the performance could be increased. Currently, experiments are conducted (as shown in Pryss et al. (2017a)) to obtain quantitative results on the actual performance increase.

### 6 ARGame development

As already discussed in Sect. 2, the combination of mobile augmented reality and serious games has recently garnered a lot of attention (Rauschnabel et al. 2017; Litts and Lewis 2019; Laine and Suk 2019; Jang and Liu 2019). In the real-life projects of ARE/A (CMCityMedia 2020), this was also a frequent demand. Therefore, the development of an implemented serious game on top of ARE/A is presented. The game is denoted with ARGame. In detail, the technical aspects of the ARGame, the required extensions of ARE/A, and impressions of the ARGame are presented. With ARGame, it can be shown that the flexible design of ARE/A enables new applications on top of it in a reasonable development time.

Being a decisive aspect, the operating principle of ARGame is delineated in the following. In this context, note that ARE/A is commercially utilized for city apps (CMCityMedia 2020). In these apps, users have the opportunity to experience many aspects of a city. With the presented ARGame, the experience shall be enhanced. To be more precise, a citizen shall be enabled when using ARGame to find avatars. These avatars are geo-tagged, and if a user is within a radius of 10 m to an avatar, then the

#### 5.1 Track optimization in practice

ARE/A manages seven detail levels. To be more specific, the tracks (and areas) are displayed using these levels, depending on the distance a user has to them. The detail levels are distinguished by the number of track points to which the track is reduced to by Algorithm 5. The levels, in turn, are managed as shown in Table 2.
Flexible development of location-based mobile augmented reality applications with AREA

The latter is displayed in the camera view of a user. Figs. 4 and 5 illustrate how these avatars appear on Android and iOS. If the user clicks on the avatar, then it disappears and the user gets credits for having found the avatar. Credits, in turn, can be used to be redeemed in the participating stores of the respective city for which the app was developed. The overall idea is that the attention of users is attracted to important points in the city. By using this principle, users learn more about the city, on the other, they are awarded. Fig. 6 illustrates received points of a user. One more feature is provided by the ARGame, which is shown in Fig. 7, it offers contextual information. The latter shall support users in playing ARGame.

Next, it is discussed how ARGame was technically integrated into the existing AREA core. Note that the technical integration is discussed based on the Android implementation as the differences to iOS are only very slightly.

If AREA is started by a user, then the activity denoted with Area20MainActivity is called. The respective onCreate method is shown in Listing 6.

Listing 6  AREA onCreate Method (Android)

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    Toast.makeText(this, "Load Data...", Toast.LENGTH_SHORT).show();

    MapAndConfig config = getIntent().getExtras().getParcelable("config");
    if (config == null) {
        throw new RuntimeException("AREAMainActivity must be given a proper MapAndConfig object");
    }

    this.config = config;
    callARFragment(getFragment(config));
    prefs.INSTANCE.openDatabase();
}
```

In the context of the onCreate method, the getFragment method is important, which is shown in Listing 7.
Fig. 4 Avatar on android

Fig. 5 Avatar on iOS

Fig. 6 Received credits on iOS

Fig. 7 Contextual information on android

Listing 7 AREA getFragment Method (Android)

```java
private AreaMainFragment getFragment(StringAbdArConfig config) {
    if (isGame(config)) {
        return AreaGameFragment.newInstance(config);
    } else {
        if (isMarketing(config)) {
            return AreaMainMarketingFragment.newInstance(config);
        } else {
            return AreaMainFragment.newInstance(config);
        }
    }
}
```
Based on the content of the config object, AREA decides which augmented reality function shall be utilized. Notably, Area20MainFragment comprises all algorithms, which have been summarized in Fig. 1. In this case, AREA shows Points of Interest, Tracks, and Areas. As the second option, Area20GameFragment means to start the ARGame. A third option, which is currently implemented, is called Area20MarketingFragment. This feature shall provide a chatbot, which appears in front of geo-tagged shops to help customers before actually entering the shop.

In the Area20MainFragment, the classic AREA functions are summarized. As already described, these are the features to display POIs, tracks, areas, and the radar. Notably, the Area20GameFragment inherits from Area20MainFragment, and mainly uses the transformation of coordinates and the radar feature. Due to the reason that in Area20MainFragment solely 2D objects are used, but 3D objects are additionally needed in Area20GameFragment, the rendering feature had to be changed for Area20GameFragment. This includes adding new functions that are shown in Figs. 4, 5, 6, and 7. Finally, the required calculations of the positions of the avatars had to be changed for Area20GameFragment.

Technically, the changes are accomplished as follows. After initializing the Area20MainFragment, the method denoted with initViews is called, which is shown in Listing 8. In this method, the radar and the button to terminate AREA are initialized. These functions are also used by the ARGame. In this method, it is also evaluated whether or not the app is allowed to use the camera function of the mobile device. Furthermore, the method initSurfaceView is initialized, which is important for the rendering of the avatars. For ARGame, the method initSurfaceView is overwritten for the rendering of avatars. In Listing 9, the method, which was overwritten, is presented. Importantly, the object, which is denoted with sensors, corresponds to the same object as in Area20MainFragment to calculate the correct and current position of the mobile device.

To position avatars within the radar, the function didUpdateLocation of Area20MainFragment is reused, which is shown in Listing 10. With the method setUpSurroundingPointsOfInterest, those avatars are determined, which are within a radius of 400m to the user. These avatars will be then displayed within the radar in the same way as displayed by Area20MainFragment.
Prior to the presentation of the procedure to initialize and start the game, Fig. 8 summarizes all classes of AREA, including those that are needed for ARGame. Most importantly, the modular design of AREA has enabled the flexible integration of ARGame.

Finally, the overall procedure of ARGame is presented. After the AREA20GameFragment is displayed, a GET request is sent to the server of the company of the city apps (CMCityMedia 2020) to eventually obtain the available games. Listing 11 shows such a request. Each ARGame has an ID, with which the available avatars for an ARGame are specified (see Listing 12).

**Listing 10** AREA didUpdateLocation Method (Android)

```java
@Override
public void didUpdateLocation(Location tLocation) {
    ...;
    areaPointOfInterest = locations.setUpSurroundingPointOfInterest(tLocation, 600);
    areaPointsView.setAdress(areaPointOfInterest);
    ...;
}
```

**Listing 11** AREA get games Request (Android)

```java
GET('/api/games')
void getAvailableGames(callback: WrappedGame[]): void;
```

Then, it will be identified whether or not a user is already subscribed to ARGame. If not, then all avatars are loaded. If the user is already subscribed for ARGame, only those avatars will be loaded that were not found by the respective user so far. The loading procedure of the avatars is done by the method addARTarget (see Listing 13).

**Listing 12** AREA get avatars for a game Request (Android)

```java
GET('/api/games'(gameId)/ar targets')
void getAvailableArTargets(callback: WrappedARTarget[]): void;
```

![Class diagram with ARGame extensions on android](image)
Flexible development of location-based mobile augmented reality applications with AREA

ARTargets are handled by AREAPointOfInterest in Area20MainFragment. They are sent to the poiStore by the Area20MainFragment in order to finally display the ARTargets on the radar. After that, the ARTargets are handed over to the ARGamingSurfaceView method (see Listing 14), which, in turn, hands them further over to the ARGamingRenderer.

7 Discussion

Currently, AREA is used in various scenarios in everyday life (CMCityMedia 2020). Three aspects are particularly important for this extensive use. First, the algorithm framework...
shown in this work (see Fig. 1) was bundled into the AREA kernel, including its modular architecture (Pryss et al. 2017a, b). Following this, the development of business applications on top of AREA becomes easily possible. This positive characteristic was mainly shown in this work based on the development of ARGame. Second, AREA reveals a proper user experience with respect to robustness and performance. The achieved robustness was also an important pillar for the development of ARGame. Experiments conducted with AREA (Pryss et al. 2017a, b) confirm that it is competitive to mobile applications that provide the same or a similar feature set at the time of conducting the experiments. Third, AREA provides the same feature set on Android and iOS. The ability to cope with the peculiarities of the different mobile operating systems, while providing the same features on all of these mobile operating systems, is highly welcome in practice. However, to keep pace with the frequent updates of the underlying mobile operating systems on one hand, and to continuously implement new features that emerge in practice on the other hand, is still a very challenging endeavor. Therefore, insights into frameworks and operating principles as shown in this work are of utmost importance. Nevertheless, in future experiments, AREA must prove its performance compared to ARKit Apple (2019) and ARCore Google (2020). In general, performance measures are currently missing for two other important aspects. First, it must be quantitatively measured how developers consider AREA compared to other solutions in terms of time and working costs. Second, it must be measured in more experiments, how AREA competes with existing other frameworks as well as other existing mobile applications. In the light of these limitations, the utilization of AREA in commercial scenarios shows its feasibility in practice.

8 Summary and outlook

This work provided insights into the development of the AREA framework. As the first contribution, a comprehensive overview of the implemented algorithms to enable location-based mobile augmented reality applications was presented. On top of this, the development of ARGame was presented, a serious game, which is used for commercial purposes. In general, the development of mobile applications is demanding when considering the peculiarities of the different mobile operating systems on the market. To cope with this heterogeneity, AREA provides the same functionality for business applications that are developed based on top of it for Android and iOS. This is enabled by enclosing all features in the AREA kernel. To show how this is technically accomplished, all steps to integrate ARGame into AREA were presented. Following such implementations, application developers can utilize AREA like ARKit from Apple or ARCore from Google to easily create their own location-based mobile augmented reality applications. Furthermore, as this work provides implementation details, this may help the community in using the insights for further developments and improvements. In this context, it was also shown that frameworks like AREA should be quantitatively assessed. Hence, AREA has been evaluated in experiments (Pryss et al. 2017a, b). These experiments have shown that AREA reveals considerable performance results compared to other mobile augmented reality applications providing a similar feature set. Furthermore, it was reported that currently conducted experiments investigate how AREA competes with ARKit and ARCore. Another experiment investigates a new feature that was developed on top of the track and area algorithm. Altogether, mobile augmented reality
Flexible development of location-based mobile augmented reality applications with AREA

Applications can support many new scenarios and fields. However, powerful solutions that can be easily developed across the different mobile operating systems in the same way are becoming more and more important. Finally, considerations on psychological factors should be also taken into account when using applications in the virtuality-reality continuum like AREA. Therefore, such investigations are pursued in future work on AREA.

Acknowledgements Open Access funding provided by Projekt DEAL.

Compliance with ethical standards

Conflict of interest Authors R.P. and M.S. have received financial support for consultation services from the CMCityMedia GmbH.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

Apple (2019) Arkit. https://developer.apple.com/arkit/. Accessed 20 Mar 2020
Apple (2020) Arcamera. https://developer.apple.com/documentation/arkit/arcamera/. Accessed 20 Mar 2020
Apple (2020) Healthkit. https://developer.apple.com/healthkit/. Accessed 20 Mar 2020
Beierle F et al (2019) What data are smartphone users willing to share in mixed reality: usability study. IEEE Access 7:71921–71932. https://doi.org/10.1109/ACCESS.2019.2919162
Ibili E, Billinghamurst M (2019) Assessing the relationship between cognitive load and the usability of a mobile augmented reality tutorial system: a study of gender effects. Int J Assess Tools Educ 6(3):378–395
Jang S, Liu Y (2019) Continuance use intention with mobile augmented reality games: Overall and multigroup analyses on pokémon go. Inform Tech People
Jung T, Lee H, Chung N et al (2018) Cross-cultural differences in adopting mobile augmented reality at cultural heritage tourism sites. Int J Contemporary Hospitality Manag 30(8):
Kähäri M, Murphy D (2006) Mara: sensor based augmented reality system for mobile imaging device. In: 5th IEEE and ACM Int’l symp on mixed and augmented reality, vol 13
Kim W, Kerle N, Gerke M (2016) Mobile augmented reality in support of building damage and safety assessment. Natural Hazards Earth Syst Sci 16(1):287
Koh LK (2019) Tourism system using mobile augmented reality and location-based services (lbs). https://dr.ntu.edu.sg/handle/10356/136510
Kooper R, MacIntyre B (2003) Browsing the real-world wide web: maintaining awareness of virtual information in an ar information space. Int’l J Hum Comput Interaction 16(3):425–446
Korinth M, Sommer-Dittrich T, Reichert M, Pryss R (2020) Design and evaluation of a virtual reality-based car configuration concept. In: Arai K, Kapoor S (eds) Advances in computer vision. CVC 2019. Advances in intelligent systems and computing, vol 944. Springer, Cham
Laine TH, Suk H (2019) Designing educational mobile augmented reality games using motivators and disturbance factors. In: Geronemenko V (ed) Augmented reality games II. Springer, Cham
Lee R, Kitayama D, Kwon Y, Sumiya K (2009) Interoperable augmented web browsing for exploring virtual media in real space. In: Proc of the 2nd int’l workshop on location and the web, ACM, pp 7
Lee T, Hollerer T (2008) Hybrid feature tracking and user interaction for markerless augmented reality. In: Virtual reality conference. IEEE, pp 145–152
Lee Y, Rhee S (2015) Efficient photo image retrieval system based on combination of smart sensing and visual descriptor. Intell Auto Soft Comput 21(1):39–50
Leighton L, Crompton H (2017) Augmented reality in K-12 education. In: Mobile technologies and augmented reality in open education, IGI Global, pp 281–290
Liao TC-L, Yang H, Lee S, Xu K, Feng P, Bennett S (2018) Augmented criminality — how mobile augmented reality crime overlays affect people’s sense of place. AoIR Selected Papers of Internet Research, 6. Retrieved from https://journals.uit.edu/ojs/index.php/spir/article/view/8437
Liou H, Yang S, Chen S, Tarng W (2017) The influences of the 2D image-based augmented reality and virtual reality on student learning. J Educ Technol Soc 20(3):110–121

Springer
Litts BK, Lewis WE (2019) Mobile augmented reality: exploring a new genre of learning. GetMobile 22(3):5–9
Liu L, Li H, Gruteser M (2019) Edge assisted real-time object detection for mobile augmented reality. In: The 25th annual international conference on mobile computing and networking (MobiCom’19). Association for Computing Machinery, New York, NY, USA, Article 25, pp 1–16. https://doi.org/10.1145/3300061.3300116
Majeed T, et al. (2019) What is of interest for tourists in an alpine destination: personalized recommendations for daily activities based on view data. J Ambient Intell Human Comput pp 1–12
Mladenovic R et al. (2019) Effectiveness of augmented reality mobile simulator in teaching local anesthesia of inferior alveolar nerve block. J Dental Educ 83(4):423–428
Paucher R, Turk M (2010) Location-based augmented reality on mobile phones. In: IEEE computer society conference on computer vision and pattern recognition workshops, IEEE, pp 9–16
Pryss R, Geiger P, Schickler M, Schobel J, Reichert M (2016) Advanced algorithms for location-based smart mobile augmented reality applications. Proc Comput Sci 94:97–104
Pryss R, Geiger P, Schickler M, Schobel J, Reichert M (2017a) The AREA framework for location-based smart mobile augmented reality applications. Int J Ubiquitous Syst Pervasive Netw 9(1):13–21
Pryss R, Schickler M, Schobel J, Weilbach M, Geiger P, Reichert M (2017b) Enabling tracks in location-based smart mobile augmented reality applications. Proc Comput Sci 110:207–214
Rauschnabel P, Rossmann A, Tom Dieck M. (2017) An adoption framework for mobile augmented reality games: the case of Pokémon Go. Comput Hum Behav 76:276–286
Reitmayr G, Schmalstieg D (2003) Location based applications for mobile augmented reality. In: Proc of the fourth Australasian user interface conference on user interfaces, Australian Computer Society, Inc., pp 65–73
Ren J, He Y, Huang G, Yu G, Cai Y, Zhang Z (2019) An edge-computing based architecture for mobile augmented reality. IEEE Netw 33(4):162–169. https://doi.org/10.1109/MNET.2018.1800132
Sánchez-Acevedo MA, Sabino-Moxo BA, Márquez-Domínguez JA (2018) Mobile augmented reality: evolving human-computer interaction. In: Management Association I (ed) Virtual and augmented reality: concepts, methodologies, tools, and applications. IGI Global, pp 200–221. https://doi.org/10.4018/978-1-5225-5469-1.ch010
Schickler M, Pryss R, Schobel J, Reichert M (2015) An engine enabling location-based mobile augmented reality applications. In: 10th int’l conf on web information systems and technologies (Revised Selected Papers), no. 226 in LNBIP, Springer, pp 363–378
Schickler M et al (2019) The area algorithm framework enabling location-based mobile augmented reality applications. Proc Comput Sci 155:193–200
Tosun N (2017) Augmented reality implementations, requirements, and limitations in the flipped-learning approach. In: Mobile technologies and augmented reality in open education, IGI global, pp 262–280
Vlahakis V, Karigiannis J, Tsotros M, Ioannidis N, Stricker D (2002) Personalized augmented reality touring of archaeological sites with wearable and mobile computers. In: Sixth international symposium on wearable computers, IEEE, pp 15–22
Yang Y, et al. (2016) Mobile augmented reality authoring tool. In: 10th int’l conf on semantic computing, IEEE, pp 358–361

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.