Univ Access Inf Soc (2019) 18:49–61
https://doi.org/10.1007/s10209-017-0584-2

LONG PAPER

Designing and implementing interactive and realistic augmented reality experiences

Alvaro Montero[1] · **Telmo Zarraonandia**[1] · **Paloma Diaz**[1] · **Ignacio Aedo**[1]

Published online: 6 November 2017
© The Author(s) 2017. This article is an open access publication

Abstract In this paper, we propose an approach for supporting the design and implementation of interactive and realistic Augmented Reality (AR). Despite the advances in AR technology, most software applications still fail to support AR experiences where virtual objects appear as merged into the real setting. To alleviate this situation, we propose to combine the use of model-based AR techniques with the advantages of current game engines to develop AR scenes in which the virtual objects collide, are occluded, project shadows and, in general, are integrated into the augmented environment more realistically. To evaluate the feasibility of the proposed approach, we extended an existing game platform named GREP to enhance it with AR capacities. The realism of the AR experiences produced with the software was assessed in an event in which more than 100 people played two AR games simultaneously.

Keywords Authoring tools · Augmented reality · Mixed reality · Interactive environments

✉ Alvaro Montero
ammontes@inf.uc3m.es

Telmo Zarraonandia
tzarraon@inf.uc3m.es

Paloma Diaz
pdp@inf.uc3m.es

Ignacio Aedo
aedo@ia.uc3m.es

[1] Universidad Carlos III de Madrid, Avda de la Universidad 30, 28911 Leganes, Madrid, Spain

1 Introduction

The possibilities that AR technology offers for superimposing digital information on the users' view of the real world have been already exploited in several areas such as education [18], medicine [13] or entertainment [23]. During the last few years, AR technology has become more accurate and has reduced the latency produced by the image processing improving greatly the immersion of the user in the augmented world. Also, AR glasses are now available for the general public increasing the possibilities of a broader adoption of this technology.

According to Azuma [2], the users of an ideal AR system should have the impression that the real and virtual objects coexist in the same space. However, most AR applications are still far from providing this type of experience, failing to achieve a seamless integration between the elements of the real and virtual environments. The computer-generated images usually overlap the vision of the user. Although this approach might be adequate in certain cases, for example when augmenting a real object with additional information, the results obtained for AR games or AR reconstructions of historical sites might not be satisfactory.

In addition, the design and implementation of AR experiences are still beyond the reach of non-expert users. We believe that the full potential of this technology will only be revealed when end-users adopt it and start creating their own AR artefacts. Although tools like Wikitude [28] and ATOMIC [1] have contributed to simplify the process of augmenting objects or environments with digital information, the AR experiences obtained lack the realism Azuma advocated.

Following these ideas, the final goal of this work is to put in the hands of end-users, who might lack the technological expertise, the tools to design and develop realistic

AR experiences, thereby reducing or eliminating the need of technical assistance during the process. As a first step towards this goal, we present a technique for implementing interactive and realistic AR experiences. The proposed approach combines the use of model-based AR techniques with the benefits that current game engines offer for the design and management of 3D virtual environments. To evaluate the feasibility of the proposed approach, the level of realism of the AR experiences produced and also to know first-hand the difficulties of implementing them AR experience with the features advocated by Azuma, we developed two crowd-controlled AR games using an existing game engine, the GREP Platform. The games were tested in an experience carried out in the auditorium of the University Carlos III of Madrid in which more than 100 people played them simultaneously.

The rest of the paper is structured as follows. In the following section, some related works are presented. Next, the proposed approach for designing realistic and interactive AR experiences is introduced. Then, we present a system prototype that extends the capabilities of a game engine to support the creation of AR experiences. Finally, the evaluation carried out to assess the effectiveness of the approach is described and the results discussed. At the end of the paper, some conclusions and suggestions for future work are outlined.

2 Related work

A variety of techniques have been proposed to enhance the realism of AR experiences. Among them, the problem of occlusion has been thoroughly examined in previous research [4, 11, 12, 14], as occlusion errors greatly contribute to destroy the feeling that the AR objects coexist in the real world [22]. The problem arises because AR objects should appear as partially or totally hidden behind real objects when these block the line-of-sight between the observer and the AR object. The implementation of occlusion effects requires a mechanism for detecting when occlusions occur and an appropriate rendering of the images so that the hidden parts of the AR objects are not displayed.

It is necessary to note here that the solution to the problem might be different depending on the type of AR display system used such as see-through and video [24]. In any case, most solutions are based either in the use of depth maps [4], 3D models of the environments [4] or contour models [3]. Depth maps assign a distance value to each pixel of the image and can be automatically produced using depth cameras or stereo cameras. This information allows the software to determine if the virtual object is in front of or behind a real one, not requiring any prior preparation of the environment for detecting occlusions. Nevertheless, this technique is not adequate for augmenting big spaces, as the scope of depth cameras is limited and the precision of stereo cameras rapidly decreases for large distances. This problem can be overcome using model-based occlusion techniques [4, 11, 14], which make use of a 3D model of the real environment to determine which virtual elements should be occluded. In this case, the disadvantage is that it is necessary to create the model and align it with the real objects before the experience. Finally, it is also possible to implement occlusion effects using the contours of the objects in the environment [3]. Despite the fact that this option does not require to design a whole model of the scenario to augment, it is still necessary to indicate for each contour which other contours are behind and in front of it.

The realism of the experience would also decrease if the AR objects move around the environment, colliding with real bodies. This problem is similar to the one posed by the occlusion effect: to detect a collision, it is necessary to first identify the contour of the object in the real world and to determine its distance to the virtual object. As in the previous case, most solutions to the problem make use of depth maps or virtual models of the environments [4]. However, in this case, depth maps perform worse than virtual models, as they do not allow the viewer to identify the collisions that are occluded by real objects.

A third issue, to give the user the impression that virtual and real objects coexist in the same space, is that the former should experience the effects of gravity. For example, if a virtual object is dropped from a height, it should fall until it strikes the floor or a horizontal surface, describing a trajectory analogous to the one a similar real object would follow. The authors of [21] propose a solution to the problem for hand-held AR devices, such as smartphones, which make use of the inertial sensors in the instrument, such as its accelerometers, to compute the gravity vector to apply.

Shadows and lighting play an important role in the realism of the experience [15, 27]. AR objects should reflect the light and produce shadows in the same way as the real objects placed in the augmented environment. In [27], the authors propose a method to estimate illumination conditions of an image based on the shading and shadows it casts. Once the illuminant directions are obtained synthetic, AR objects are rendered with the correct illumination effects. The augmentation of outdoor scenes poses additional challenges to the ones of indoor settings due to the higher number of parameters that might reduce the verisimilitude of the integration of the virtual objects in the real world [19]. For example, to achieve a realistic illumination, the designer might need to consider not only the position and insensitive of the sun or any other light source, but also the colour of the sky and its illuminance [16]. In addition, the realism can also decrease if the virtual objects do not behave and respond to the atmospheric conditions in the way the real

objects do. The AR systems presented in [17, 26] are able to represent and simulate the effect of the wind flow in the virtual objects in a scene.

Finally, the sound can also help to produce a cohesive integration of the virtual and real elements. Not only can it be expected that the virtual objects would produce sound in the same way as the real counterparts, but it has also been demonstrated that the use of spatial sound in AR experiences contributes to the perception of depth and position [25, 30].

All the works mentioned previously aim to enhance the realism of the AR experience by modifying the characteristics and properties of the virtual objects in the scene. The authors of [6, 9, 10] approach the problem from another perspective. In these cases, the objective is to modify and adjust the users' view of the real environment in order to produce the sensation that this responds realistically to the presence or actions performed by the virtual elements in the scene. On the one hand, in [9, 10], stylization algorithms are applied to modify the image captured by the camera to give the real objects a visual appearance similar to the virtual ones. In this specific case, it might be arguable to say that the result obtained is realistic. In any case, the coherence of the AR scene produced is greatly enhanced, making it difficult for the viewer to distinguish between real and virtual objects. On the other hand, image analysis algorithms are used in [6] to generate physically plausible animations of objects in the video captured by the camera. This can be used to give the viewer the impression that objects vibrate or are deformed responding to forces or actions carried out by the virtual elements.

Table 1 summarizes the different realistic effects reviewed and the proposed solutions. As shown in the table, the realism of an AR experience can be enhanced by implementing many different types of effects. However, the solutions currently proposed only address one or two specific features. What is needed is a more general approach to allow the designer of the AR experience to choose among a variety of effects and to use them in combination to achieve a seamless integration of virtual elements in a real scene. The work of this paper is focused on covering the realistic effects of Occlusion, Collision, Gravity, Light Reflection, Shadows, Wind and Sound.

3 Approach

The main goal of this work is to support the design and implementation of realistic and interactive AR experiences. Following this aim, we propose to combine AR model-based techniques with the possibilities that current game engines offer for implementing interactive three-dimensional virtual spaces. This type of software provides different components for supporting the design of realistic virtual scenes, such as physics engines, collision detection, textures, lighting and shading. Furthermore, they also supply the means for specifying the way the virtual elements should behave and react to the users' commands as well as the state of the other elements of the scene. Therefore, we propose to set up the AR scenes as virtual scenes that take place in a three-dimensional virtual replica of the environment to augment. This replica should be aligned with the users' vision of the environment, so that the virtual elements will appear as superimposed to the corresponding elements in the real world. Once the two worlds are aligned, the background virtual elements, such as floors or walls, will be made invisible. As a result, the remaining virtual elements will appear as integrated into the real world. For example, the further they are, the smaller they would look in the users' view. In the same way, as the virtual background elements are invisible and aligned with real objects, when a virtual object collides or passes behind one of them, the user will be given the impression that it bumps into or is hidden behind the corresponding real one, respectively. The tasks required to create AR

Table 1 Summary of techniques and realistic effects

Techniques	Effects							
	Occlusion	Collision	Gravity	Light reflection	Shadows	Wind	Reality adjustment	Sounds
Model-based	[4]	[4]						
Depth maps	[4]	[4]						
Contour model	[3]							
Inertial sensor			[21]					
Light estimation				[15, 27]	[15, 27]			
simulation of forces			[4]			[17, 26]		
Stylization algorithms							[9, 10]	
Physical modelling							[6]	
Spatial sound modelling								[25, 30]

experiences following this approach can be organized in four different groups:

- *Environment modelling* Which results in the definition of a 3D replica of the environment to augment;
- *AR scene composition* Which includes the alignment of the 3D replica with the users' view of the environment that represents and the definition the main elements of the scene;
- *Simulation of the environmental conditions* In which the AR scene is enhanced with realistic effects to resemble the illumination and atmosphere of the real setting;
- *Definition of the scene animation and user interactions* In which the virtual objects are animated and the rules that govern the interaction of between them and the users are specified.

At the end of this section, Table 2 shows a summary of the tasks that includes recommendations and suggestions.

3.1 Environment modelling

The first step in the definition of the AR experience is to build a 3D replica of the space to augment. The 3D reconstruction is a well-studied area of research in computer vision. Different authors have proposed techniques for acquiring the 3D input data using laser range scanner [5], depth cameras [8], SLAM (Simultaneous Localisation and Mapping) systems [7], etc. However, the selection for environment modelling way must keep in mind that it will be able to be carried out by end-users. As commented in Sect. 1, this paper is focused on presenting a technique for implementing interactive and realistic AR experiences as a previous step to empower end-users to create their own AR experiences. Therefore, it is necessary to provide them with the means to carry out the modelling process by themselves. Ideally, these means should support the modelling of a wide range of environments, either indoor or outdoor. At the same time, they should not require a high level of expertise to be used, or to rely on too specific or expensive instruments that the user might not find available. These requirements discard, for example, the use of depth or stereo cameras as they are not adequate to be used in large-size buildings, as the elements at the back end might be impossible to distinguish.

Following these ideas, we tested three different approaches for capturing 3D models of environments:

- *Modelling using SLAM systems* Although these systems are normally used for building maps of unknown environments, it is possible to use them for creating 3D reconstructions of objects or environments, moving the camera or capturing device around the space to model [7]. The main advantage of this approach is that the modelling process can be carried out semi-automatically. However, the accuracy of the model produced depends greatly on the characteristics of the environment. Better results are obtained for small settings than for large ones, outdoor locations or places lacking distinctive features that the software can recognize and use as landmarks. Furthermore, the whole environment is modelled as one single shape. This would make necessary to split up the 3D model obtained in different parts or blocks, so that later on we can specify which of them produce occlusion or transparency.
- *Modelling using the information in a building plan or a map* Another possibility is to design a 3D replica of the space to augment using the dimensions and distances provided by a building plan. The main advantage of this method is that it does not require to use special cameras and devices. Once the model is produced, the scale at which it should be displayed in the augmented scene can be determined by measuring the distance from the AR viewer to the closer element of the environment represented virtually. We tested this approach modelling environments of different sizes, and the results obtained were satisfactory. Nonetheless, the process was too tedious and complex to be carried out by users who might lack expertise in modelling. Moreover, the process also depends on the availability of plans of the environment to augment.
- *Modelling the environment in the virtual space* The 3D model can be built directly in the virtual space, arranging blocks that represent the paths the virtual characters could walk through, the objects they could collide with and the elements that could produce occlusion. To help the designer align the blocks with the corresponding parts of the real setting, an image of the environment can be projected behind the virtual model. Using this image as a reference, the designer can place and modify the blocks until they overlap correctly in the image of the corresponding element of the environment. The main advantage of this approach is that the model produced is already scaled and aligned with the image of the environment to augment. Furthermore, as the model is composed of a collection of blocks, their properties can be specified separately.

The pictures of Fig. 1 illustrate the procedure followed for modelling an environment as a collection of 3D blocks. The left picture of Fig. 1 depicts a picture of a setting that includes indoor and outdoor areas separated by a window. The right picture of Fig. 1 shows the model prepared for augmenting the scene. In this picture, the blocks used to represent the different parts of the setting are depicted with different colours to facilitate differentiation. As shown in the picture, the indoor area has been modelled using two blocks

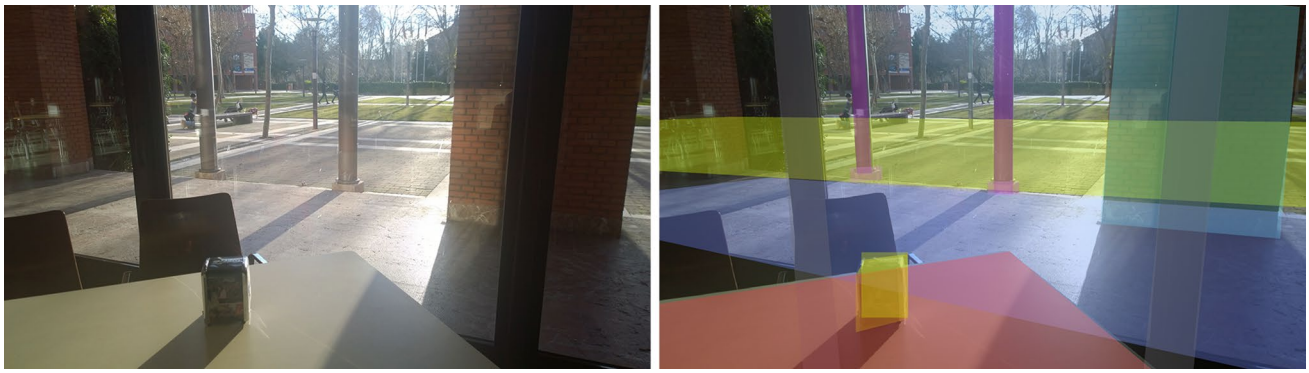


Fig. 1 Example of the modelling of an environment using 3D blocks

to represent the table and the napkins holder. For the outdoor area, two blocks model the two levels of the floor, 5 blocks more to represent the columns and a last one for the window.

3.2 AR scene composition

Once the virtual replica of the setting to augment has been modelled, the next step is to define an AR scene based on it. As previously explained, to support realistic AR experiences, the virtual elements should be aligned with the elements of the real world that represent. In addition, it is necessary to specify which parts of the 3D model of the environment will produce occlusions, semi-transparencies as well as the ones that the virtual elements can collide with. The AR scene can be composed as a VR scene that contains the following elements:

- A background video of the environment to augment projected on a pane in the back of the virtual world;
- A virtual camera positioned to capture the scene with the same field of view than the video camera that captures the video of the real world;
- The virtual model of the environment to augment placed between the virtual camera and the background video pane. The scale and position of the model should be adjusted so that when looking from the position of the virtual camera to the background video, the model overlaps the images of the real objects that it replicates.

Once the main elements of the AR are correctly positioned and aligned, the virtual objects and characters to augment the real world should be placed in the virtual model at the position they would occupy in the real setting. The next step is to modify the textures of the different parts of the model of the environment to make them invisible to the AR viewer. This will make the virtual objects appear overlaying the background video, at a position, height and size that will support the sense of perspective and the feeling that they are part of the real world. The textures to apply to the elements of the environment model will depend on the type of the part of the real world they represent and its position from the AR viewers' point of view. For example:

- *Non-occluding parts* The parts of the model that represent floors, paths, walls at the sides or the back of the scene and, in general, any part of the model behind which the virtual objects cannot be hidden can be made disappear using an invisible texture.
- *Occluding parts* The parts of the model that produce occlusion can be made invisible using a texture displaying the portion of the background video that is hidden by their shape. This will cause the parts of the virtual objects behind them to not be displayed in the final video of the AR scene.
- *Semi-transparent parts* The translucent effect produced by the parts of the model that represent glasses or semi-transparent surfaces can be achieved applying them filters that modify the colour or deform the objects behind them.

Finally, it is necessary to specify the parts of the environment model that the virtual objects can collide with. In most game engines, collision detection is supported by an invisible component attached to the 3D models named collider. The physics engine detects when two colliders contact and trigger the corresponding collision effect. Therefore, collisions in the AR scene can be implemented just by adding or activating colliders for the parts of the model that the virtual objects can hit with. As these models are invisible to the AR viewer, it will have the impression the virtual objects are colliding with the real elements displayed in the background video.

Figures 2 and 3 illustrate an AR Scene composed for the example introduced in the previous section. The left picture in Fig. 2 shows the way the camera and video-background pane were organized to keep the model aligned with the view

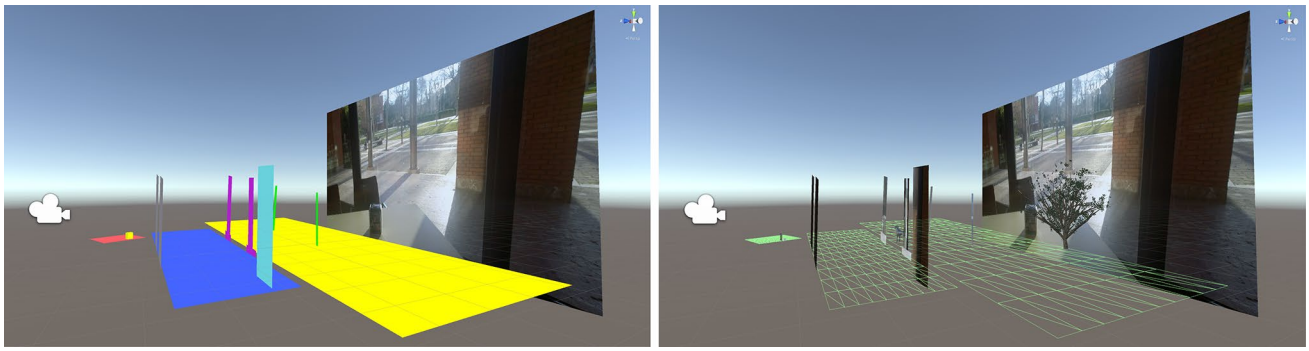


Fig. 2 Example of an AR scene composition

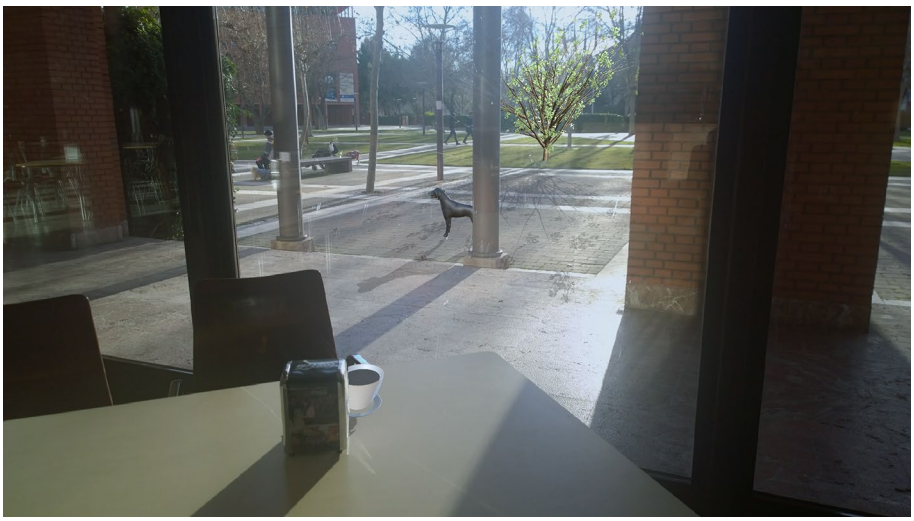


Fig. 3 View of the augmented scene from the perspective of the AR viewer

of the real world. The right picture in Fig. 2 depicts the way the scene looks in the virtual world after placing the virtual objects and making the model of the environment disappear. In this case, the virtual objects used to augment the scene were a cup of coffee, a dog and a tree in the back of the scene. As shown in the picture, the floors and the table were made invisible, whereas the columns were assigned with a video texture to produce the occlusion effect. Figure 3 illustrates the final view from the perspective of the AR viewer of the augmented scene.

3.3 Simulation of the environmental conditions

As previously stated, the realism of the AR scene can be enhanced by simulating in the virtual world the illumination and atmosphere conditions. Current game engines provide several components and camera-effects for creating realistic atmospheres in virtual scenarios, including light sources, wind simulators, gravity settings or spatial sound, among others:

- *Virtual torches* Light and shade effects can be implemented by adding and orienting virtual torches to the initial AR scene. Modifying the position, orientation and intensity of the torches, the designer can introduce ambience and direct light effects to adjust the final colour of the virtual objects and their shades.
- *Wind sources* The designer can also take advantage of the wind simulators provided by current engines. By adding wind zones to the scene and adjusting their force and orientation, it is possible to reproduce the subtle effect of the wind in cloths, trees and the virtual objects that the physics engine can animate automatically.
- *Gravity* Current physics engines can also reproduce the effects of the gravity in the objects in the scene, so they fall in the direction of the ground or slip down a ramp. The designer can choose to activate or not this effect for each virtual object in the augmented scenario.
- *Spatial sound* Audio clips can be attached to the virtual objects so they produce sound as their real counterparts. The volume of the audio will be different the further or closer they are from the virtual camera that represents the AR viewer.

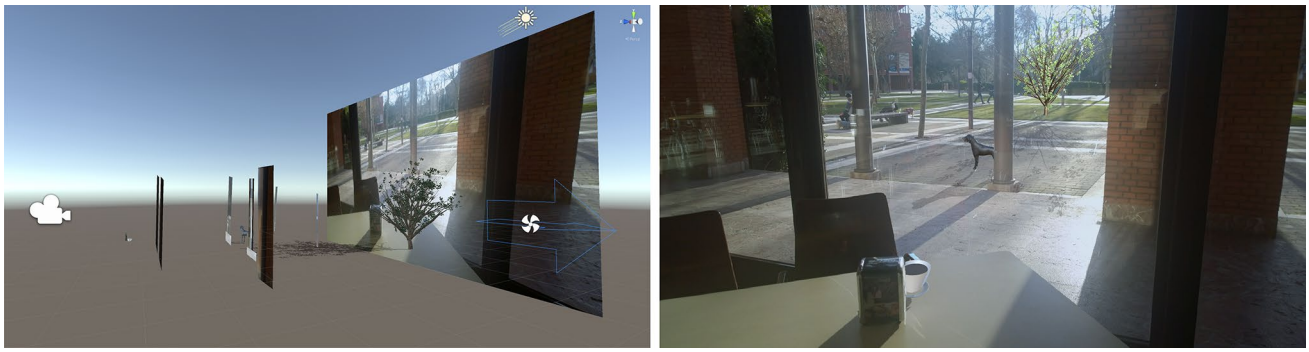


Fig. 4 Example of environmental conditions simulation in an AR scene

Figure 4 illustrates the enhancement of the previous AR scene composition with some effects for simulating the environmental conditions. As shown in the picture on the left-hand side, a virtual torch representing the sun has been added to the top of the virtual world. The orientation and intensity were adjusted until the shadow of the tree and the dog matched the direction of the shadows of the real elements in the scene. A wind source has also been added at the back of the scene to softly shake the branches of the virtual tree. The final view of the augmented scene is shown on the picture of the right-hand side.

3.4 Scene animation and user interaction

Current game engines provide sophisticated animation systems to control and sequence animations that represent the actions and states of the characters in the games. Making use of these systems, it is possible to animate AR scenes to enhance their realism or create interactive experiences. This process consists of two major tasks:

- Definition of the set of animation clips for representing the different states of the virtual objects in the scene;
- Specification of the logic that drives the activation of the animation clips.

Game engines support this second task by allowing the designer to define state machines that control when to activate an animation, repeat it, stop it or switch from one to another. Based on the type of conditions used to activate the transitions between the objects, it is possible to specify:

- *Non-interactive animated AR scenes* In which the animations are automatically activated without user intervention. Enhancing an AR scene with non-interactive animations can help to improve the realism of the scene. For example, a representation of a virtual dog will be more realistic if it moves his tail periodically;
- *Interactive AR scenes* In which the users control one or more virtual objects in the scene. In this case, it is necessary to provide the designer with the means for defining an adequate interaction style for the users to introduce the control commands in system. For example, speech recognition and hand gestures are frequently used in wearable AR systems [20]. In this case, the designer should be able to specify the commands and gestures that the system should recognize and to link them with transitions of characters animation state machines.

Figure 5 depicts a picture taken after animating the example AR scene described in the previous section. In this case, we defined four animations for representing the dog walking and one more for the idle state, which the user could activate using a gamepad.

4 AR engine prototype implementation

To evaluate the feasibility of the proposed approach, we extended a game platform named GREP (Game Rules scEnario Platform) [29] with a module to support AR experiences. The GREP platform aims to empower non-expert users to create 3D digital games by themselves, minimizing

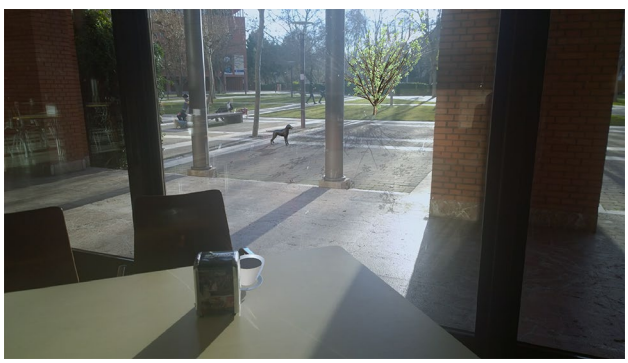


Fig. 5 Example of final AR scene

Table 2 Summary of tasks

Stage	Task
Environment modelling	Objective: to create a three-dimensional virtual model of environment to augment
	Recommendations/suggestions: small places can be modelled using SLAM systems, whereas bigger areas can be sketched using 3D blocks that represent its main elements
AR scene composition	Objective: to align the 3D replica with the user's view of the environment that represents and to define of the main elements of the scene
	Recommendations/suggestions: the scene can be composed using a background video of the environment to augment projected on a pane in the back of the model of the virtual world. The scale and position of the model should be adjusted so that when looking from the position of the virtual camera to the background video, the model overlaps the images of the real objects that it replicates. Next, the texture of the elements of the model should be modified to make them invisible to the AR viewer, applying invisible texture to the non-occluding parts, a texture displaying the portion of the background video that is hidden by their shape to the occluding parts and adding to it colour filters to the translucent parts. Finally, colliders should be added to the parts of the model the objects can impact with
Simulation of environmental conditions	Objective: to enhance the realism of the scene by simulating in the virtual world the illumination and atmosphere conditions of the real setting
	Recommendations/suggestions: take advantage of the components provided by the game engines to design realistic virtual scenes, as virtual torches, wind sources or physics engines. Audio clips can be attached to the virtual objects so they produce sound as their real counterparts
Animation and interaction	Objective: to define the ways in which the users will interact with the elements in the scene and how these behave and react to the users' actions
	Recommendations/suggestions: link the virtual objects in the scene with animation clips for representing their different states. When specifying the logic that drives the activation of the clips, include some periodic automatic activations for some of them, to improve the realism of the scene. Design and implement different types of interaction styles for introducing the control commands, so that the different preferences of the users can be accommodated

dependence on technical assistance in the process. GREP has a special focus on serious and educational games and has been developed using the Unity engine. The AR module (AR GREP) expands the GREP functionalities to support playing AR games in scenes specified following the approach described in the previous section. In the next sections, we briefly introduce the GREP system and describe the implementation of the AR module.

4.1 GREP

The GREP system provides a set of tools to allow the rapid definition of digital games designs, which can be exported as XML files and interpreted and processed to generate 3D games. More specifically, it provides two core applications: the *GREP Editor*, which supports the design of the game scenes and the definition of the game rules, and the *GREP Player*, which allows the players to retrieve game design files and generates a 3D virtual environment to play them.

The *GREP Editor* includes a graphical interface tool that allows the designer to create static scenes by adding, modifying or removing graphical resources retrieved from different repositories of assets. The user can start defining the scene from scratch, specifying its size, floor texture and background elements. Alternatively, he or she can select *pre-defined environment models* from the ones available in the platform resources repository. These can define a whole dungeon or a farm that the user can enrich and modify adding new elements to it. When the definition of the scene is completed, the user can start describing actions and behaviours for the elements and characters in the scene and rules of games to be played. To speed up the game definition process, the tool allows to describe them as combinations of more simple games that can be played simultaneously or in sequence at each different game stage. When the definition of the scene and the game is completed, the design can be exported as an XML file.

The *GREP Player* is a game engine able to process descriptions of digital games specified using the *GREP Editor* and to generate 3D games based on them. Firstly, the player reconstructs the scenes described in the XML files of the game, retrieving from the platform repositories the resources referenced in them and instantiating them at the positions, size and orientations requested. Once the virtual scene is reconstructed, the player processes the part of the file containing the game rules definition, configuring and activating an appropriate set of animations and controls to implement them.

4.2 AR GREP

The new AR module developed for the GREP platform allows playing AR games created using the *GREP Editors* and a 3D model of the environment to augment. The players of the AR games visualize the game action as displayed on top of a video of the real world captured by a camera positioned in a fixed location. The characters and objects in the games move and interact in an invisible virtual 3D replica of the environment currently displayed in the video, giving the players the impression that they are integrated into the real world.

Following the approach described in the previous section, AR GREP supports this type of experiences using a special type of VR scene (*AR Scene*) that includes a fixed virtual camera and a background pane to render the video captured by the camera of the computer or device that runs the system. The distance of the pane to the camera is specified in the game description contained in the XML files. These also include a reference to a new type of *pre-defined environment model* that when instantiated at the corresponding position between the camera and the background pane produces the overlapping effect. This new type of model is composed of a set of 3D blocks, each of which includes a properties section specifying its level of transparency and occlusion. The AR GREP model processes this information for each block in the model and selects the appropriate texture to apply to produce the desired effect. Finally, to support the recreation of the ambience of the real setting, the AR GREP can interpret tags in the game XML files specifying the position, orientation and intensity of virtual torches, wind boxes and the activation of the gravity force for each virtual object in the scene. The definition of the latter has been enriched to include not only the animations used to represent their states but also audio clips to play when these are activated.

Currently, the AR GREP Editors do not provide support for designing the 3D models to represent the environment. This process needs to be carried out using the Unity editor, adding and modifying blocks directly in the AR scene, following the approach described in Sect. 3.2. To support this process, the platforms asset repositories provide a new type of 3D block model that includes a script with the properties of the block to specify. Once the model is built, the designer adds it to the platform repositories as a new type of *pre-defined environment*, so that it can later be retrieved and used in the *GREP Editors*. At present, the position of the background pane and the distance to the camera required to overlap the 3D model to the video should be directly specified by the designer in the XML file. The ambient elements should also be included in the XML files manually.

5 Evaluation

The aim of the evaluation was to test the level of realism of the AR experiences implemented using the approach in a real setting. With this objective, two AR games were developed for the AR GREP Player. The games were played in the auditorium of the University Carlos III of Madrid, in the context of an event organized as part of the European Researchers Night. This was an open event, so the audience was composed of not only university staff and students, but also people of all ages with an interest in science. In total 232 people attended that night. The game activity took place in the auditorium itself, with the characters moving along its corridors and around the audience. To merge the game elements and the real world, a camera was placed in the centre of the auditorium stage, looking at the audience stands. The image captured by the camera was processed by the AR GREP module, which added to it the virtual representations of the game. Participants watched the video projected on a large screen placed on the stage in front of them.

The implementation of the games required modelling the auditorium stands, which has a capacity of 1052 seats. The seating area is 29 m wide and 24 m long, and it slopes obliquely downward towards the stage. The rows of seats are divided into 5 blocks separated by two aisles that cross the auditorium horizontally and another five aisle stairs that cross it vertically. The modelling process was carried out following the approach described in the previous section. First of all, a picture was taken with a camera from the centre of stage looking at the stands of the public. This image was rendered in the background pane of the AR scene of the AR GREP module, while the virtual camera was positioned and orientated so that its field of view captured the image. Once the reference image and the camera were configured, we started modelling the corridors and the stairs. For this part of the model, we required nine blocks that were connected with each other replicating the paths the virtual characters could follow. Next, the tilt of the stairs was adjusted until each block perfectly overlapped in the background video the corresponding real element it represented. To prevent the characters from falling out of the scene, walls were added at the sides and ends of the corridor, delimiting all the paths. Finally, a last set of blocks was added to the scene on top of those paths partially hidden by walls and rows of seats. In total, we used 28 blocks to model the auditorium.

Once the model was finished, the properties of the blocks were specified. The ones used to represent the floor and the paths borders were made transparent and had a collision property, while the ones that hide parts of the paths were assigned occlusion. For this experience, it was not necessary to apply semi-transparency. At the end of this process, the auditorium model was exported as a Unity prefab and

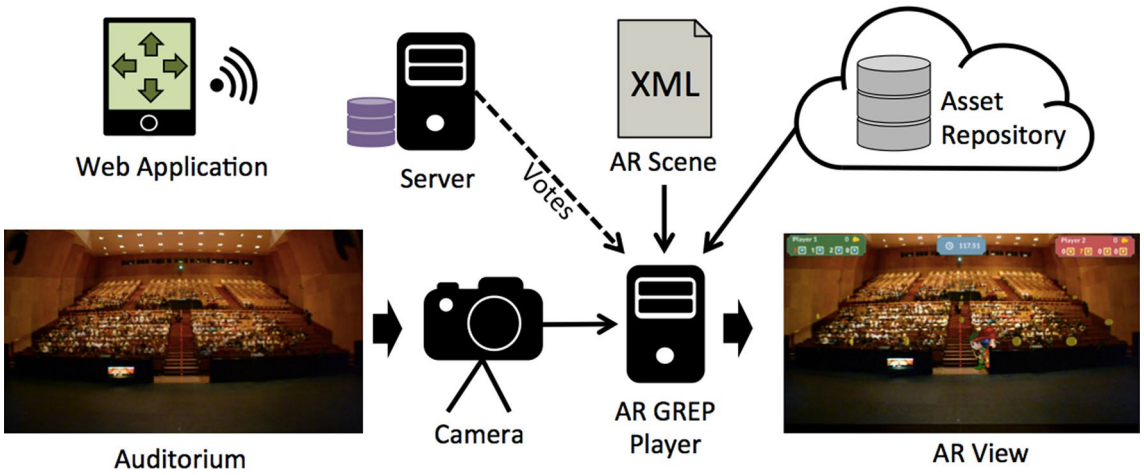


Fig. 6 Architecture of the system

stored in the GREP platforms repository as a pre-defined environment.

In order to involve all the audience in the experience, we designed two crowd-controlled games to play with the auditorium model created. The games required to organize the audience in two teams, each of which would control a different game character in real time. The participants would use their mobile phones to vote for the next movement of their character, and the one that obtained most votes per second was executed. In the first game, the characters had to collect some coins scattered around the passages of the auditorium. The one who collected more coins before the time limit expired won the game. In the second game, each of the characters played the role of prey and predator, respectively, so that the predator had to capture the prey before the time limit expired. Both games required the audience to coordinate their choices about the next movement they wanted the character to perform. The design and implementation of the games were carried out for the most part in the AR GREP Editors. The game scenes were created adding to the model of the auditorium the coins and the characters required for the games. In addition, virtual torches were added at the top of the virtual world, oriented to the virtual model in the same way as the ones placed in the real auditorium. As the auditorium was an indoor space, we did not require to use wind boxes, and the direction and force of the gravity vector were not changed. Once the scenes were completed, the rules that govern the action of the games were described, and the final design exported as GREP XML files. Finally, we implemented a web application for collecting the votes from the audience and added a new communication module to the GREP Player to be able to control the characters movement with the results of the voting process. Figure 6 depicts the final architecture of the system.

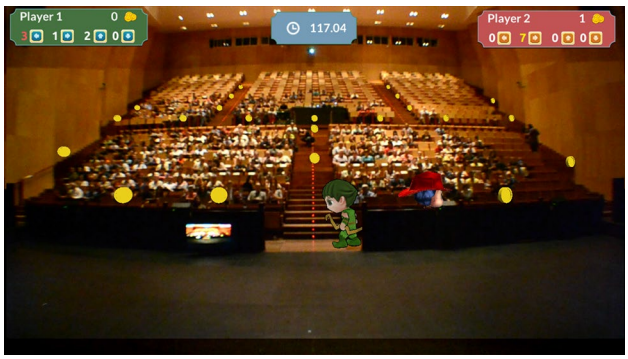


Fig. 7 Screenshot of AR collecting coins game

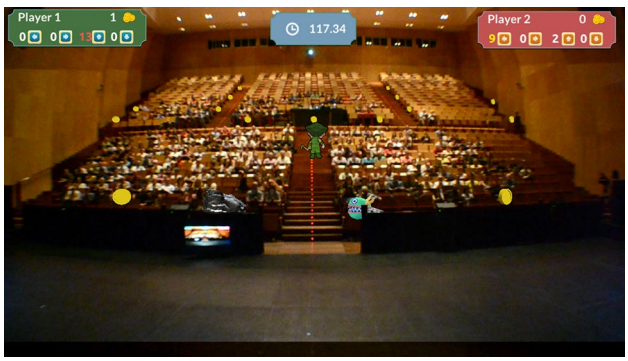
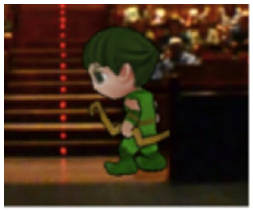
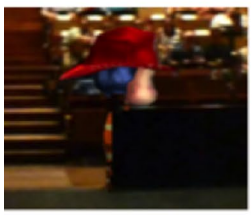
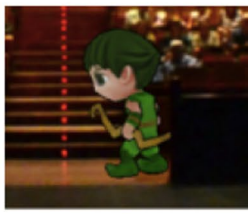
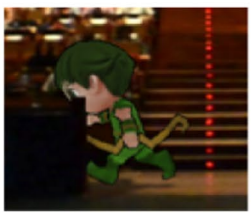
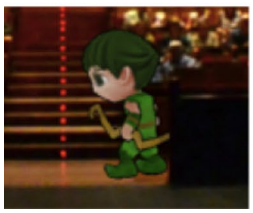
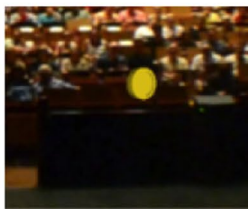


Fig. 8 Screenshot of AR prey–predator game

Figures 7 and 8 depict two screenshots taken during the plays of the first and second game, respectively. As shown in the pictures, on the top left and right side of the screen, two panels displayed the movement votes received from the audience teams in the last second, whereas the time was displayed on the top centre. The games were played by more than a hundred people simultaneously. More specifically,

Table 3 Realistic effects

Perspective	
	
Near	Further
Occlusion	
	
Occluded	Non Occluded
Light	
	
Low Intensity	High Intensity
Shadows	
	
Collision	
	
Gravity	
	
Affected	Non Affected

the log information revealed that 121 different devices connected to the system during the experience. The average number of movement votes sent per device during the two games was 235.7.

6 Discussion

The games provided perspective, occlusion, light, shadows, collision and gravity effects so the grade of realism obtained in the experience was quite high and the integration between the elements of the virtual and real world was carried out seamlessly. Table 3 illustrates the way the effects implemented looked during the play with different screenshots. As shown in the pictures, as the characters moved along the auditorium corridors, they looked partially occluded by the audience stands and walls each time they passed behind some of them. They also looked smaller the further in the room they went, and they projected shades as they would have done if really being there. As previously mentioned, in this experience, we did not use effects of wind, semi-transparent surfaces or spatial sounds. In any case, these effects have been already implemented in the current version of AR GREP, and it would have been easy to integrate them into the AR scene if necessary. In fact, one of the main advantages of the approach is that once an initial static AR scene is prepared, it is very easy to add or remove the other effects to it. This could support the designer in testing different configurations for maximizing the realism of the scene, or adapting the ambient elements to the current atmosphere of the real setting.

With regard to the limitations of the approach, it is necessary to note that, as any other AR video see-through system, the realism of the experience can be compromised by any delay from capturing the video to displaying the AR image on screen. For this specific experience, and due to the large dimensions of the auditorium, we required to use an external camera that could record video at a wide angle. The transference of the data from this external device to the AR system caused a perceptible lag in the initial configuration of the system that can be eliminated using a USB capturer that does not compress the video signal. In addition, the use of the system is currently restricted to experiences in which the position of the camera in the real world is kept fixed. In order to maintain the alignment between the virtual and real elements, the viewpoint of the camera should be the same as the one used for building the model of the environment and setting up the position of the camera and background pane in the AR scene.

Although the goal of this paper is mainly focused on analyzing the feasibility of the approach and realistic degree of experience, it should be highlighted that the game was played by about 52.2% of the people. Unfortunately, it is not

possible to infer what age range had the greatest participation or getting user opinions. Therefore, evaluations focused on these purposes should be done in future experiments.

7 Conclusions and future work

We presented an approach for implementing realistic and interactive AR experiences that combines the use of model-based AR techniques with the benefits that current game engines offer for the design and management of 3D virtual environments. To support the evaluation of the feasibility of the proposed approach and the realism of the experiences delivered, we developed a prototype of an AR system that extended the functionalities of an existing game engine, the GREP system. The prototype was tested in an activity carried out in the auditorium of the University Carlos III of Madrid in which the audience interacted with the elements of an AR scene that implemented effects of occlusion, perspective, collision, lighting and shading. The audience seemed to genuinely enjoy the experience, and the virtual objects were successfully merged with the real ones. However, further evaluations would need to be carried out to gather more insight into the users' opinions and to compare the results with the ones obtained when using other AR techniques. Once the feasibility of the approach has been corroborated, the next step of our work is to develop an authoring tool to support non-expert users in the design and implementation of this type of experiences. The tool will be implemented as an extension of the GREP editors, expanding its current functionalities to support the modelling of environments, the definition of AR scenes and their enhancement with ambient effects. We believe that many different areas, such as education or cultural heritage, could greatly benefit if the experts in the field are empowered to design and implement realistic AR experiences by themselves. In addition, different solutions are currently being explored to allow using the proposed approach with a mobile camera, instead of one with a fixed position in the real world.

Acknowledgements This work is supported by the project CREAx and PACE funded by the Spanish Ministry of Economy, Industry and Competitiveness (TIN2014-56534-R and TIN2016-77690-R).

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. ATOMIC Authoring Tool download—SourceForge.net. https://sourceforge.net/projects/atomic-project/ (2008). Accessed 20 April 2017
2. Azuma, R.T.: A survey of augmented reality. Presence Teleoperators Virtual Environ. **6**(4), 355–385 (1997)
3. Berger, M.-O.: Resolving occlusion in augmented reality: a contour based approach without 3D reconstruction. In: 1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1997. Proceedings, pp. 91–96. IEEE (1997)
4. Breen, D.E., Rose, E., Whitaker, R.T.: Interactive occlusion and collision of real and virtual objects in augmented reality. Munich, Germany, European Computer Industry Research Center (1995)
5. Curless, B., Levoy, M.: A volumetric method for building complex models from range images. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, pp. 303–312. ACM (1996)
6. Davis, A., Chen, J.G., Durand, F.: Image-space modal bases for plausible manipulation of objects in video. ACM Tran. Graph. (TOG) **34**(6), 239 (2015)
7. Davison, A.J., Reid, I.D., Molton, N.D., Stasse, O.: Monoslam: real-time single camera slam. IEEE Trans. Pattern Anal. Mach. Intell. **29**(6), 1052–1067 (2007)
8. Du, H., Henry, P., Ren, X., Cheng, M., Goldman, D.B., Seitz, S.M., Fox, D.: Interactive 3D modeling of indoor environments with a consumer depth camera. In: Proceedings of the 13th International Conference on Ubiquitous Computing, pp. 75–84. ACM (2011)
9. Fischer, J., Bartz, D., Straber, W.: Stylized augmented reality for improved immersion. In: Virtual Reality, 2005. Proceedings. VR 2005. IEEE, pp. 195–202. IEEE (2005)
10. Fischer, J., Bartz, D., Straber, W.: Artistic reality: fast brush stroke stylization for augmented reality. In: Proceedings of the ACM Symposium on Virtual Reality Software and Technology, pp. 155–158. ACM (2005)
11. Fischer, J., Regenbrecht, H., Baratoff, G.: Detecting dynamic occlusion in front of static backgrounds for AR scenes. In: Proceedings of the Workshop on Virtual Environments 2003, pp. 153–161. ACM (2003)
12. Fortin, P.-A., Hebert, P.: Handling occlusions in real-time augmented reality: dealing with movable real and virtual objects. In: The 3rd Canadian Conference on Computer and Robot Vision, pp. 54–54. IEEE (2006)
13. Fuchs, H., Livingston, M.A., Raskar, R., Keller, K., Crawford, J.R., Rademacher, P., Drake, S.H., Meyer, A.A., et al.: Augmented reality visualization for laparoscopic surgery. In: International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 934–943. Springer (1998)
14. Fuhrmann, A., Hesina, G., Faure, F., Gervautz, M.: Occlusion in collaborative augmented environments. Comput. Graph. **23**(6), 809–819 (1999)
15. Haller, M., Drab, S., Hartmann, W.: A real-time shadow approach for an augmented reality application using shadow volumes. In: Proceedings of the ACM Symposium on Virtual Reality Software and Technology, pp. 56–65. ACM (2003)
16. Herdtweck, C., Wallraven, C.: Estimation of the horizon in photographed outdoor scenes by human and machine. PloS one **8**(12), e81462 (2013)
17. Ishii, H., Ben-Joseph, E., Underkoffler, J., Yeung, L., Chak, D., Kanji, Z., Piper, B.: Augmented urban planning workbench: overlaying drawings, physical models and digital simulation. In: Proceedings of the 1st International Symposium on Mixed and Augmented Reality, pp. 203. IEEE Computer Society (2002)

18. Kaufmann, H., Schmalstieg, D.: Mathematics and geometry education with collaborative augmented reality. Comput. Graph. **27**(3), 339–345 (2003)
19. Kolivand, H., Sunar, M.S.: Realistic real-time outdoor rendering in augmented reality. PloS one **9**(9), e108334 (2014)
20. Kolsch, M., Bane, R., Hollerer, T., Turk, M.: Multimodal interaction with a wearable augmented reality system. IEEE Comput. Graph. Appl. **26**(3), 62–71 (2006)
21. Kurz, D., Benhimane, S.: Gravity-aware handheld augmented reality. In: 2011 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 111–120. IEEE (2011)
22. Mendez, E., Schmalstieg, D.: Importance masks for revealing occluded objects in augmented reality. In: Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology, pp. 247–248. ACM (2009)
23. Piekarski, W., Thomas, B.: Arquake: the outdoor augmented reality gaming system. Commun. ACM **45**(1), 36–38 (2002)
24. Rolland, J.P., Holloway, R.L., Fuchs, H.: Comparison of optical and video see-through, head-mounted displays. In: Photonics for Industrial Applications, pp. 293–307. International Society for Optics and Photonics (1995)
25. Sodnik, J., Tomazic, S., Grasset, R., Duenser, A., Billinghurst, M.: Spatial sound localization in an augmented reality environment. In: Proceedings of the 18th Australia Conference on Computer-human Interaction: Design: Activities, Artefacts and Environments, pp. 111–118. ACM (2006)
26. Ulbricht, C., Schmalstieg, D.: Tangible augmented reality for computer games. na (2003)
27. Wang, Y., Samaras, D.: Estimation of multiple directional light sources for synthesis of augmented reality images. Graph. Models **65**(4), 185–205 (2003)
28. Wikitude—The Worlds leading Augmented Reality SDK. http://www.wikitude.com (2008). Accessed 20 April 2017
29. Zarraonandia, T., Diaz, P., Aedo, I.: Using combinatorial creativity to support end-user design of digital games. Multimed. Tools Appl. **76**(6), 9073–9098 (2017)
30. Zhou, Z., Cheok, A.D., Yang, X., Qiu, Y.: An experimental study on the role of software synthesized 3D sound in augmented reality environments. Interact. Comput. **16**(5), 989–1016 (2004)