Introducing Mokap: A Novel Approach to Creating Serious Games

Javier Torrente[1], Ángel Serrano-Laguna[2], Conor Fisk[3], Breid O'Brien[3], Wanda Aleksy[3], Baltasar Fernández Manjón[2], Patty Kostkova[1]

(1) University College London
Computer Science Department
Gower Street, London
{j.torrente, p.kostkova}@ucl.ac.uk

(2) Complutense University of Madrid
C Profesor Jose Garcia Santesmases
sn, 28040 Madrid
{angel.serrano@fdi.ucm.es, balta@fdi.ucm.es}

(3) University College London Hospitals
Euston Road, London
{Conor.Fisk, Breid.O'Brien, Wanda.Aleksy}@uclh.nhs.uk

ABSTRACT

In this paper we introduce *Mokap*, a novel authoring and development toolset for serious games. Mokap aims to address two specific challenges: provide support for early development stage design and facilitate involvement of domain experts. More specifically, on the one hand, it aims to make serious game authoring possible for people with no technical background. On the other hand, Mokap provides explicit support for activities related to analysis and design, especially useful during the early stages of development. In this sense, Mokap aims to facilitate the transition from the designers' board to the programmers' environment and also to enhance communication between the different roles of the development team. Developed as an open source environment, Mokap is oriented to mobile devices (tablets and smartphones), a kind of technology most educational stakeholders are familiar with. We present the main design concepts behind Mokap, along with some technical insight into the platform and the description of the game *Listen With Lemur* as a case study.

Categories and Subject Descriptors

D.1.7 [**Programming Techniques**]: Visual programming;
K.3.1 [**Computers and Education**]: Computer uses in education – *distance learning, computer-managed instruction*;
K.8.0 [**Personal Computing**]: General – *games*.

General Terms

Design, Human Factors.

Keywords

Mokap, authoring tools, serious games.

1. INTRODUCTION

Digital games have become a media of great economical and socio-cultural impact. Taking into account just the US market, the total revenue of the digital games sector has grown from $2.6 billion in 1996 (73.8 million units sold) to $11.7 billion in 2008 (almost 300 million units sold) according to the Entertainment Software Association [20]. Although the digital games sector has suffered a moderate recession in the last years [21] due to the economic crisis, it is still able to generate massive revenues. For example, *Grand Theft Auto V*, released in September 2013 by *Rockstar*, generated more than $1 billion in retail sales during its first three days on sale, being the digital entertainment asset to faster reach this milestone, including digital games and films [29]. Also, some authors have not hesitated to compare the digital games sector with fine arts such as painting, music, or movies [2, 19, 28]. Actually the boundary between digital games and art is more blurred every day. For example, the influence of digital games in movies is increasing, with a growing number of movies based on games such as *Prince of Persia*, *Street Fighter*, *Tomb Raider* or *Resident Evil*.

Furthermore, digital games have trespassed the entertainment field, reaching areas like health or education giving form to what is informally called *serious games* [37]. The term serious games refers to those digital games that have a purpose beyond recreation [25], like the aforementioned health [1, 5, 10, 23, 47], but also advertising [44], crowdsourced research [15] or awareness raising [6, 48, 53]. However, education is one of the most relevant fields for serious games, where they are proposed as an effective means to increase students self-involvement in their own learning [17, 32, 37] due to their ability to keep players focused and concentrated to complete complex tasks [16, 25], reaching states of optimal flow [11, 36]. This ultimately results in a more meaningful and long lasting learning, yielding sometimes a higher academic performance [14, 31, 33, 45, 52]. In this regard, the effectiveness of serious games when properly designed and implemented is beyond debate, as it has been widely backed up with recent experimental evidence [4, 7, 12, 30, 43, 51].

The current importance of digital games has also resulted in the growth and diversification of the activity of creating digital games. For this purpose, numerous approaches and tools are available. Nonetheless, development of digital games is always an effort-consuming activity that requires a delicate balance of multidisciplinary skills [8]. If the game has a serious/educational purpose, it is also necessary to leverage the playful and educational components of the game [40]. Overall, this makes serious game development expensive and hard to scale [7], which constrains the number of serious games available in the market. There is a need to bridge this gap with game development formulas that allow significant cost reductions [22] but without constraining the educational value [54]. One of the approaches is to bring game development closer to educators, allowing educational communities to fulfil their own needs for game-based content with a higher level of autonomy [18]. The importance of stakeholder involvement in the design of serious games has been demonstrated on a number of projects, for example the eBug [24] and edugames4all [38].

There are some authoring tools available that try to simplify serious game development and bring it closer to the educational community. This is the case of *eAdventure* (http://e-adventure.e-ucm.es). This kind of tools contrast with general purpose tools like *Adventure Game Studio* (http://www.adventuregamestudio.co.uk/) or *Unity (*http://unity3d.com/*)*, which are more complex and lack of explicit educational support as they allow to create games of different genres and with different degrees of complexity.

One of the main limitations of current game development environments is lack of explicit support for game design. Designed for programmers, game design and authoring tools are difficult to use for other key stakeholder groups frequently involved in serious game design such as educators, designers or content experts involved in serious game development.

In this paper we introduce *Mokap* (http://www.mokap.es), a new toolset for serious game development that aims to fill the two aforementioned gaps: make serious game development easy to use by the educational community and providing explicit support for activities related to analysis and design that are so important during the early stages of development, where productivity, flexibility and agility to capture, share and test new ideas is more necessary than having full functionality. In this sense, Mokap aims to facilitate the transition from the designers' board to the programmers' environment and also to enhance communication between the different roles of the development team. Mokap is oriented to mobile devices, like tablets and smartphones, as this type of technology allows the team to work anytime and anywhere. Tablets also add productivity in this context, as they provide specific features, like the camera or the seamless integration with different applications installed in the device, that are quite helpful to boost the creative processes associated to the early stages of game analysis and design.

2. EXISTING TOOLS FOR SERIOUS GAME DEVELOPMENT

The offer of tools and environments available for digital game development is varied in terms of factors like game genre, game purpose (serious/recreational), platform or budget available. Thus it is not our intention to make a deep analysis, but to provide a concise overview of the state of the art in game development environments, stressing current limitations, structured in two sub-sections. In 2.1 we present an overview of general-purpose game development tools, while in 2.2 we focus on tools oriented to education and serious games.

2.1 Professional Game Development Tools

We present professional game development tools in decreasing order of complexity and cost.

In this regard, AAA toolsets are on the top of this segment. These are the more complex and professional (and also expensive) toolsets available in the market, which are used mainly to develop AAA games, a term used to refer to games with the highest development budgets and levels of promotion. These software packages provide a fully-featured game engine powered with plenty of functionality (artificial intelligence, physics engine, animations, etc.). Different advanced tools are built on top of that engine, optimized for particular tasks usually carried out by specialized professionals, like sprites and animation editors, level design editors or map design tools. This makes AAA toolsets the perfect choice for large development and skilled teams.

CryEngine (http://cryengine.com/) and *Unreal Engine* (https://www.unrealengine.com/what-is-unreal-engine-4) are two outstanding examples of AAA toolsets. Developed by *Crytek* and *Epic Games* respectively, they were originally created for two successful AAA games (*Unreal Tournament* and *FarCry*) and then made available for sell to other game development studios.

There is also a segment of professional game authoring tools that, although still very powerful in functionality, are much less complex than AAA toolsets. Their functionality is usually provided through an integrated authoring and programming tool. Their excellent balance between complexity and functionality plus their attractive cost makes them the perfect choice for mid-size and small professional teams, and also for amateur development. The most representative example of this type of tool is *Unity* (http://unity3d.com/). It comes equipped with all the functionality to create cross-platform games (PC, Mac, Web, mobile and console platforms supported) of different types and levels of complexity.

The lower segment of professional game tools is filled by game programming libraries, frameworks and engines. They can be commercial or free, with multiple open-source options. They all have in common that they are mostly oriented to programmers, and therefore they are most focused on the coding part of the games than high-level authoring. Some examples are *LibGdx* (http://libgdx.badlogicgames.com/) or *Ogre3D* (htt://www.ogre3d.org/).

All these tools have in common some limitations. First of all, they are oriented to professionals and therefore are rather inaccessible to people without a strong background in computer science and game programming.

Second, they are designed to meet the implementation needs of the team and tend to be ineffective in supporting the creative processes related to game concept and design, which are present throughout all the development but especially in early stages. These processes, which are indispensable for success, are usually lead by people with creativity and design backgrounds but limited technical knowledge. Subsequently, most game design activities are still conducted using traditional non-digital tools (e.g. pen-and-pencil), and their main outcome is usually a game design specification document that is then passed to the programmers. This is the main instrument teammates with different backgrounds (e.g. designers and programmers) have to articulate their communication [35]. The communication problem is even worse in serious game development, where domain experts have also to get engaged.

The lack of explicit support for game design slows down development, since it is harder to transition from the design table to the programming environment. Besides, it hinders rapid prototyping and agile development, as it takes longer to accommodate any changes on the proposed design to a working prototype. It seems necessary, therefore, to explore the creation of new tools that also support game design, that are flexible and responsive to change, and where people with different backgrounds (domain experts, programmers, designers, etc.) can provide input.

2.2 Specialized tools

Moving away from professional tools, there are solutions more specifically oriented to education. These tools sacrifice functionality to simplify the game creation process so they can be used by a less technical public. They allow highly motivated teachers and students to create simple games with a high educational value. One of the first examples is *GameMaker*, developed by Mark Overmars [42]. *Scratch* is another popular tool that tries to teach programming to kids through digital game creation [46], which also has a tablet-oriented variant recently published for the iPad (http://www.scratchjr.org/).

Compared to professional tools, education-oriented tools mean an improvement in terms of cost reduction and involvement of domain experts. But, although they simplify serious game development, the effort needed to create them is still too high for educational standards. There are several factors that these tools usually miss out and that result in lower productivity. For example, these tools do not usually provide assets that game authors can put directly into the game, like characters, animations, or game scenarios. As a consequence, it is necessary for their users to have skills for gathering, adapting and/or creating artwork resources. Another factor is the integration of programming concepts into the authoring tools. Even if no coding is strictly required with these tools, users still need to learn and be able to apply concepts like 'variable', 'if' or 'loop', which are complex. It is necessary, therefore, to go one step further in the simplification of serious game creation to bring it even closer to the educational community. Finally, most of these tools are designed for desktop environments, failing to reflect the current shift towards mobile computing [13]. We think that to reach the educational community it is necessary to bring serious game authoring to tablets and smartphones, as teachers, students and parents are now more familiar with them than with PCs.

2.2.1 eAdventure

In this category the *eAdventure* (http://e-adventure.e-ucm.es) tool deserves especial attention, as it is the precursor of the Mokap project. eAdventure was the result of years of research in serious game authoring at the e-UCM research group of the Complutense University of Madrid. eAdventure helps instructors to develop their own serious games, or at least get involved and directly contribute in their production by following two strategies. First, it is supported by a game design and development methodology called EGDA [49] that facilitates collaboration of professionals from different backgrounds (e.g. game programmers and domain experts). Secondly, eAdventure provides an educator-oriented game authoring tool that does not involve programming and optionally a storyboard visual language that facilitates design and communication [50].

eAdventure is focused on the point-and-click adventure game genre (in the style of games such as *Monkey Island* or *Myst*), which is a genre that has traits that make them more suitable for educational purposes than others [3], like their strong problem-solving underpinnings and general predominance of reflection over action [17, 27]. eAdventure has been successfully applied in developing different serious games and game-like simulations, especially in the medical domain. For example, it has been used to train medicine students in Hematology [39] and Central Venous Catheter insertion procedures [41], to educate professionals in deceased organ management [9] and to teach high schoolers basic cardiopulmonary resuscitation [34].

3. MOKAP: A NEW APPROACH TO SERIOUS GAMING

Mokap is the response of the e-UCM group to the challenges serious game development faces. It aims to facilitate game development and bring it closer to the educational community. Mokap has been completely designed from scratch to provide the user a seamless, modern, pleasant and social game creation experience that integrates state-of-the-art design principles.

3.1 A Modern Game-Creation Experience

Mokap tries to adapt serious game creation to current technological trends and make it accessible to anyone.

In first place, Mokap is optimized for a mobile experience. The preferred platform is a 10" tablet, although it can also be used in 7" tablets and smartphones of any size. This helps bring game authoring to platforms that are more natural for students and teachers.

Secondly, Mokap tries to deliver a user experience that is as pleasant and rewarding as users are accustomed to have in their mobile devices. As with any other type of mobile app, to reach a wide audience it is necessary to integrate current best practices in user interface design. In this sense, Mokap embodies good application of Google's Material Design principles (http://www.google.com/design/spec/material-design/introduction.html), which are now considered the reference for successful mobile development, especially in Android. In this way, Mokap's user interface is bold, graphic and intentional to get the user immersed in the experience of creating serious games. Mokap uses motion and animation throughout all its user interface, but not only to please the eye but fundamentally to convey meaning, drive user attention and help the user understand what is going on at any moment with a low cognitive load. All the interface is designed to be responsive and foster user self-discovery of the app through exploration.

Thirdly, Mokap views serious game authoring as a social experience. This social component is part of the core of its user experience. With Mokap users can easily share their content with others (experts, students, etc.) and also establish collaborations to create new content, or explore new content created by the community right from the app.

3.2 Mokap Community

The Mokap Community is still a work in progress (Figure 1). When finished, it will be an online place for Mokap users to download pieces of content directly onto their game projects. Users will also be able to share games or parts of them so other users can reuse them. But the community is not only fed by user-generated content, as the Mokap team is also making an effort to push new content regularly to boost the development of serious games by Mokap users.

The kind of content that is available on the community is varied. There are animated characters, for example, already set up and ready to be used. But there are also pre-set menus and other configurable artifacts to be added into the games. Some of the content is provided 'as is', but most of it will support customization.

3.3 Main Functionalities

From a high-level user perspective, Mokap provides several interesting functionalities for creating games. First, it allows composing game scenes by combining elements from different sources. For example, these elements can be gathered from the device image gallery, or they can be captured as photos or videos, taking advantage in this way of the unique features that mobile technology provides. Mokap also supports drawing with the finger or a stylus, which is a great feature to support creative mental processes when designing the games, and for rapid prototyping. Finally, Mokap allows importing rich assets from the community, so users do not have to deal with multimedia edition and adaptation, which is usually not a skillset most teachers and students have.

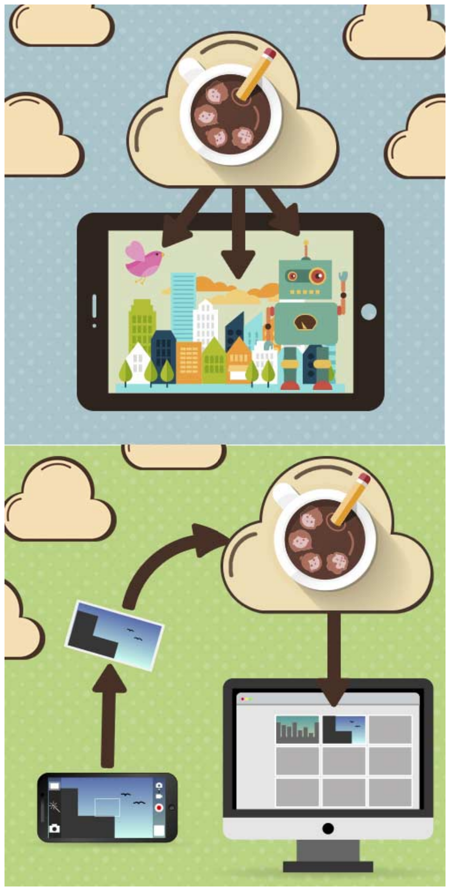


Figure 1. Diagram of the Mokap Community serving content pieces to a Mokap client application (top) and also receiving a game created by a user to be shared with other users (bottom).

Once game scenes have been composed, Mokap allows to easily get them connected to compose a navigational game environment. Scenes can also be enhanced with ambience effects (e.g. snow, rain, sun effects) and interactivity can be added to any game element. Users can configure how the game should react to any user interaction by combining effects from a wide list. These effects include animation (e.g. make elements move, rotate, scale, etc.), change elements' visibility, add new elements to the scene, change variable values, or get sounds played, among others.

Mokap also integrates with other popular apps available on Google Play to create a richer game creation environment. This is the case of Pixlr Express (© Autodesk Inc.), an image edition tool that lets users do basic image editing right on their mobile device and also add amazing effects with no effort. Finally, Mokap allows publishing and sharing games via Google Drive.

4. THE MOKAP SOFTWARE ECOSYSTEM

In this section we further describe Mokap from a technical perspective.

4.1 Architecture

Mokap is more than an authoring tool. It runs on top of a powerful software architecture that provides a lot of functionality (Figure 2). The Mokap authoring tool is just one of the upper layers.

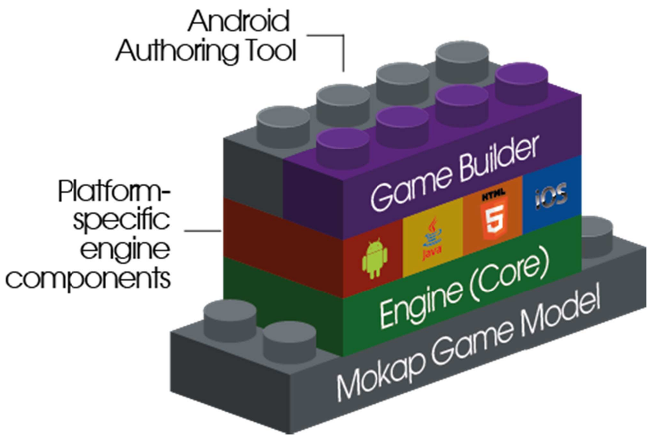


Figure 2. Mokap software architecture.

The base layer of this architecture is the Mokap Game Model (section 4.2) that fully describes the contents of the game. The Mokap engine (section 4.3) is responsible for parsing and interpreting the game definition. Finally, there are two upper layers built on top of the game engine and the game model, which are meant as utilities to build games. The first is the Android Authoring Tool (section 4.4), which is a high-level authoring tool oriented to people with no technical background nor programming knowledge. The second is the Game Builder (section 4.5), which is a software library that provides an API to help programmers write Mokap games.

4.2 Mokap Game Model

The Mokap Game Model is a JSON-based (http://www.json.org/) specification that fully describes any Mokap game. All the game elements and scenarios are internally described using this game model. For a Mokap game to be considered valid, it must fully comply with this model. Otherwise the engine will not be able to interpret it.

The Mokap Game Model is designed following an *entity-component* architecture [26]. In this context, an entity is just any element that can be found in the game. In this way, scenes, static elements, menus, controls, and interactive elements are all considered entities. Entities can also contain other entities (children), allowing for a hierarchical and structured way to represent games. The functionality of each entity is determined by the number and type of components it has (note that in this context a component has a different meaning than in classic software design). This allows changing the behaviour of any entity in a very flexible manner, as components can be added or removed at runtime as needed.

Figure 3 shows an example of an entity in Mokap with three components. The first component is an image, which defines the visual representation of the entity when displayed in the game. The second component adds animation to the entity, which will oscillate on the screen. Finally, the third component defines that the entity can be 'touched' by the user, and it specifies how the game has to react to that interaction through two effects: go to a new scene and change a global variable.

```
{
    x: 292,
    y: 348,
    components: [{
        class: image,
        uri: 02-04-Banana.png
    },
    {
        class: rotatetween,
        delay: 0.1,
        repeat: -1,
        yoyo: true,
        duration: 1.5,
        relative: true,
        easeEquation: sine,
        rotation: 20
    },
    {
        class: behavior,
        event: {
            class: touch
        },
        effects: [{
            class: goscene,
            sceneId: scenes/s3.json,
            duration: 0.3,
            transition: fadeIn
        },
        {
            class: changevar,
            variable: levelSelected,
            expression: i1,
            context: global
        }]
    }]
}
```

Figure 3. Top: Example of a Mokap game entity with three components. Botton: Visualization of the entity in the Mokap engine.

4.3 The Mokap Game Engine

The Mokap Engine is the software component that parses, interprets and executes the games. In essence it follows a classic game engine structure, where the logic is organized in a loop where first input are read, then the world is updated and finally rendered.

Most of the logic related to the update() phase in the loop is carried out by *systems*. In this context, a system is a piece of logic that knows how to deal with a particular type of component. This allows for a flexible and scalable way to structure the logic of the game engine.

For example, the entity described in Figure 3, will be processed by three different systems. One system is responsible for processing the *image* component, which results in loading the associated image and get it displayed on the game world. Another system is responsible for processing the *rotatetween* component, which controls the interpolation that it defines (changes the entity's rotation on screen according to a function specified). Finally, a third system is responsible for dealing with any touches performed over the entity, and launching the corresponding effects as specified.

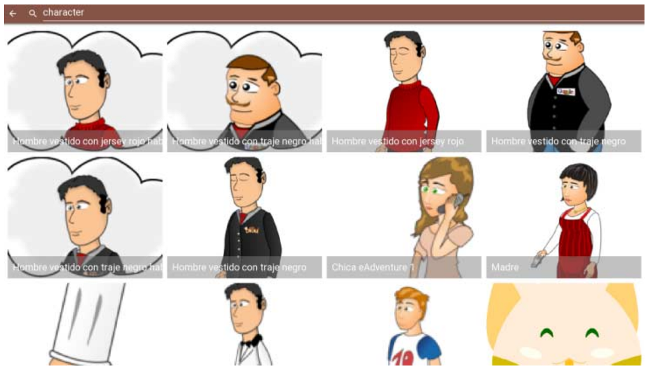

character
Hombre vestido con jersey rojo
Hombre vestido con traje negro
Hombre vestido con jersey rojo
Hombre vestido con traje negro
Hombre vestido con traje negro
Hombre vestido con traje negro
Chica eAdventure
Madre

Figure 4. Snapshot of the Mokap Android Authoring Tool. Image shows a search performed on the Mokap community backend to retrieve available characters that can be integrated into the game.

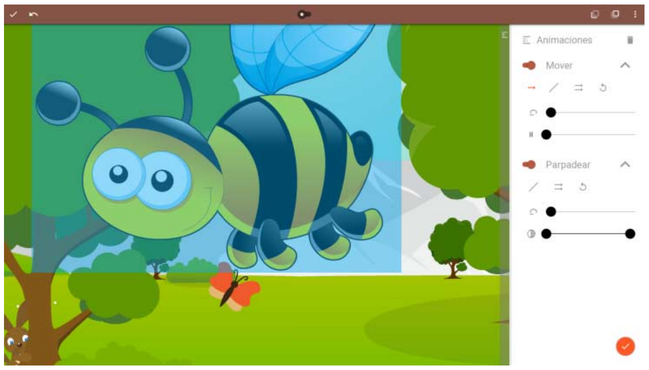
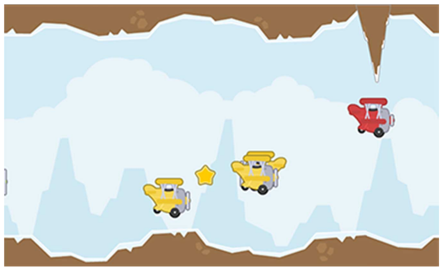


Figure 5. Snapshot of the Mokap Android Authoring Tool. Image shows configuration of animations of a game element (a bee).

Another important aspect of the Mokap engine is that it is designed to run on a multiplatform environment. This allows cross platform distribution of the games, which is essential to facilitate deployment in educational settings. This way, the engine is divided in several software modules (Figure 2). The first is the engine core, which provides platform-agnostic basic functionality. This is built upon the LibGDX (http://libgdx.badlogicgames.com/) game development library which runs on OpenGL2ES directly. On top of the engine core, different modules provide platform-specific functionality, like access to the file system. This allows deploying the games on the Web (using HTML5), desktops (as Java standalone applications), and Android devices (as native apps). In the future this will also allow running the games on iOS mobile devices.



```
@Override
protected void doBuild() {
    /* Build game with one scene with background
    image */
    ModelEntity firstScene =
        singleSceneGame("background.jpg").
        centerOrigin().getLastScene();

    /* Add parallax component to background so it is
    always kept centered on screen */
    parallax(firstScene.getChildren().get(0), 0F);
    /* Create up and down rocks as children of the
    main scene entity. Get them aligned
    automatically. */
    entity(firstScene, "rocks-up.png",
          VerticalAlign.DOWN, HorizontalAlign.LEFT).
          centerOrigin().parallax(0.5f);
    entity(firstScene, "rocks-down.png",
           VerticalAlign.UP, HorizontalAlign.LEFT).
           centerOrigin().parallax(0.5f);
    /* Add init behavior to the scene with two
    effects: 1) Register var that points to plane
    being tracked 2) Add component to the camera for
    chasing the plane being tracked */
    initBehavior(firstScene);
    changeVar(chasedPlaneVar, newestEntityRef,
              ChangeVar.Context.GLOBAL);
    addComponent("(layer scamera)",
                 makeCameraTracking());

    /* Create script running all the time that
    creates a plane per second*/
    infiniteTimer(firstScene, 1,
                  makeAddPlaneToSceneScript());

    /* Create the star that marks the plane that is
    being tracked by the camera (chases the tracked
    plane)*/
    entity(firstScene, "star.png", 20, 400).
          tags("star").centerOrigin();
}
```

Figure 6. Snapshot of a simple plane game developed with the Mokap Game Builder (top) and a snippet of the code that builds it (bottom).

4.4 The Android Authoring Tool

The Android authoring tool is the most visible software product related to Mokap. It is a user-friendly, WYSIWYG (what-you-see-is-what-you-get) editor optimized for tablets but that runs well also on smartphones. It is designed for people with no experience in serious game development, and thus its use is quite straightforward.

The Mokap Android Authoring tool is built directly on top of the engine. This way, the user is always getting accurate feedback on what the game is going to look like when students play it. It also allows switching between edition and play modes to efficiently support an agile edition-testing cycle.

Figures 4 and 5 show two snapshots of the Mokap Android Authoring Tool.

4.5 The Game Builder

The Game Builder is a software module specifically oriented to programmers. This allows professionals with plenty of expertise to get full advantage of all the functionality provided by the game engine. It provides a Java API that allows fast game development.

The Game Builder is focused on productivity. It provides game developers with different pieces of functionality and solutions for recurrent actions. First, it provides an API for building the Game Model. Second, it provides convenient functionality for efficient managing art resources and imagery, like image analysis and automatic optimization, easy layout, etc. Once the game is built, the Mokap Game Builder allows packaging the game as a standalone app for any of the supported platforms.

Figure 6 shows a simple planes game developed using the Game Builder. In the game, every second a plane of a random colour is added to the scene at a random position. It moves along a horizontal axis with also a random speed. At any time the player can click (or touch) any of the planes, and the camera will start tracking it. A little star marks the plane that is actually being tracked. The figure shows the game and also a snippet of code showing how it was built with Game Builder (the code shown accounts for around 60% of the total code of the game).

5. CASE STUDY: LISTEN WITH LEMUR

In this section we describe as a case study the game *Listen With Lemur*, developed by University College Hospital London using Mokap technology.

5.1 A Game to Train Hearing in Kids with Cochlear Implants

Listen With Lemur (see Figures 7 and 8) is a simple game for little children (18-24 months old) who have recently received a cochlear implant. A cochlear implant is a small device that is surgically placed in the ear of people that are severely hard-of-hearing or deaf that gives them a sense of hearing by transforming sound into electric signals (http://www.nidcd.nih.gov/health/hearing/pages/coch.aspx). As these kids have never heard before in their lives, they need to develop basic skills to start recognizing sounds. The main purpose of the game is to help them train those skills.

The game has a total of eleven scenarios spreading more than 60 animated animals and characters. Two of these scenarios are meant to help the kids train their hearing skills while most of the others are designed to evaluate their progress.

5.2 Development and Cost Estimation

Listen With Lemur has been developed using the Mokap architecture. The Mokap Android Authoring Tool was used to quickly transform the design specs provided by the domain experts into a working prototype. Then the Game Builder was used to build a fully functional, high end and detailed version of the game. The project has taken around 130 hours of work to implement, including development, testing and artwork. For that, a total of 9,000 lines of code were needed. These figures are quite low comparing to current industry standards, which shows the efficacy of Mokap to help agile development of high-end serious games with reduced costs.

Figure 7. Welcome screen of the game *Listen With Lemur*, developed with Mokap.

Figure 8. Scene of the *Listen With Lemur* game where kids can explore sounds by interacting with the animals partially hidden in the pond.

6. FINAL REMARKS

In this paper we have introduced Mokap, a novel software ecosystem for developing serious games providing both a high level authoring tool (oriented to authors with no strong technical background) and also a low-level game programming library (oriented to game programmers). Mokap aims to help address two of the challenges related to serious game development: (1) facilitate involvement of non-technical stakeholders, like educators or designers, and help articulate communication between them and game developers by providing a common tool that all stakeholders can use; and (2) support early game design where flexibility and creativity are more important than full functionality.

7. ACKNOWLEDGMENTS

The e-UCM research group has been partially funded by Regional Government of Madrid (eMadrid S2013/ICE-2715), by the Ministry of Education (TIN2013-46149-C2-1-R) and by the European Commission (RAGE H2020-ICT-2014-1-644187).

Our more sincere thanks to all the team involved in the development of the Mokap platform: Dan Cristian Rotaru, Antonio Calvo and Manuel Freire, and the game *Listen With Lemur*, without whom this would not have been possible: Julie Hare (Consultant Speech and Language Therapist), Liz Cullen (Specialist Speech and Language Therapist), Caroline Munro (Specialist Teacher of the Deaf), Alexandra Wheeler (Specialist Teacher of the Deaf) and Marcos García – Ergüín (Lemur's graphic designer).

8. REFERENCES

[1] Akl, E.A. et al. 2013. Educational games for health professionals. *The Cochrane database of systematic reviews*. 3, 3 (Jan. 2013), CD006411.

[2] Aldrich, C. 2003. *Simulations and the Future of Learning: An Innovative (and Perhaps Revolutionary) Approach to e-Learning*. Jossey-Bass Inc.

[3] Amory, A. et al. 1999. The Use of Computer Games as an Educational Tool: Identification of Appropriate Game Types and Game Elements. *British Journal of Educational Technology*. 30, 4 (1999), 311–321.

[4] Annetta, L.A. et al. 2009. Investigating the impact of video games on high school students' engagement and learning about genetics. *Computers & Education*. 53, (2009), 74–85.

[5] Arnab, S. et al. 2012. *Serious Games for Healthcare: Applications and Implications*. IGI Global.

[6] Bang, M. et al. 2006. The powerhouse: A persuasive computer game designed to raise awareness of domestic energy consumption. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 3962 LNCS, (2006), 123–132.

[7] Barzilai, S. and Blau, I. 2013. Scaffolding Game-Based Learning: Impact on Learning Achievements, Perceived Learning, and Game Experiences. *Computers & Education*. In Press, (Aug. 2013).

[8] Blow, J. 2004. Game Development: Harder than you think. *Queue*. 1, 10 (Feb. 2004), 28.

[9] Borro-Escribano, B. et al. 2013. Developing game-like simulations to formalize tacit procedural knowledge: the ONT experience. *Educational Technology Research and Development*. (Nov. 2013).

[10] Brox, E. et al. 2011. Healthy Gaming - Video Game Design to promote Health. *Applied clinical informatics*. 2, 2 (Jan. 2011), 128–42.

[11] Chen, J. 2007. Flow in games (and everything else). *Communications of the ACM*. 50, 4 (2007), 31–34.

[12] Cheng, M.-T. et al. 2013. An educational game for learning human immunology: What do students learn and how do they perceive? *British Journal of Educational Technology*. (2013), n/a–n/a.

[13] Cisco 2014. Cisco Visual Networking Index : Forecast and Methodology , 2013 – 2018. *Middle East*. June, (2014), 2013–2018.

[14] Connolly, T.M. et al. 2012. A systematic literature review of empirical evidence on computer games and serious games. *Computers & Education*. 59, 2 (2012), 661–686.

[15] Cooper, S. et al. 2010. Predicting protein structures with a multiplayer online game. *Nature*. 466, 7307 (Aug. 2010), 756–60.

[16] Dede, C. 2009. Immersive Interfaces for Engagement and Learning. *Science Magazine*. 323, 5910 (2009), 66–69.

[17] Dickey, M.D. 2005. Engaging by design: How engagement strategies in popular computer and video games can inform

instructional design. *Educational Technology Research and Development*. 53, 2 (2005), 67–83.

[18] Eastwood, J.L. and Sadler, T.D. 2013. Teachers' Implementation of a Game-Based Biotechnology Curriculum. *Computers & Education*. (Feb. 2013).

[19] Van Eck, R. 2007. Building Artificially Intelligent Learning Games. D. Gibson et al., eds. Information Science Publishing.

[20] ESA, E.S.A. 2009. *Essential Facts about the Computer and Videogame Industry*. Entertainment Software Association.

[21] ESA, E.S.A. 2014. *Essential facts about the computer and videogame industry*.

[22] F A S 2006. *Summit on Educational Games: Harnessing the power of video games for learning*.

[23] Farrell, D. et al. 2011. Computer games to teach hygiene: An evaluation of the e-Bug junior game. *Journal of Antimicrobial Chemotherapy*. 66, (2011), 39–44.

[24] Farrell, D. et al. 2011. Developing e-Bug web games to teach microbiology. *Journal of Antimicrobial Chemotherapy*. 66, (2011).

[25] De Freitas, S. 2006. Learning in Immersive Worlds: A review of game-based learning. JISC e-Learning Programme.

[26] Garcia, F.E. and de Almeida Neris, V.P. 2014. A Data-Driven Entity-Component Approach to Develop Universally Accessible Games. *Universal Access in Human-Computer Interaction. Universal Access to Information and Knowledge SE - 49*. C. Stephanidis and M. Antona, eds. Springer International Publishing. 537–548.

[27] Garris, R. et al. 2002. Games, Motivation and Learning: A Research and Practice Model. *Simulation & Gaming*. 33, 4 (2002), 441–467.

[28] Gee, J.P. 2003. *What video games have to teach us about learning and literacy*. Palgrave Macmillan.

[29] GTA 5 retail sales pass $1 billion in three days: 2013. *http://www.computerandvideogames.com/430331/gta-5-retail-sales-pass-1-billion-in-three-days*. Accessed: 2015-01-26.

[30] Hainey, T. et al. 2013. Students' attitudes toward playing games and using games in education: Comparing Scotland and the Netherlands. *Computers & Education*. In Press, (Aug. 2013).

[31] Hwang, G.-J. and Wu, P.-H. 2012. Advancements and trends in digital game-based learning research: a review of publications in selected journals from 2001 to 2010. *British Journal of Educational Technology*. 43, 1 (Jan. 2012), E6–E10.

[32] Kirriemur, J. and McFarlane, A. 2004. *Literature review in games and learning*.

[33] Lazareck, L.J. et al. 2010. Learning by gaming - Evaluation of an online game for children. *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC'10*. (2010), 2951–2954.

[34] Marchiori, E. et al. 2012. Video-game instruction in basic life support maneuvers. *Emergencias*. 24, (2012), 433–437.

[35] Marchiori, E.J. et al. A narrative metaphor to facilitate educational game authoring. *Computers & Education*.

[36] Mayo, M.J. 2009. Video Games: A Route to Large-Scale STEM Education? *Science*. 323, 5910 (Jan. 2009), 79–82.

[37] Michael, D. and Chen, S. 2006. *Serious Games: Games that Educate, Train, and Inform*. Thomson.

[38] Molnar, A. et al. 2012. Who poisoned hugh? - The STAR framework: Integrating learning objectives with storytelling. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 7648 LNCS, (2012), 60–71.

[39] Moreno-Ger, P. et al. 2010. Application of a low-cost web-based simulation to improve students' practical skills in medical education. *International Journal of Medical Informatics*. 79, 6 (Jun. 2010), 459–67.

[40] Moreno-Ger, P. et al. 2008. Educational Game Design for Online Education. *Computers in Human Behavior*. 24, 6 (2008), 2530–2540.

[41] Moreno-Ger, P. et al. 2008. Online Learning and Clinical Procedures: Rapid Development and Effective Deployment of Game-Like Interactive Simulations. *Transactions on Edutainment I*. LNCS 5080, (2008), 288–304.

[42] Overmars, M. 2004. Teaching Computer Science through Game Design. *Computer*.

[43] Papastergiou, M. 2009. Exploring the potential of computer and video games for health and physical education: A literature review. *Computers & Education*. 53, 3 (Nov. 2009), 603–622.

[44] Pempek, T.A. and Calvert, S.L. 2009. Tipping the Balance: Use of Advergames to Promote Consumption of Nutritious Foods and Beverages by Low-Income African American Children. *Arch Pediatr Adolesc Med*. 163, 7 (2009), 633–637.

[45] Perrotta, C. et al. 2013. *Game-based learning : Latest evidence and future directions*. (NFER Research Programme: Innovation in Education). Slough: NFER.

[46] Resnick, M. et al. 2009. Scratch : Programming for all. *Communications of the ACM*. 52, 11 (2009), 60–67.

[47] Rosser, J.C. et al. 2007. The impact of video games on training surgeons in the 21st century. *Archives of surgery*. 142, 2 (Feb. 2007), 181–186.

[48] Swain, C. 2007. Designing Games to Effect Social Change. *Digital Games Research Association (DiGRA)* (2007), 805–809.

[49] Torrente, J. et al. 2014. Development of Game-Like Simulations for Procedural Knowledge in Healthcare Education. *IEEE Transactions on Learning Technologies*. 7, 1 (Dec. 2014).

[50] Torrente, J. et al. 2008. Instructor-oriented Authoring Tools for Educational Videogames. *8th IEEE International Conference on Advanced Learning Technologies (ICALT'08)* (Santander, Spain, 2008), 516–518.

[51] Tuzun, H. et al. 2009. The effects of computer games on primary school students' achievement and motivation in geography learning. *Computers & Education*. 52, 1 (Jan. 2009), 68–77.

[52] Wilson, K.A. et al. 2009. Relationships Between Game Attributes and Learning Outcomes: Review and Research Proposals. *Simulation & Gaming*. 40, (2009), 217–266.

[53] Wright, V.H. et al. 2009. Cyberbullying. *Journal of Computing in Teacher Education*. 26, 1 (Sep. 2009), 35–42.

[54] Young, M.F. et al. 2012. Our Princess Is in Another Castle: A Review of Trends in Serious Gaming for Education. *Review of Educational Research*.