AGORAS: TOWARDS COLLABORATIVE GAME-BASED LEARNING EXPERIENCES ON SURFACES

Alejandro Catala, Fernando Garcia-Sanjuan, Patricia Pons, Javier Jaen and Jose A. Mocholi
Grupo ISSI, Departamento de Sistemas Informáticos y Computación (DSIC), Universitat Politècnica de València
Camino de Vera s/n, 46022 Valencia, Spain

ABSTRACT

Children nowadays consume and manage lots of interactive digital software. This makes it more interesting and powerful to use digital technologies and videogames supporting learning experiences. However, in general, current digital proposals lack of in-situ social interaction supporting natural exchange and discussion of ideas in the course of creative learning activities. Moreover, learning activities based on games should put the emphasis on the phase of creating artifacts for the game ecosystem, not on just playing or consuming contents. With the aim of supporting learning activities focused on "playing to create games", this paper presents the implemented middleware to support this process from a technical point of view. The system allows the creation on an interactive tabletop of both single and advanced entities composed of shapes that can be enacted according to physics principles and rule-based behavior to create game ecosystems for learning purposes.

KEYWORDS

Game, Learning, Tabletop. Collaboration, Entity, Middleware.

1. INTRODUCTION

Nowadays children and teenagers have access to lots of interactive digital media and software in many different forms by means of several types of devices. This continuous contact with technologies contributes to digital literacy and makes them advanced citizens in the digital age. As a consequence, these users will be the base of skilled users in the future who will be able to deal with more complex and advanced software than ever.

Digital media and software should be used as a help supporting learning in combination with traditional materials rather than using the software as an exclusive mean for learning, as reported by McFarlane (McFarlane, 2002) or as the pedagogical model elaborated in the work by Gros for formal learning settings (Gros, 2007). In particular, these authors involved commercial videogames taking advantages of the knowledge that students already have on technologies as digital natives. Videogames are useful in this respect because they also maintain both motivation and engagement usually high (Michael, 2005), which is important for learning activities. In addition, they explain that using commercial video-games instead of software to conduct specific tasks is advantageous for several reasons: they are cheaper than designing and ordering the development of specific digital serious games; and in spite of the low customization capabilities of the commercial video-games, given that the pedagogical model can slightly change from one teacher to another and from one issue to other, the wider range of commercial video-games available gives flexibility in the purpose of enriching the learning setting. All this means that teachers demand platforms that permit activity customization for the development of learning sessions rather than a video-game to support already pre-established learning activities.

This vision is not essentially new, although videogames had not been applied in a generic way for learning purposes until these studies in real classroom settings. Before the digital age, Clark Abt already wrote about *serious games* as tools for learning and the way they should be used (Abt, 1970). Basically, he considered that games are useful tools for learning when they are played and consumed. However his major contribution is the idea that they are more powerful when teachers and tutors consider *playing to create games* as the primary learning activity and involve students in the process. In his opinion it is therefore more

rewarding from a learning point of view because this vision entails design tasks, making up rules, and producing the materials and the logics behind the game to be created co-operatively rather than just understanding pre-established game rules.

The perspectives illustrated above suggest that it will be more useful for teachers to have flexible digital tools available at hand that may be integrated in their respective pedagogical models for formal instruction; and that creative and social processes as the ones considered before the digital age along with higher motivation provided by the use of technology are of special interest for learning. Thus, our future long-term purpose is to provide a framework with these characteristics to be used in learning activities. Teachers would be able to create incomplete game ecosystems and design several creative learning exercises around them. These exercises would be addressed by the students in small groups, interacting and collaborating around the table. The creation of artifacts to complete the ecosystems would support the kind of authorship proposed by Abt, whereas the resulting game ecosystem could be enacted to test whether it works as expected, triggering reflection and discussion processes for refining the created artifacts.

The main contribution of this paper is the middleware that has been implemented to support such a framework from a technical point of view. This middleware allows the creation and enactment of virtual ecosystems on interactive tabletops composed of 2D entities. An example about a vintage game ecosystem is presented to illustrate and test the basic middleware functionality. On the implemented infrastructure several learning activities can be prepared, ranging from storytelling performances to programming physics simulation.

This paper is organized as follow. Section 2 describes some related work, focusing on those supporting some kind of artifact creation by children or teenagers. Section 3 introduces the entity-based model to support the construction of game ecosystems. Section 4 describes the middleware architecture implemented to support the enactment of game ecosystems. Section 5 describes one of the sample ecosystems that has been created to test the implemented middleware. Section 6 explains the associated rationale to use the platform for supporting learning activities. Finally, this paper concludes by summarizing the contribution of this paper.

2. RELATED WORK

Although there are a variety of proposals presenting software systems to create game ecosystems, in this section we are interested in works that are more focused on the creation of some kind of entities for learning activities. LogoBlocks is a graphical programming language to support programming for the LEGO programmable brick (Begel, 1996). The language uses a drag&drop methaphor in a Windows-Icons-Mouse-Pointers (WIMP) user interface. The *brick* is a small computer that can be embedded and used in LEGO creations by reading from sensors and controlling engine activations. In this way, children and teenagers can create ecosystems where robots are physical programmable entities.

Scratch is a graphical programming environment that allows children to program interactive stories, games and animations composed of a set of sprite-based objects (Maloney, 2004). The programming language to specify behavior is also based on a drag&drop metaphor of virtual blocks representing instructions. The main screen of the tool shows the stage and the sprite representing the entities, allowing program debugging, and testing new ideas increasingly and iteratively. The environment is a single-user application based on WIMP interaction.

Another interesting work is Agentsheets (Repenning, 2000). It is a tool based on agents that allows users to create simulations and interactive games. Users can create simulations of sprite-based agents in a 2D world arranged in a rectangular array. The users are responsible for designing the visual aspect of agents by drawing icons, so that these agents are actually sprite-based entities. Their behavior is based on event-based rules which are edited following a visual approach to the rewriting rule paradigm.

Topobo is a 3D constructive assembly system that allows the creation of biomorphic forms like animals and skeletons (Parkes, 2008). This is achieved by means of pieces embedded with kinetic memory. Topobo is designed to be a user interface that encourages creativity, discovery and learning through active experimentation with the system. It can help students to learn about several educational concepts on physics such as balance, center of mass, coordination and relative motion.

Finally, another interesting work is ShadowStory (Lu, 2011). It is a storytelling system inspired in Chinese traditional shadow puppetry. Children use a Tablet PC to create digital animated characters, and then they are allowed to perform stories on a back-illuminated screen, controlling the characters with simple movements by means of orientation handheld sensors. Thus, to some extent the activities considered in this system would support the main aspect of the Abt's ideas since children are allowed to create the entities and later use them to perform a public story intended as a learning task.

As seen in the previously described related work, there are many different technologies used to support the idea of creating some sort of games or interactive media. The difference with regards to our approach is primarily the social component provided by tabletop technology. It supports more balanced interaction in smalls groups and natural interaction with fingers and hands. Moreover, the combination of mechanisms for simulating and enacting ecosystems has not been considered in any of the other approaches. Our work unites the physics-based simulation along with logics behavior based on rules. It provides a higher level of customization and possibilities for a wider range of activities.

3. PHYSICALLY-BASED ENTITY MODEL

Aiming at an environment in which the main tasks would focus on the creation of game ecosystems, it is important to consider the factors that will determine the primary editing units of such ecosystems. The ground technology being considered is an interactive tabletop. In contrast to other existing technologies, we have chosen this one because we are interested in fostering collaboration and discussion during the task of creating the game. This social aspect is relevant as it enhances the learning process by exchanging and refining ideas between participants. Because this technology supports easily and effectively the sharing of digital objects on the flat surface (Hornecker, 2008), the characteristics of the game ecosystems to be elaborated should take advantage of this. Thus the game ecosystem should consider the composition of parts, being carried out collaboratively, and the interaction techniques should be easy to perform following typical gestures such as tapping, dragging fingers, etc. Specifically, we have considered a game ecosystem to be a surface where 2D virtual entities can be located and simulated according to intelligible physics (i.e. kinematics).

In addition, we have taken two design decisions with respect to the form of the entities that will determine the capabilities of the model required and the range of alternatives for learning activities in the future. Firstly, we would like to be able to include concepts such as the instantiation of entities. It would allow creating instances at runtime under some conditions, reusing preexisting definitions and dealing with abstractions. It would be what classes are for OO programming languages. These elements are useful to develop the computational thinking skill (Wing, 2006), considered as a very important one for personal development.

Secondly, although entities could be single bitmaps or sprites, and this would be probably enough to build most 2D game ecosystems, we are interested in allowing also the definition of entities composed of architectural units in the same way that a skeleton is composed of bones. In this way, we could also support the creation of more advanced behaviors and the more interesting learning scenarios based on storytelling containing puppets.

To fulfill these goals we have defined a meta-model to cope with the visual representation and physical properties of entities (see Figure 1). Basically, the stage or game ecosystem is composed of embodied entities. An embodied entity-type (*EmbodiedEntityType*) is physically defined in terms of its internal structure (*StructureType*) and the protocostume (*ProtoCostume*) covering such a structure. The structure can be composed of a single component (*StructuralComponent*) or several components put together by joints (*StructuralJoints*) to create more complex structures. These components are simply covered by a visual representation by means of images (*ProtoSkin*). Figure **2** shows several examples of entities that can be built using the meta-model. Essentially, the structure-type is like the internal skeleton that determines the physical simulation whereas the proto-costume is like costumes or the dresses of the structures used for visual representation purposes. From these elements, entities are simply instantiated embodied entity-types.

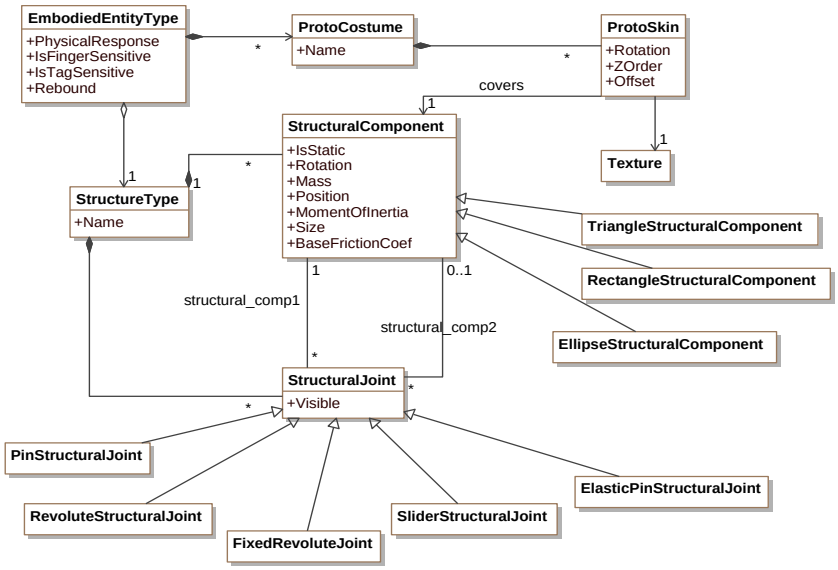

Figure 1. Meta-model for entity types.

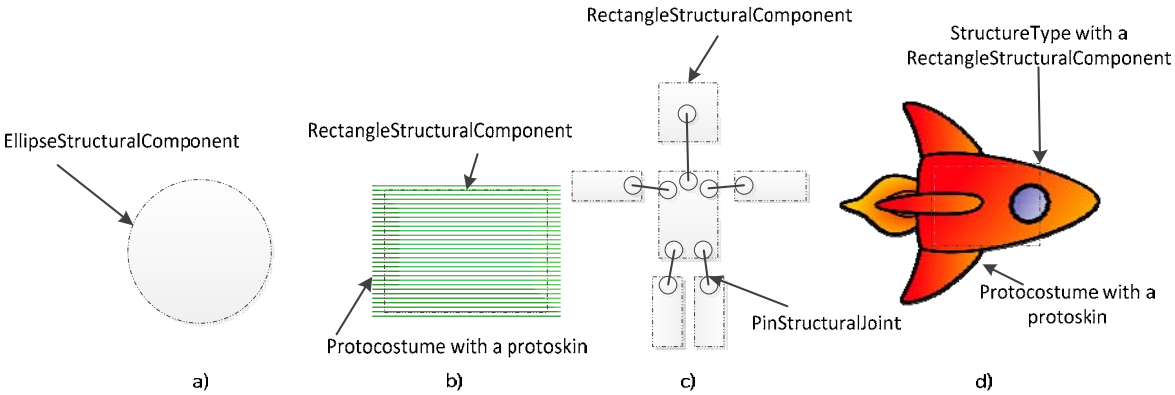

Figure 2. Entity types examples.

4. ARCHITECTURE AND ENACTMENT MODEL

The designed architecture leading the implementation of the middleware to process and enact ecosystems defined in terms of the meta-model described above is explained in this section. There are three layers in which the middleware can be broken down (see Figure 3). The Model layer refers to the definition and specification of the game concepts and their storage. This layer is responsible for managing the concepts presented in the previous section concerning the stages, entities, rules, etc. available in the game ecosystem being created, by giving support to update and retrieve all these basic elements.

The Controller layer holds the core functionality to orchestrate the simulation of a world stage. Basically the simulator has to take a stage to be simulated from the ecosystem, and all the data from the Model layer. The simulator has an event queue for the event occurrences produced during the simulation. Three types of events are queued: those thrown by the actions when executed in this layer, those being consequence of the physical simulation carried out by the underlying physics engine; and those related to the gestures or interactions of the user on the surface. This queue is regularly consumed by the rule processor, which determines which rule must be triggered, and eventually performs the execution of the action of the matched rules. Actions can involve changes in logical entity properties (e.g. increment the variable “Hits” of a block in a Breakout game) or in visual properties of the game (e.g. change of the visual representation or skin of an

entity). The Controller layer uses the services from the Model and View layer to address these two types of run-time changes respectively. In this way, the simulator controls the evolution of the stage simulation by consuming events and invoking services on the model or view as needed.

Finally, the View layer is responsible for visualizing the representation of entities under simulation. It offers a core set of view services that allow us to include entities and change their visual properties from the Controller layer.

The View layer is by far the most complex layer to implement. In order to perform the visualization of physics components, this layer relies on the Farseer physics engine[1] . It is an open-source physics engine simulator that allows the simulation of shapes defined internally in terms of bodies, geometries and joints. Since Farseer does not understand the concepts in which the visual structure of entities is defined (i.e. *StructuralComponents* and *StructuralJoints*), the View layer has to translate the components of the entities in our model into Farseer primitives. Moreover, it also has to maintain the correspondence between these elements in Farseer and the model in order to be able to track which entities are eventually producing physical events such as collisions. In this way, when two shapes collide, the middleware is able to determine which entities these shapes belong to, and therefore it is able to throw a physics event occurrence associated to the involved entities that will be queued in the Controller layer for further processing. All the components of this middleware have been implemented in C# and using the Microsoft Surface SDK.

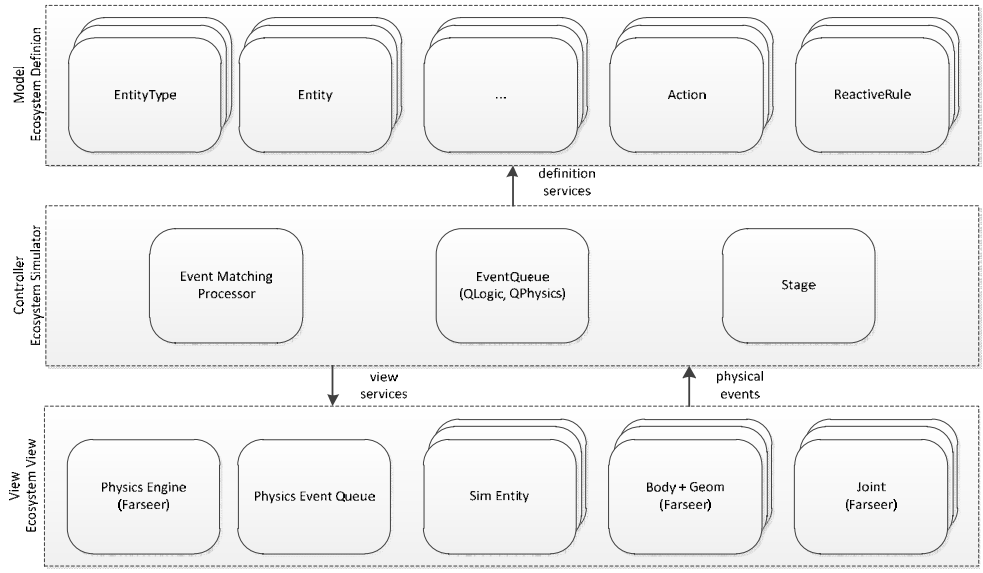

Figure 3. Architectural logical view of the implemented middleware.

5. MIDDLEWARE VALIDATION: A BREAKOUT GAME

The middleware allows the instantiation of any game ecosystem defined in terms of the model. Tests are necessary to prove that the concepts and the definitions present in the meta-model along with the corresponding implemented operationalization are functionally correct. To carry out this kind of validation, vintage game ecosystems seem good candidates for the test because they are well understood and provide a fair level of complexity. In particular, a basic *Breakout* game[2] ecosystem is presented here.

In the original game, there are some layers of bricks filling the upper side of the screen, a paddle in the lower side and bouncing ball. The user has to move *left-right* the paddle to prevent the ball reaching the lower side of the screen, and he/she has to get points by breaking bricks when the ball hits them.

For the present use case, we have considered three extended features with respect to the original game: bricks have a longer life, represented by an integer property “Hits”, which counts how many times the brick

[1] Farseer Physics Engine in Codeplex: http://farseerphysics.codeplex.com/
[2] Breakout Videogame: http://en.wikipedia.org/wiki/Breakout_%28video_game%29

can be hit until destruction; there are two paddles whose scope is limited to a half of the stage intended for two players; and two players are present, who get points up in their scoreboards when they break bricks after hitting the ball with their own paddle. Figure 4 shows a picture of a user playing the game ecosystem being enacted.

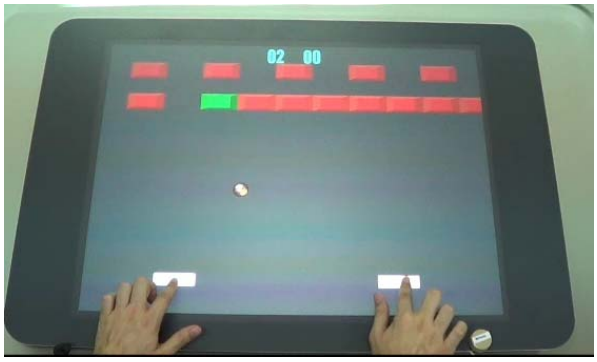

Figure 4. User playing the game ecosystem being enacted by the middleware.

The use of this game as a testing example is suitable for several reasons. Firstly, it is simple enough to perform a meaningful proof. Secondly, it requires multiple instances of the same type (e.g. several bricks) which could have different property values (e.g. one brick could require 2 hits in order to be destroyed whereas others could require only 1 hit). Thirdly, the type of rules required to control the evolution of the game are simple enough as to be intelligible and explained. Overall, it requires that the orchestration performed by the Controller layer to work properly, combining the physical events (e.g. event Collision between ball and brick) and logical events (e.g. the property Hits of a brick reaches 0, and as a result it will have to be destroyed and removed from the on-going simulation).

The screen has been modeled as a stage in our middleware. The main entities to be simulated are the bricks, the paddle and the ball. In addition, displays for the scoreboards are defined. All of them required defining the corresponding *structure-type* and the *protocostumes* in the Model layer, which specify the physics behavior and the visual representation of the entities respectively. **Erro! A origem da referência não foi encontrada.** illustrates graphically these concepts from the model. Note that two protocostumes have been defined for bricks since we would like to represent the bricks in a different way according to the value of the Hits property (e.g. in red those whose Hits property value is set to 1 and green when it is set to 2). The definition of types, structures and costumes provides the specification of the look and the behavior in terms of physics. Moreover some rules have been defined in the game ecosystem, which represent the logics of control governing the gameplay. Although it is not possible to show here all the required rules for space limitation, Figure 6 shows two sample rules defined in the ecosystem. They illustrate the expressiveness supported by the rule language implemented in the middleware. On the one hand, Figure 6(a) specifies a rule that establishes to 1 the property *LastPaddleHit* of the *Ball* when the *Paddle1* collides against the *Ball*. It will be useful to know which scoreboard has to be incremented when the ball hits a brick. On the other hand, Figure 6(b) specifies that the property Hits of the brick collided by the Ball has to be decreased when the collision occurs.

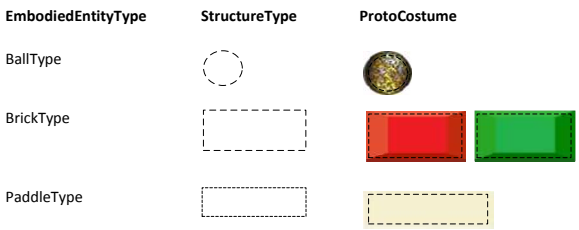

Figure 5. Breakout game modeling.

```
PRIORITY: 10
IF S: Paddle1 THROWS an EVENT E: Collision
AND E.SlaveEntity = Ball
THEN WITH T: Ball
PERFORM O: T.LastPaddleHit = 1
```

(a)

```
PRIORITY: 15
IF S: ANY BrickType THROWS an EVENT E: Collision
AND S. Hits > 0 AND E.SlaveEntity=Ball
THEN WITH T: ANY BrickType
SO THAT E.Entity = T.Value
PERFORM O: T.Hits = T.Hits -1
```

(b)

Figure 6. Sample rules.

6. ENHANCING LEARNING: PLAYING TO CREATE GAMES

The implemented middleware presented in this paper allows the creation of both simple and complex entities enacting them in the world ecosystem according to physics principles. Since the system supports not only the creation of game ecosystems but also the creation and customization of partial ecosystems to be completed, this platform facilitates a range of activities that can be proposed to enhance learning, following the ideas discussed by Abt and Gros in their respective works and leading to activities under the vision of "playing to create games". It provides enough flexibility for teachers to design sessions and decide which parts should be already provided and which ones should be created or completed by pupils.

The activities can be oriented towards several core tasks. For example, the system can be used to create characters for storytelling purposes so that the surface is the stage and anchorage points for entities and gravity are defined so that entities resemble like flattened puppets to perform the story. Another possible scenario can be focused on teaching essential physics by creating single entities as punctual masses aiming at some specific goal and then exploring how the system evolves.

In addition, more complex and complete ecosystems can be devised by including reactive behavior expressed by means of rules, as shown in the *Breakout* game example. Besides the specification of the visual aspect, the rule behavior can be edited by means of a rule editing tool invoked from the world editor. The implemented editor follows a set of drag& drop interaction techniques for some part of the rules combined with data-flow expressiveness. Data-flows allow users to edit expressions collaboratively using their fingers and some tangible objects like pucks for input on the surface without using any keyboard or mouse peripherals. Each task to complete the definition of a game ecosystem is oriented as a collaborative learning activity.

To demonstrate the flexibility and usefulness of the system for creativity learning tasks, the system has been used in two experiments involving teenagers. In these experiments subjects faced problems such as creating articulated entities (Catala, 2012a) and functional Rube-Goldberg machines (Catala, 2012b). The activities were considered within a discussion-action-reflection loop so that students could create the proposals collaboratively and interact between them. The tests showed that tabletop technology maintains students motivated, helps in sharing digital objects more effectively and that the aforementioned loop facilitates fairer co-operation interaction patterns, which are positive in terms of social skills development. Figure **7** shows two subjects interacting to create a structure for their puppet-like entity.

Figure 7. Two users creating an entity.

7. CONCLUSION

On the idea of providing a platform for learning activities under the vision of "playing to create games", this paper describes the implemented middleware to support such a learning framework from a technical point of view. It relies on a meta-model for physically-based 2D entities that can be enacted according to physics principles and rule-based behavior. The system uses an interactive tabletop as ground technology, which provides several advantages over other existing approaches. It does not only keep motivation in high levels but also encourages social participation and enables more natural interaction in collaborative tasks.

Our proposal focuses on the creation of several artifacts to create game ecosystems. It is flexible since it provides a desirable level of customization for teachers, so that they can design the learning sessions as needed. In this sense, the combination of two enactment principles (physics and rules) introduces even more possibilities for designing a wider range of activities than those proposals that only use either programmable simulations or storytelling live performances.

There are also some clear limitations. On the one hand, the use of the tabletop is not for masses but intended for small groups, so that the discussion and interaction can be properly conducted from a pedagogical viewpoint. On the other hand, the type of game ecosystems is based only on 2D performances and simulations, leaving out more advanced interesting scenarios based on 3D concepts. Nevertheless, it is still a powerful tool for teachers as it allows them to create partial ecosystems for many different learning scenarios adapted to their needs as discussed in the paper.

ACKNOWLEDGEMENT

This work was funded by the Spanish Ministry of Education under project TSI2010-20488. Our thanks to Polimedia for the support in computer hardware.

REFERENCES

Abt, C., 1970. *Serious Games*. Viking Press, New York, USA.

Begel, A., 1996. LogoBlocks: A Graphical Programming Language for Interacting with the World. MIT, Boston, USA.

Catala, A. et al, 2012a. Exploring Direct Communication and Manipulation on Interactive Surfaces to Foster Novelty in a Creative Learning Environment. *In International Journal of Computer Science Research and Application*, Vol. 2, No. 1, pp. 15-24, ISSN 2012-9572.

Catala, A. et al, 2012b. Exploring tabletops as an effective tool to foster creativity traits. *In Proceedings of the Sixth ACM International Conference on Tangible, Embedded and Embodied Interaction* (TEI '12), Kingston, Canada, pp. 143-150.

Gros, B., 2007. The Design of Learning Environments Using Videogames in Formal Education. *Proceedings of IEEE International Workshop on Digital Game and Intelligent Toy Enhanced Learning*, Jhongli, Taiwan, pp. 19-24.

Hornecker, E. et al, 2008. Collaboration and Interference: Awareness with Mice or Touch Input. *Proceedings of the ACM conference on Computer supported cooperative work*, San Diego, USA, pp. 167-176.

Lu, F. et al, 2011. ShadowStory: Creative and Collaborative Digital Storytelling Inspired by Cultural Heritage. *Proceedings of SIGCHI conference on Human factors in computing systems*, Vancouver, Canada, pp. 1919-1928.

McFarlane, A. et al, 2002. Report on the educational use of games. *Teem: Teachers Evaluating Educational Multimedia*, Cambridge, UK.

Maloney, J. et al, 2004. Scratch: A Sneak Preview. *Proceedings of the International Conference on Creating, Connecting and Collaborating through Computing*, Kyoto, Japan, pp. 104-109.

Michael, D. and Chen, S., 2005. *Serious Games: Games that Educate, Train, and Inform*. Course Technology PTR, Mason, USA.

Parkes, A.J. et al, 2008. Topobo in the wild: longitudinal evaluations of educators appropriating a tangible interface. *Proceedings of SIGCHI conference on Human factors in computing systems*, Florence, Italy, pp, 1129-1138.

Repenning A. et al, 2000. AgentSheets: End-User Programmable Simulations. *In Journal of Artificial Societies and Social Simulation*, Vol. 3, No. 3.

Wing, J. M., 2006. Computational Thinking. *In Communications of the ACM*, Vol. 49, No. 3, pp. 33-35.