MDA Game Design for Video Game Development by Genre

Vargas R. E.[1], Arellano G. G.[2], Beltrán H. D.[3],
Zepeda L. Z.[4], Quintero R. R.[5], Vega L[6].

Instituto Tecnológico de Culiacán
Sinaloa, México
{ing.rosavargas@gmail.com[1], gau_15@hotmail.com[2],
hector_beltran@outlook.com[3], leopoldozpd@gmail.com[4],
rquintero@itculiacan.edu.mx[5], vega.itc@salazarvega.net[6]}

Abstract. Game´s development process remains a difficult task due to game platform's increasing technological complexity and lack of game´s development methodologies for unified processes. In this work we show a way to develop different types of arcade games genre using Model Driven Architecture (MDA). We present a metamodel for game design that allows the specification for a high level abstraction independently of platform. This proposal shows that it is possible to generate a 2D game from the essential characteristics that make up such type of video game. Also, some model transformation rules to generate executable Java code from a specific model are shown.

Keywords: Model-driven software development, MDA, Game development.

1 Introduction

Several authors have shown that current game development is more difficult than a few decades ago. Due to the increasing technological complexity of the platforms for which these games are developed [1, 2]. Thus, standardized methodologies are needed to enable technological development [3]. Furthermore, during game development, error detection is also another frequent problem; to solve it, a common practice consist on edit code directly. However, after several iterations this causes loss of design documentation. Consequently, the code becomes harder to maintain and extend. Models are a rapid and inexpensive strategy to define a software solution [4] that preserve design concepts by helping to maintain design documentation [2]. In this paper we show a proposal for the development of different types of arcade games genre: such as mazes, shooting, racing and fighting. MDA is used to facilitate multiplatform software development while keeping-up with technological evolution [5].
The rest of the paper is organized as follows: Section 2 presents the State of the Art review. Section 3 describes MDA basic concepts. Section 4 illustrates the proposed design methodology. Section 5 draw conclusions and future work. Finally, section 6 presents some contributions.

2 State of the Art

In this section, we present contributions related to Game Design editing tools for tile based games creation. Gideros Mobile [6], is a platform for easy mobile 2D games and applications development. The platform allows developers to write their code once, but to run it on different resources. In addition, it aims to test applications immediately after they are written. The proposed design method maximizes the efficiency and effectiveness of the developers. Stencyl [7], is a platform for building 2D games for computers, mobiles and web. The platform consists of the following modules for game creation process: behavior editor, tileset editor, actor's editor, and stage designer. Allowing developers to write the code once and run it for any game´s style creation. Both of these tools are related to basic thoughts of MDA in terms of improving productivity and reusability. However, the editor tool [6] is not platform independent, it was built just for mobile and also have difficult access to native device features. Furthermore, editor tool [7] apparently the IDE, consumes too much RAM. In [8], the authors evaluates effectiveness of UML on game development as within other areas, through modeling can predict improvements in productivity. Such work evaluates the applicability and suitability for game development, together with the establishment of a process by using UML modeling. The introduction of object oriented and the MDA approach in game development supports productivity and reusability. On the other hand, to the best of our knowledge, only one effort has been development for aligning the design of video games with the general MDA paradigm. In [2] the authors show 2D game design through a metamodel allowing the specification using three fundamental perspectives: gameplay, control and graphical user interface.

3 Design Methodology

To explain our methodology design, we begin by reviewing MDA basic concepts and proceed with an example. The MDA approach [3] proposes to define software building process based on a set of models, a new way to develop software by transforming an input model into an output model. These models are organized and aligned in three viewpoints: 1) The Computation Independent Model (CIM) describes the system without showing details about how it is constructed. 2) The Platform Independent Model (PIM) reflects the functionalities, structure and behavior of a system and contains no specific information of platform or technology used in realization. 3) The Platform Specific Model (PSM) is more implementation oriented and corresponds to a first binding phase of a given PIM to a given execution platform. Using a series of transformations rules, also called model transformations, the software system is developed from a PIM to source code. The transformation rules establish correspondences between a source metamodel and a target metamodel allowing the transformation of models to be specified in model transformation languages such as QVT [2], MOFScript [3], etc. Transformation engine applies transformation rules in the source model and returns the target model. Following this approach we propose to overcome the game development process in three steps: In step 1, we define the PIM for arcade

games types. In step 2, an instance of the arcade PIM is created. In step 3, a set of model-to-text transformations rules produces the source code from the arcade model.

3.1 Step 1: Metamodel Definition

Figure 1 shows the PIM metamodel for arcade games types where the root element (*Game*) is generalized in the class (*Maze*). Also, classes are created and to build the *Maze*. They establish relationships among them by assigning names and cardinality. For example, the class *Maze* may contain one or more classes *Worlds*, which in turn contains classes *Enemy*, *Objetive*, and *Obstacle* that can appear in the game one or more times. The *Enemy* class has attributes such as name, number of enemies per level, speed movement, and motion animation. Also, attack move is specified by method *MoveAttack ().*

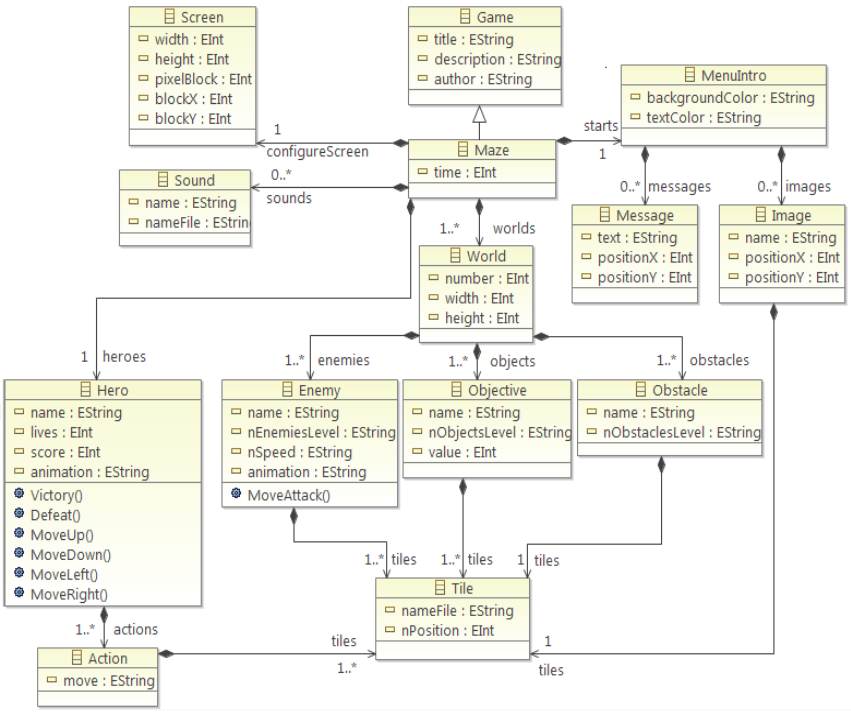

Fig. 1. PIM for Arcade Games Types.

3.2 Step 2: Model Creation

Once the metamodel has been created, a dynamic instance is also created to obtain an XMI file which defines values for the metamodel element's attributes. This in turn, allows to create a model of the desired game. In Figure 2 we present a XMI code

segment which represents an instance of the metaclass *Obstacle*, where obstacles elements defined as *rocks* will be represented by an image (defined in the attribute NameFile):

```
<obstacles name="rocks" nObstaclesLevel="27, 46, 47, 48, 49, 50">
   <tiles nameFile="image/rock.gif" nPosition="14"/>
</obstacles>
```

Fig. 2. Metamodel Instance.

3.3 Step 3: Model to Text Transformation and Code Generation

In Figure 3 we show one of the transformation rules of the enemies in the code generation phase by using MOFScript language [3]. This is a part of the template code, where is defined the corresponding tile and the file path for each element described as *enemy* or *obstacle* in the model. In the label *enemiesLevel* is specify its name and number of enemies per level. For this example, it specifies four enemies in the first level and so on:

```
mdl.World::generateEnemiesLevel(){
  self.enemies->forEach(e:mdl.Enemy){
   final int lev'e.name'[] = {' e.nEnemiesLevel'};\n'
   }
```

Fig. 3. Transformation Rule.

Figure 4 shows the game obtained after transformation process, in which four enemies can be represented by a *flame*. *Rocks*, that represent obstacles, may also be observed.

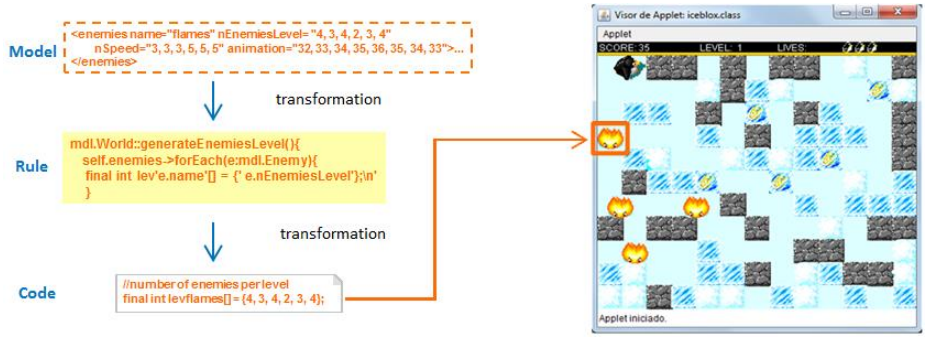

Fig. 4. Maze Game Generated.

4 Conclusions and Future Work

In this paper, we have introduced our MDA methodology for arcade game design. This work presents a semi-automatic process. The choice of MDA as a guiding methodology is justified by the need of building the video game using standardized

means. There are a number of advantages using MDA: 1) it is possible to check the transformation rules for correctness given that they are specified in a structured, precise way, independent of any implementation. 2) MDA models are a uniform and standard way for video Game Design and 3) this kind of transformations is carried out to support model evolution (Perfective and Corrective). As future work, we plan to extend the metamodel to create a design of a more complex games that offers greater expressiveness to increase functionality. This is because we work with models and we have the ability to extend our models at any time. Another aim is to develop a tool for game development using the model-driven approach.

5 Contributions

Our main contributions are: 1) a method for arcade games genre development. 2) a set of transformation rules to apply that supports our methodology and 3) a standardized methodology for novice game designers.

6 References

1. Blow, J. (2004, February). Game Development: Harder Than You Think. *ACM Queue, I*(10), pp. 28-37.
2. Montero, E., & Carsí, J. A. (n.d.). MDA y Desarrollo de Videojuegos.
3. Kepple, A., Warmer, J., & Bast, W. (2003). *MDA Explained: The Model Driven Architecture™: Practice and Promise of the Model Driven Arquitecture.* Pearson Education, Inc.
4. Quintero, J. B., & Anaya, R. (2007). MDA y el papel de los modelos en el proceso de desarrollo de software. *Revista EIA*, pp. 131-146.
5. Miller, J., & Mukerji, J. (Eds.). (2001, July 9). *Model Driven Architecture (MDA).* Retrieved from Document number ormsc/2001-07-01.
6. Gideros Mobile. (n.d.). *Official Site*. Retrieved from Gideros Mobile: http://www.giderosmobile.com/
7. Stencyl, LLC. (2013). *Official Site*. (Stencyl, LLC.) Retrieved from http://www.stencyl.com/
8. Inoue, T., & Shinkawa, Y. (2008). Applying MDA to Game Software Development. *CEIS*, (pp. 454-459).