A New Multiplayer Environment for Creating Adventure Games

David Gouveia, and Carlos Vaz de Carvalho

***Abstract*—**Nostalgia Studio is an extension of project SELEAG whose goal was to evaluate the use of serious games for learning history and social relations with an extensible multiplayer online game based on collaboration and social interaction. The game that was developed was a graphic adventure game known as Time Mesh which managed to achieve very positive results, but due to technical constraints, some of the initial goals could not be explored in depth. This article presents the first results of the implementation of this game editor and engine.

***Keywords*—**adventure games, game engine, game editor, game-based learning.

I. Introduction

THE Project SELEAG—Serious Learning Games—was an European project created to evaluate the use of serious games for learning history, culture and social relations, through the implementation of an extensible, online, multi-language, multi-player, collaborative and social game platform. The project counted with the collaboration of seven different partners and resulted in the creation of a point-and-click adventure game called Time Mesh [1,2].

Time Mesh tells the story of a child who stumbles upon a time machine, and uses it to travel back in time in order to correct certain events in history that have occurred differently from what we know. Most of the game is based on actual historical facts, and covers three different time periods: the age of discoveries, the industrial revolution, and the Second World War. It would have been interesting to allow more scenarios besides these three to be added to the game, but with the technology in which the game was created, this was not possible without significant resources and training.

The game was created using an Adobe Flash based game engine which allowed it be played directly from within most web browsers. The game was integrated into a webpage that required users to register and create teams in order to collaborate with each other. Collaboration with other teams was necessary because there were two versions of each game scenario, and knowledge from one version was required to advance in the other. However, the game itself had no knowledge about the existence of the other team, and all interactions were handled outside the game, on the web platform.

David Gouveia is with Virtual Campus Lda (e-mail: david_gouveia@virtual-campus.eu).

Carlos Vaz de Carvalho is with GILT – Graphics, Interaction and Learning Technologies R&D group of the Instituto Superior de Engenharia do Porto (e-mail: cmc@isep.ipp.pt).

Although Time Mesh was originally meant to be played autonomously by students in their free times, the game also ended up being tested and used in a classroom context, and has already been played by over 1.000 different users across several European countries, often as part of school activities. In general, results have been positive in terms of children enjoyment.

Nostalgia Studio is an extension of project SELEAG and solves the technical constraints of that environment. This article presents the first results of the implementation of this game editor and engine.

II. Adventure Games

Video games can be categorized under a multitude of genres, such as action, adventure, role-playing, simulation, or strategy, depending on their gameplay and interaction styles. This work focuses entirely on adventure games, so it makes sense to start by providing some insight into that genre.

There are some elements that can be found in the majority of adventure games [3]:

- They are usually built around a narrative...
- ...in which the player participates by assuming the role of a very small subset of its characters.
- They are usually driven by puzzle solving rather than by a physical effort...
- ...and rely mostly on exploration...
- ...object manipulation...
- …or dialogue.

Some of these elements make adventure games significantly different from other genres:

- Action - Instead of relying on the player's reflexes like most action games, adventure games rely on reasoning and logical thinking, which results in a more relaxed experience.
- Strategy - Although strategy games also rely on logical thinking, they usually require the player to control a large amount of units, such as an army or entire nation, while adventure games focus on the main character(s) in the narrative.
- Role-playing - While both genres have a lot in common, adventure games are deterministic in nature and follow a fixed path, while role-playing games have a statistical background which introduces randomness and variation.

They also lack the management components usually present in role-playing games (e.g. party management, skill management) [4].

Since adventure games place a lot of emphasis on story, it is common for them to draw inspiration from other narrative-based media - such as literature and film - and to encompass a wide variety of themes, from comedy to horror, or fantasy to science-fiction. On the subject of action games, there are some adventure games that incorporate action sequences into their gameplay, but they are always treated independently (e.g. as mini-games) and remain secondary to the narrative and puzzles. On the other hand, if a game has some adventure elements but the gameplay is all action based, it is usually categorized under the sub-genre of action games known as the action-adventure, which will not be covered here.

Based on the traits described so far, it is also easy to understand why there has not been much effort put into creating adventure games with a multiplayer online component. Most multiplayer online games rely on the game's progression being flexible, and allowing a large number of players to interact simultaneously. In an adventure game, where the entire story progression has been fixed from the start, and the number of controllable characters in the plot is limited, multiplayer game design becomes a lot more challenging. But on the other hand, puzzle solving and exploration are both elements that would benefit from cooperation in a multiplayer environment, at least on a limited scale.

Like almost every other genre in existence, the adventure game genre has a few sub-genres with distinctive features. The most common distinction is made between text adventure games and graphic adventure games. Text adventure games do not have any graphics, so they rely entirely on text to present the narrative and to receive player input. Text adventure games are also frequently known by the synonym, interactive fiction. Graphic adventure games, on the other hand, provide a visual representation of the world, and can employ a few different control schemes for player interaction. The most common control scheme for a graphic adventure game is the use of a pointing device (e.g. the mouse) or a touch screen interface, and games using that approach are known as point-and-click adventure games. The ratio between narrative and puzzles can vary widely from game to game, resulting in very distinct experiences. Some adventure games place most of their emphasis on puzzles, while demoting the narrative to a secondary role. These are sometimes known as puzzle adventure games, and a popular example of this type of game is the 1993 best-seller, Myst.

On the other hand, there are also some adventure games that focus almost entirely on narrative, sometimes at the cost of reducing interaction and gameplay to the bare minimum. A common example of this approach is the Japanese visual novel genre, which is a form of interactive fiction with anime style graphics, and minimalistic gameplay that consists mostly of a few branching choices in the game. But unlike most adventure games, visual novels usually have multiple story paths and multiple endings, and player choices have a strong effect in the outcome of the game.

III. Graphic Adventure Game Engines

A game engine is a software system designed to assist in the creation and development of video games. This is usually accomplished by providing tools and reusable components that would otherwise need to be implemented by the development teams. Some game engines, such as Game Maker, are very generic and flexible in order to be useful for different game genres (e.g. platform games, simulation games, first-person shooters). Other game engines, such as the ones described in the following sections, are designed for a specific game genre and usually provide most of the gameplay constructs required for that type of game, at the cost of abandoning the flexibility needed to create something different from that pattern.

While many games are still built from scratch or with custom game engines, an increasing number of game development studios make use of game engines that have been developed externally. This allows costs to be reduced dramatically and for teams to concentrate on creating actual game content. Even when game engines are developed internally, they are usually reused in future projects, so the benefits are the same. Some game engines such as the UDK or the CryENGINE bring the technologies that power some of the most advanced games to the masses, while others such as Unity3D are very simple to use and provide a very attractive platform for students, hobbyists and indie game developers to create their projects.

Finally we have those game engines that have been tailored for a specific game genre. Due to these constraints, it is possible to find game engines in this category that require no programming experience from the user, allowing people from all areas to create their own games. For example, two popular game engines in this category are M.U.G.E.N (for the creation of 2D fighting games) and RPG Maker (for the creation of tile-based role-playing games).

Adventure Game Studio (AGS) was first released in 1997 as an MS-DOS program called "Adventure Creator". It was developed by British programmer Chris Jones, and highly inspired by Sierra On-Line's graphic adventure games, as the first version of the engine was built specifically for low resolution, keyboard-controlled adventure games. The initial adoption of the engine was slow, but by 2001 it had achieved widespread popularity, and managed to keep an edge over its competitors (e.g. AGAST and SLUDGE) due to the size of its community and the large number of games being released every year [5].

The Wintermute Engine (WME) is a graphic adventure game engine developed by Jan Nedoma in the Czech Republic, and inspired mostly by the SCUMM engine and the Broken Sword series. The engine started out as a personal project of the author, but was eventually released to the public in 2003.

Wintermute had a great reception since the beginning, probably due to its balanced mix between usability and flexibility, and manages to remain a popular game engine to this day. One of the distinguishing features of this engine is the support for 3D characters, which was added in 2005, and allows the creation of games with graphics like Grim Fandango or The Longest Journey. The engine became open-source in 2007, and a portable version known as WME Lite was released in 2011, making the engine easier to port to other platforms, such as the Mac OSX 10.6 and iOS [6].

Visionaire Studio is a graphic adventure game engine that was primarily developed with simplicity and usability in mind. It started out as a school project by Timotheus Pokorra and Thomas Dibke in 1998, but they were later joined by Alex Hartmann and Robert Neumann as the project grew. The first public version of Visionaire was released around 2001, with the second version being released in early 2004. At the time of writing, the engine is in its third version and still being actively developed. The biggest selling point of the engine was that it did not require any scripting knowledge by the user, and every action in the game can be created directly through the editor's interface, although advanced users can still write scripts directly in the Lua programming language. The creators are also currently working on new additions to the engine, such as multi-platform support and 3D characters, which should make it an even stronger contender in the field [7].

It is easy to realize that there is a lot in common between the three game engines presented before such as having rooms, characters, objects, items, actions and dialogue sequences. That's because all of these traits are pretty much required by the majority of graphic adventure games, and therefore must be supported by any graphic adventure game engine.

In general terms, we could say that the after the game has been designed, the process of implementing a graphic adventure game goes through three different stages. The process is iterative and each of the stages will usually need to be visited multiple times and in different orders. Those stages are:

1. Creating the game world with all of its rooms.
2. Populating the game world with objects and characters.
3. Bring the game world to life with scripted actions and dialogues.

The first stage consists of creating all the environment that composes the game world, which in virtually every graphic adventure game, consists of a set of rooms. Rooms need to carry metadata that allows characters to navigate within and between them. In order for players to navigate within a room, the bounds of the floor and any obstacles in the room must be marked. As for navigation between rooms, that is usually accomplished by adding doors, or invisible regions in the rooms, that teleport the player to a new destination after examining or entering. This information is usually added manually by the developer when creating the room. The second stage consists of creating characters and objects and adding them to the rooms created on the first stage. Another important concept of most graphic adventure games is the item, which is an object the player can hold in his inventory for later use. Some items are directly related with objects in the world, while others are added directly to the player's inventories without. The third and final stage consists of adding behavior and interaction to the world. This is usually done by defining chains of automated action sequences that take place when certain events occur. This is the stage where the narrative and puzzles are added to the game.

IV. Nostalgia Studio

When looking into the target audience for Nostalgia Studio, it is clear that there are two very distinct groups of users: those who will be using it to create games, and those who will be playing the completed games. Nostalgia Studio handles this distinction by providing two separate applications: the Nostalgia Editor aimed at the first group, and the Nostalgia Client aimed at the second group. In order to reduce code duplication and improve the organization of the framework, both the editor and the client work on top of a third component known as the Nostalgia Engine, which contains all the shared functionality [Gregory, 2009]. Also, although the client can be executed independently from the editor, the editor does take advantage of the client in order to allow its users to preview the games they are creating. To summarize, Nostalgia Studio is composed by three separate components:

- Engine - The Nostalgia Engine is a dynamic library that provides most of the functionality required by the editor and the client. In other words, it does most of the heavy lifting for Nostalgia Studio. It is divided into two layers, one for low-level game-agnostic features (such as graphics, audio, input, networking) and one for high-level game-specific features (such as the gameplay, actions, rooms and characters).
- Editor - The Nostalgia Editor is the application used to create new games in Nostalgia Studio. It is based around a drag-and-drop interface and strives to insulate the user from the implementation details that are usually required to create a game. Its goal is basically to create content that the engine and client knows how to execute, and to provide a way to edit that content.
- Client - The Nostalgia Client is the application used to play games created with Nostalgia Studio. Most of its features are actually implemented in the Nostalgia Engine, and the client only behaves as a front-end for the player to have access to those features. It is also used by the editor as a previewer when creating games.

The figure below demonstrates the relationship between each of these components. Notice that the engine is internally divided into two separate layers, labeled Core and Adventure respectively. More information on this organization will be provided in the next section. Notice also how the relative sizes of each component varies, in order to reflect the fact that almost all of the functionality is provided by the engine, while the client does little more than act as a front-end for the engine.

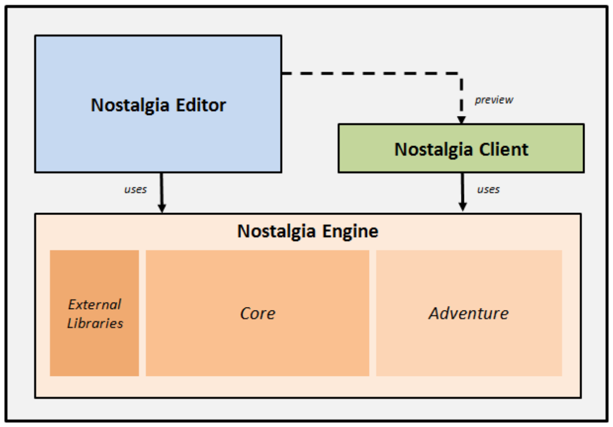

Fig. 1 Nostalgia Studio Architecture

A. Engine Architecture

Nostalgia Engine is the component that provides most of the functionality present in Nostalgia Studio. The most prominent detail in the engine architecture is that it is divided into two layers. Also, Notice the arrows in the diagram which represent the direction in which these layers interact with each other. The main goal of each of these layers is described below.

- Core Layer- Provides general game-agnostic functionality which might be useful for almost any game. This layer relies on several external libraries for support of low-level operations such as graphics, audio playback and networking.
- Adventure Layer - Adds functionality that is specific to the implementation of graphic adventure games. This layer is built on top of the core layer and does not communicate with any of the external libraries directly.

Each layer exists in separate library files (i.e. separate DLLs) which are included as references by both the engine and the client.

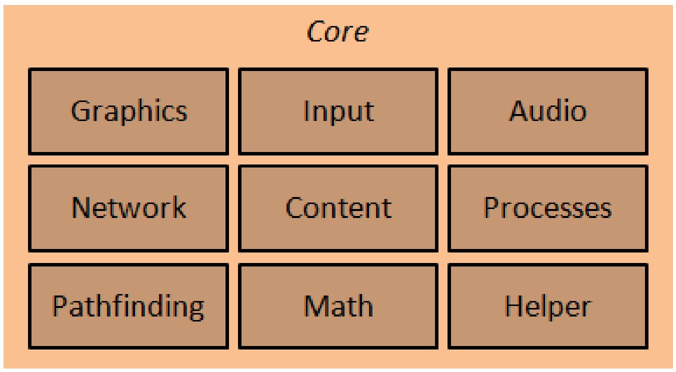

Fig. 2 Core Layer

Despite none of the features present in the core layer being directly related to the gameplay portion of graphic adventure games, this is still where most of the processing in Nostalgia Studio takes place. The functionality provided in this layer is extensively used by the rest of the application, and covers a very large spectrum of different technical areas.

Unlike the core layer which focuses mostly on general-purpose features, the adventure layer only provides functionality that is directly related to the implementation of graphic adventure games. It relies heavily on the core layer and ties all of the pieces together in order to create a working game. It is divided into the five modules represented below.

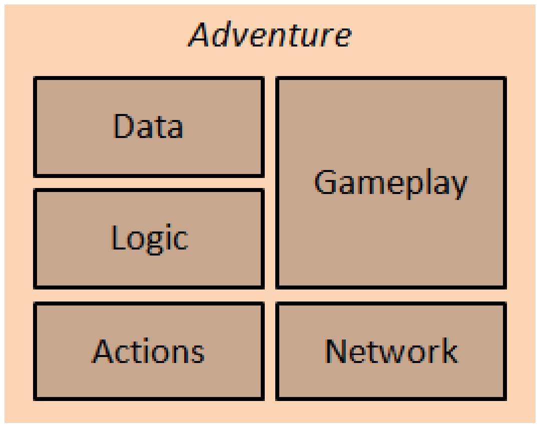

Fig. 3 Adventure Layer

V. Results and Discussion

After a complete prototype of Nostalgia Studio was implemented, it was sent to a selected group of users, together with a few instructional videos and a demo game that was built using the editor. This demo made use of the multiplayer features of the engine and was designed so that it required the two characters to collaborate in order to perform certain actions. After the users had experimented with the editor and the demo, they were asked to answer a survey about their general opinion of the editor and of the multiplayer component of the game.

First the subjects were asked how many adventure games they had already played before, in order to ascertain their familiarity with the genre. Two of them had never played any adventure game before, but the majority had played at least two adventure games before. Then their previous experience with computer programming was evaluated. Results were varied, with some of the subjects having no previous programming experience, while others had moderate or considerable experience. However, results from the next section showed that there was not a predictable correlation between having previous programming experience and being able to use the application.

Users were asked to select two out of six different categories related to the gameplay in an adventure game. The results show that the most important portions of an adventure game in the eyes of the users were the puzzles and the story, although a few users also favored exploration and characters.

On the second page of the survey, users were asked to rate their experience with the Nostalgia Studio application on seven different categories. Results were extremely positive with every category getting an average score above 4 values on a scale of 1 to 5. All of the averages were very close to each other it is hard to determine which category the users enjoyed the most, although statistically speaking, performance and visual appearance were the categories with the highest score.

In general though, the aspect about the editor that most users commented positively about was the editor's simplicity and ease of use, in particular among people with no programming experience, who were surprised that it was possible to create a game as easily as this. This result is extremely positive since that was one of the main goals that this project tried to achieve.

The third and final page of the survey was dedicated to evaluating the usage of multiplayer collaboration in adventure games. The demo game provided along with Nostalgia Studio was used as a tool for the users to experience this type of gameplay and form their opinion. They were then asked to rate the importance of five different features in the context of a multiplayer adventure game. This time the results were a lot more stable, and the feature that was considered most important for a multiplayer adventure game was having puzzles that required collaboration between players. This choice makes make perfect sense when we consider that the feature of adventure games that users voted on the most on the first page was the puzzles.

VI. Conclusion

This project started out as a way to address some of the limitations which were found during the development of Time Mesh. In particular, the tools which were used to develop the game were too complex for the general public, making it difficult for other people outside of the development team to create their own educational scenarios for the game. Furthermore, the multiplayer collaboration which was supposed to be one of the central points of the game, ended up being too disconnected from the gameplay itself, and was not given the proper attention.

As for Nostalgia Studio, there is still a lot of room for improvement. One of the top priorities for the application now would be add support for localization into the engine and the editor, and to incorporate a save and load system into the game. The interface of the action editor could also incorporate a visual programming scheme which could make the experience more appealing to the general public.

Acknowledgment

This paper has been partly funded by the project "Serious Games Project – SEGAN", no. 519332-LLP-1-2011-1-PT-KA3-KA3NW. Lifelong Learning. Education, Audiovisual and Culture Agency of the European Union.

References

[1] SELEAG. (2011). Serious Learning Games. Retrieved September 2011, from http://www.seleag.eu
[2] TimeMesh. (2011). Time Mesh. Retrieved September 2011, from http://timemesh.eu
[3] Adams, E. (2006). Fundamentals of Game Design. Prentice Hall.
[4] Barton, M. (2008). Dungeons and desktops : the history of computer role-playing games.
[5] AGS. (2011). Adventure Game Studio Legal Information. Retrieved January 2012, from http://www.adventuregamestudio.co.uk/aclegal.htm
[6] WME. (2010). Wintermute Engine License Information. Retrieved January 2012, from http://dead-code.org/home/index.php/license/
[7] Visionaire. (2011). Visionaire Studio License Information. Retrieved January 2012, from http://www.visionaire-studio.net/cms/licences.html