OpenGames: A Framework for implementing 3D Collaborative Educational Games in OpenSim
Ioannis Champasas, Ioannis Leftheris, Thrasyvoulos Tsiatsos, Theodouli Terzidou, Apostolos Mavridis
Aristotle University of Thessaloniki, Department of Informatics, Thessaloniki, Greece
ichampsa@csd.auth.gr
ilefther@csd.auth.gr
tsiatsos@csd.auth.gr
lterz@csd.auth.gr
apmavrid@csd.auth.gr

Abstract: The main goal of this paper is the presentation of the design and implementation of a game framework, for building 3D collaborative educational games. The final framework, which is called OpenGames, supports the creation of such games, in the virtual worlds of the open source platform, OpenSim. This paper presents the steps followed for the implementation of this framework, its technical aspects, as well as the first impressions and evaluation results after the conducting of a game based educational activity.

Keywords: 3D Virtual Game Based Learning, Virtual Learning Environments, OpenSim, Game Framework, Serious Games

1. Introduction

Game Based Learning (GBL) is a rapidly evolving process in which educational computer games are integrated into the traditional educational approach. This kind of games motivate and help students to acquire new knowledge and skills in a way that traditional teaching would not be feasible or could be difficult to achieve. In these games, educational and entertainment aspects co-exist and can help in achieving better learning outcomes. According to Prensky (2001), the basic features such games have are the following:

- **Rules**: Establishing some rules is essential for the smooth conduct of the game, as well as supporting the players.
- **Objectives**: Players should try to achieve a specific goal.
- **Result and Feedback**: The game should provide an additional incentive beyond learning, to motivate more the students. Furthermore, there should be appropriate feedback to each correct or incorrect action of a player during the game.
- **Concurrency**: The existence of competition between players or those and the computer can achieve better learning outcomes.
- **Interaction**: It should be clear that the decisions the players make, will be those who will judge the final result, e.g. between two groups answering questions, the winner should be the one, whose members studied more.
- **Plot**: In some cases the existence of a plot in the game can draw more attention from the students.

The goal of this work was to develop an easy to use and configurable game framework that can be used to create 3D collaborative educational games with such characteristics. OpenSim was the platform of our choice for developing this framework, after a research that took place at the beginning. The final implementation can be used to create hidden questions type games in which students participate in teams and try to discover and answer correctly multiple choice questions hidden in a virtual world. During the implementation, attention was paid into making the potential games as customizable as possible, so that are able to adapt to different learning objects or situations. After the implementation was over, and in collaboration with another thesis titled “Design and Evaluation of Cooperative Educational 3D Game,” a snapshot of a game was created in a virtual world of OpenSim. The result was the conduct of an educational activity in which undergraduate students of the Department of Informatics of our University participated. In this activity an educational game was created, with questions based on the undergraduate course “Web Learning Environments”. It was the first time the framework was tested in real conditions and the first impressions were very promising. The few problems we encountered were insignificant and they were not technical related but mostly procedural.

In order to investigate the effectiveness of the final framework, this paper addresses the following research questions (RQ):
RQ1: What feelings did the experience of 3D virtual environments create to the students during their participation in a game learning activity based on OpenGames framework?

RQ2: Can OpenGames be utilized to create serious games that motivate the participated students to study better the course material?

The rest of this paper is organized as follows. The next section describes the functional requirements that were set initially and which the final framework should meet. In the following section we present the gaming platforms Unity, Second Life and OpenSim. Afterwards we describe the components of the final framework as well as our first evaluation results and impressions by using it for the first time in the educational activity. Finally, some concluding remarks and suggested next steps to be followed are described in brief.

2. Functional Requirements

In this section we present the functional requirements of the implementation and analyze the rules and the functionality of the educational games that we want to be able to create with the framework. Before we begin, let us focus on the general objectives that were set initially:

- The final framework should provide flexibility.
- The structure of the code should be simple and open to changes without great cost.
- The game should be transferable in virtual worlds and the framework should operate independently of the objects that exist in each one.
- The educator should be able to easily change the questions and the number of these should be variable.
- Key elements of the game should be customizable.
- There should be detailed statistics after each game, available to all players.
- It should support multiple players per team and their number should be variable.

2.1 Educational Games

With the final framework we want to create 3D collaborative educational games (hidden questions type) in virtual worlds. Let us be more specific though and analyze the rules and procedures of such a game with the following snapshot of one.

2.1.1 Goal

In the game four (4) teams participate:

- Red Team
- Blue Team
- Orange Team
- Green Team

The goal of the game is players of each team to discover the hidden objects in the world that contains multiple choice questions and answer them correctly before the time runs out. Each correct answer earns points for their team and at the end, the team with the highest score is the winner of the game.

2.1.2 Preparation

Once the player enters the virtual world, he will see at the top left of the screen the HUD (Head Up Display), an object which provides useful information about the game:

- **TIME:** It displays the appropriate messages for team registration and ending of the game. During the game it displays the remaining time before it ends.
- **QUESTIONS:** It shows the number of all unanswered questions for each team. During the game the numbers are updated in real-time when a question is answered.
- **POINTS:** Here the player can see the points of each team in real-time. Each team starts with some bonus points.

When the player sees the message about team registration, he/she should go to the appropriate virtual location and register to his/her team by clicking on the corresponding object of each team. A message should appear for each registration and once all players have registered, the game can be started and the time at the HUD should start counting backwards.
2.1.3 The Game

Once the game starts, the players should try to discover the virtual objects which contain hidden questions in the game environment. By left-clicking on an object that contains a question, a question dialogue pops up, otherwise nothing happens.

The player can answer a question by clicking on one of the three available answers, while clicking the HELP button, the question reloads with a help text. Every time a player opens a question, his team is charged with some points and that’s why all the teams already have some points at the beginning. This also means that players from the same team should be careful not to open at the same time an unanswered question, because the points are charged more than once.

In any case, the answer of the fastest player of the team is submitted. Moreover, after each answer, there is appropriate feedback which informs the player for the result. Finally, if a player clicks the HELP button, his team is charged some points, and the help text will be available to his teammates in case he doesn’t give an answer, with no further point charging.
2.1.4 End of the Game and Winner

When one team answers all the available questions, the game time remaining decreases depending on our initial selection. If all teams answer all the questions or the timer expires, then the game ends automatically. The team with most points wins the game and furthermore, two prizes are rewarded to the teams that:

- answered the most questions correctly (Reward of the Wise)
- found all the questions first (Reward of Speed)

After the game ends, the players have access to detailed game statistics, by clicking the corresponding in-game virtual object.

3. Selection of a Platform for Implementing the Framework

In order to create educational games, such as the one described in the previous section, we should use a platform that provides all the tools we need to get started. It is obvious that a professional designer and programmer with the relevant expertise can create it from scratch with basic programming tools. Such process is time consuming and requires a lot of effort, so it is better and less risky to use ready-made software packages with which we can implement our base framework, focusing more on proper educational design and less on technical details. In this section we will briefly review some of the most popular platforms we can find on the internet and describe the reason we chose OpenSim. Of course, each one has its own peculiarities, but they are great tools for creating educational games, without that being their sole purpose.

3.1 Unity

Unity is one of the most popular development tool for creating computer games or other interactive 3D content. The environment runs both on Microsoft Windows and Mac OS X, while the games created with it, run on most platforms, from PCs to mobile phones and game consoles. The package includes a modern graphics engine and an editor, and is free to be used for non-commercial projects. This platform is ideal for creating any type of educational game, and offers all the tools one might need. Apart from the rich potential and the active community it has, it is not an optimal solution in cases where there is limitation of time (as in our case) or the technical knowledge required for creating the game. In such cases, we may need to turn our attention elsewhere, at higher level platforms, even if we face some limitations by making that choice.

3.2 Second Life

Second Life is an online virtual world, created by Linden Lab and became available in March 2002. Registration is free and the connection to its virtual worlds is made with applications called Viewers. Within these worlds, users can interact and communicate with each other, participate in activities or trade items and services. Moreover, with the appropriate fee, they have the ability to create new worlds and add interaction to them with the LSL scripting language.

Second Life is a platform that can support the creation of educational games and has all the basic tools for developing a learning activity, by giving us the ability to create 3D content and add the interaction we want. Furthermore, we can purchase or get free 3D objects from the online marketplace, which we can then be used in our implementation. The downside is the fact that we need to pay for the purchase or rent of virtual land in order to access the feature of content creation. Moreover, there will be further costs if we would like for example to upload images or textures to be used in our creations. These limitations may be vital for the choice of platform, no matter how good it is.

3.3 OpenSim

OpenSim is an open source platform for hosting virtual worlds. It can be used to simulate environments similar to those of Second Life, as it supports the core messages of the protocol. According to its developers the platform does not try to copy Second Life in any way, but it tries to be innovative and provide the basis for the server of the future 3D Internet.

The feature that sets it apart from Second Life is the freedom it gives. We can connect and work for free at some popular virtual worlds already created or even better, we can use the platform to create
our own virtual world and manipulate it in the way we like without further costs. To end up, the
development of the game on the OpenSim platform could take place in a fully controlled environment,
but something like that would require our own server.

3.4 Choice of Platform
The choice of the implementation platform was not a difficult task. We wanted to create in a short
period of time an infrastructure that would make it possible to easily conduct 3D collaborative
educational games with functionality similar to what described in the previous section. Immediately,
our attention turned to Second Life and OpenSim platforms. The reason was the fact that the
implementation could be done quickly in environments that already support multiple users in a 3D
virtual environment which can easily be modified depending on our needs (both for creating the world
and the interaction). Unity, though an eminently game creation platform with enormous potential that
exceeds all design goals set, will significantly delay the implementation, which is not desirable. Having
to choose between Second Life and OpenSim, we decided to use the latter for the following reasons:

- There is no extra cost for creating 3D content as with Second Life, while they share almost
the same features.
- They support the same scripting language LSL (almost all functions supported in OpenSim),
which serves our needs, in accordance with the functional requirements set.
- It gives us the freedom to control the implementation environment as well as the conduct of
educational activities. This is feasible because we own a server, something that maybe is not
available in many circumstances (it is mandatory for educational activities where many
students will participate).
- Another important advantage is that the final structure can be used to conduct educational
games in any world of OpenSim, as well as in Second Life with the appropriate modifications.

Finally, it should be mentioned that one of the most important disadvantages of our choice was that
the dialogues that can be created in OpenSim for multiple choice questions, have a limit on the
characters they display (that applies for Second Life too). This is something restrictive but not
prohibitive in using this platform for our implementation.

4. Framework Implementation
In this section we present the tools we used and the technical aspects of the final implementation. We
describe in detail the structural elements of the framework and analyze the structure of the code it
contains. Additionally, we present the configurable options we have before starting a game, as well as
the interface and the options of the admin menu that controls the flow of the game.

4.1 OpenSim
OpenSim was the implementation platform. Its development continues rapidly by the open source
community and it is constantly improved over time. Despite the fact that it is still in alpha version, it
has proven to be stable enough for real world usage. Installation and configuration are simple tasks,
and everyone with minimum effort can easily set up a virtual world to start working with.

4.1.1 Imprudence Viewer
Imprudence is one of the best viewers for usage with the OpenSim platform. It is based on the official
viewer of Second Life, and it is open source. Its development continues and it primarily focuses on
having the best compatibility with OpenSim. It has proved to be ideal for use with it during the
implementation, because other popular viewers such as Hippo or Phoenix caused a lot of problems,
interfering with our work.
4.1.2 LSL Scripting

LSL (Linden Scripting Language) is the scripting language of Second Life and it is largely supported by OpenSim. Its syntax is similar to C language but its logic is different. It is essentially a state-event driven language, which means firstly, that code is executed when events occur in the virtual world and secondly, there can be multiple states in the same script which contains different code (to be executed for the same events, depending which one is active). LSL is the foundation on which our framework was build.

In practice, each object in the virtual world can contain multiple LSL scripts, which run in real time and perform various tasks, depending on virtual world events. For example, we could have a script in an object which sends a friendly greeting to a user as soon as he clicks on it. In general, LSL seems very promising and has great potential. It can be used to add interactivity, to save data, to send messages or even communicate with remote servers and other in-game objects’ scripts to exchange data. Its syntax is simple but it comes with some limitations. For example tables, as we know them in other languages, have a poor implementation, called lists, which are not as flexible and powerful. Certainly, however, LSL is proved to be the key element of the platform which makes it ideal for creating 3D educational games.

4.2 Framework Components

The final framework consists of three major elements, which are the LSL code, the HUD object and the statistics website. Each one plays an important role in our implementation, and their aspects are described in more detail below.

4.2.1 LSL Code

LSL code is the most important element of the framework, as it is responsible for conducting a game. It consists of 16 different scripts, which communicate and exchange data throughout the game.

- **Base script**: The main logic of the game exists here. All the customizations can be made in this script with easy to change variables at the top of it. Also, an authorized user gains access to game menu when he touches the object that contains it.
- **Team registration scripts**: These execute team registration functions. Four different objects are required, with each one registering a user to the corresponding team, when touched by one.

- **Questions scripts**: This script should be included in every object we want to display a question, when a player touches it, during the game. We were able to create identical code for all the question objects with the contract that they need to be given specific names: 'Q1', 'Q2', 'Q3' etc.

- **Results script**: After the game ends, when a player can touch the object with this script the statistics website will show up.

- **HUD scripts**: Includes 9 scripts which display game information received by the base script to all players.

### 4.2.2 HUD

The framework includes a ready to be used HUD object, which consists of 10 different objects linked as one. It can be imported to a virtual world and then initialized by placing the HUD scripts to the corresponding linked objects. Obviously the creation and use of any such object is possible.

### 4.2.3 Statistics Website

The statistics website is a dynamic PHP-MYSQL site that can be uploaded to any server with MYSQL and PHP support, and be used just by entering its URL address to a variable in the base script. It uses a database to store the data that the script automatically sends when the game is over. Also, HTML, CSS3 and Javascript code was created for the presentation of this data. Finally, it is worth noting that the statistics contain information about the finish time of each team, its points, rewards, correct and wrong answers as well as the questions for which help was requested.

### 4.3 Configuration and Management of the Game

The game can be managed by a basic menu, which appears only to authorized users, by clicking the object which contains the base script. Before we look at the options provided, let us present the customizations that we have at our disposal. At the top of the base script we can change a lot of game variables such as:

- The name of the game
- The statistics website URL address
- The duration of the game in seconds
- The game time left after one of the teams answers all the available questions
- The maximum number of players per team
- The bonus points given to each team when the game starts
- The points a team will be charged when a player opens a question
- The points a team will be charged when a player asks for help in a question
- The points a team will get when a player answers correctly a question
- The list of names of authorized users
- The list of questions
- The list of help text in each question
- The list of correct answers for each question
After the customization, we can navigate through 4 different game states from the menu options. In any state, there is the option 'Reset', which resets the game and all relevant data is deleted. Specifically:

- **Default**: The game has not started yet. By selecting ‘Setup’ we go to the next state.
- **Setup**: In this state players can register to their teams and after that, the game can be started by selecting the relevant option.
- **Game running**: In this state the game is running and the players start playing. From here we can either end the game manually or ask for the available statistics, up to that time. When the time ends, the game finishes and we go to the finish state.
- **Game finished**: By selecting the option ‘Running’, we can return to the game. The ‘BackUp’ option sends the statistics data, something useful in case the backup process goes wrong in the first place. Moreover, we have the option to display the final statistics to the game chat.

4.4 Code Structure

The code structure is simple and based on two key elements. One is the different code states the base script has and the other is the exchange of messages throughout all states of the game between the various scripts used by the framework. The structure of each LSL script has the following hierarchy:

- **Variables**: Declaration and initialization of script variables
- **Functions**: Declaration and implementation of functions (useful for code that often used)
- **States**: Several states can exist in the same script with the default always executed on script initialization. After that we can dynamically change code states in runtime.
The base script includes many variables, functions and states, and this is where the heavy workload happens. The other scripts usually use some variables and have only one state, the default. As mentioned, the central logic of the game (management, team registration, game data and statistics) occurs in the base script. The remaining objects are mainly used as intermediaries to send vital information to it for further processing. For example, when entering the Setup state all game objects automatically initialized and the questions, along with their help text and correct answer, are send to the appropriate question objects for local processing.

```c
/* Variables */
...

/* Functions */
...

/* States */
default
{
  ...
}
state setup
{
  ...
}
state running
{
  ...
}
state finish
{
  ...
}
```

Figure 5: Code structure
5. Activity

In this section we refer in brief to the educational activity that took place after the implementation was over and we present the first evaluation results. The framework integrated to a virtual OpenSim world that was created in parallel with the implementation. The activity took place in 2 phases with each phase involving twelve (12) students who were randomly divided into four (4) teams of three (3). The scenario and rules of the game are the similar to those described in the functional requirements section. Each team should discover and correctly answer as many questions as possible in the given time. At the end of the game, the team with the most points wins. We created thirty (30) multiple choice questions, which were added to the basic script of the game along with a help text. In addition, the following parameters were defined:

- **Game time**: 2 hours
- **Game time left after a team answers all the questions**: 10 minutes
- **Starting bonus points**: 1000
- **Points charged to a team when a player asks for help in a question**: 30
- **Points charged to a team when a player opens a question**: 20
- **Points the team wins for each correct answer**: 150

### 5.1 First Phase

In the first phase of the activity we encountered some problems that did not allow the proper conduct of the game, as was originally planned. These were not technical but mostly procedural issues, as some in-game question objects were left unlocked by accident. This led to some players removing objects from the game, something that made some questions unavailable and we had to end the game manually at some point.

The only issue concerning the framework was related to the results object that shows the statistics webpage in the in-game’s web browser. Many players touched it simultaneously and continuously, resulting in a delay or failure to open the browser. The issue was resolved by providing the URL address of the statistics website to all players via the game chat. Finally, it’s worth noting that one team lacked a player and had only two members, something that fortunately wasn’t an issue because the framework supports such cases.
5.2 Second Phase

In the second phase we had better luck as the game conducted smoothly and crowned with complete success. None technical issues occurred and the code executed properly throughout the game but unfortunately, as in the first phase, one team lacked a member. Also, during the game, a player said that the HUD did not display all data, but that was expected, since he removed and enabled again his HUD. In such cases the HUD will be updated only when the data changes. This was by design for saving resources by only updating data on the HUDs when we have to.

Figure 7: Second phase snapshot (students answering questions)

5.3 Evaluation: Methodology and Results

Before the activity took place, all players were provided with a detailed game manual in word format, which contained detailed instructions and images about installing imprudence viewer, entering the virtual world and playing the educational game. After the end of the activity students were asked to answer an electronic survey. The evaluation combined quantitative and qualitative data gathering techniques. Quantitative data were obtained from questionnaires, where qualitative data were obtained from individual interviews, personal observations and recorded videos. Concerning the participants' previous experience in virtual environments, 52% of them had an experience in such environments, which related to digital games or other educational software.

We then analyze the statistical conclusions regarding the initial research questions.

- **What feelings created the experience of 3D virtual environments to the students during their participation in a game learning activity based on OpenGames framework?**
  According to the questionnaire completed by students after their participation in the game activity, 40% declared that the feelings that triggered them the experience from 3D virtual environments were pleasant and 60% satisfactory.

- **Can OpenGames be utilized to create serious games that motivate the participated students to study better the course material?**
  According to the answers given, 52.5% and 28.5% out of the students “agreed” and “agreed completely” that the virtual game activity motivated them to study better the course curriculum, whereas 19% were neutral.

After the completion of the activity, students were asked to give a short interview on the experience gained from their participation in the game activity. The students in their majority stated that it was an activity that attracted and motivated them in general; “attractive”, “innovative”, “interesting”, “impressive” were some of the words that students used to characterize the specific activity. Some of the students expressed their interest in the future of the framework by suggesting updates and new features they would like to see. On the other hand, the interviews of the students did not highlight any
major issues with OpenGames and most of the few problems encountered were small and not related to the framework.

6. Conclusions and Future Work

In this paper we presented the game framework we created for conducting 3D collaborative educational games on the OpenSim platform. The final framework works correctly, it can be used easily and is highly customizable, so that meets the learning objectives the educator wants to achieve. It is a first step towards building a robust educational game framework, which proved to be stable by the activity that took place at the end of the implementation. The small technical issues arisen, was mostly due to the immaturity of the platform, but its active development reassures that they will be solved and new features will be introduced.

Our future work aims at improving the game framework by solving its small issues and enriching it with new features. We plan to create a web interface for customizing all aspects of the game, so that educators do not have to deal with LSL coding. Moreover, we want to implement the support of variable number of teams as well as new ways of using points to make the game even more interactive. Finally, we hope that the game framework becomes a tool that helps us build richer 3D collaborative educational games and better evaluate the learning outcomes by using them, along with traditional educational approaches.

References