Technical Analysis and Approaches for Game Development in Second Life

André Cruz
cruz@dotlink.pt
FEUP; INESCPorto
Porto, Portugal

António Coelho
acoelho@fe.up.pt
FEUP/DEI; INESCPorto
Porto, Portugal

Armando Sousa
asousa@fe.up.pt
FEUP/DEEC; INESCPorto
Porto, Portugal

Abstract — **The development platform of a digital game is frequently defined before any game design document. Without a proper examination, this could exclude other viable solutions. Although the online virtual world Second Life is sometimes erroneously denominated a game, it can be used as a development platform for many purposes. This paper analyses game creation within the Second Life virtual world and possible approaches for the implementation of game assets and game logic. There is also an analysis of what game types could be more suited for development under this platform. Experiences learned from the developed prototype are shown and hints are given as to evaluate different techniques and expose best practices.**

Keywords – serious games; virtual worlds; game development; second life; immersive environments; linden script language; sculpties.

I. INTRODUCTION

Games are commonly associated with the goal of entertainment, but they can also be used for learning, training, marketing and community-building [1]. They are also segmented into genres, determined by the game mechanics [2]. When developing a game, the orientation of their genre can be often conditioned by the goal, as for instance, it's commonly used the simulation genre for training purposes. According to [3], the delivery platform(s) and development tools should be selected only after determining the game goals and game concept. In many cases, however, this is already pre-established. Even if the concept had made these choices obvious, it may work in several platforms and a plethora of software tools can help build the application.

With the recent arrival of persistent virtual worlds, and millions of active users in a daily bases, new opportunities emerge in game production and delivery. A virtual world environment can be defined by the following characteristics: it exists whether a user is logged in or not, it is populated by many users, it provides the illusion of 3-D space, avatars represent users in-world, and an interactive chat function is available [4]. Second Life (SL) is an end-user virtual online world developed by Linden Lab since 1999. SL offers a free executable client for Windows, Mac and Linux platforms. The creation of avatars is free and easy although email confirmation is needed. The main hardware requirements are a high end 3D computer graphics card. Terrain licences must be purchased to Linden Labs and it is perfectly possible to build several stories high as avatars can natively fly on free air.

Virtual worlds like Second Life have recently become a possible option for game development, offering some interesting and unique characteristics that could be suited for some type of implementations, as they excel in interaction, multiuser experience and the prebuilt environment engine may decrease development costs and time. It can be also be the best option if the targeted user demographic is already a resident in the virtual world, but new or inexperienced users may have a lengthier learning curve, as they need to become familiar with the interface of the virtual environment, even before the game starts.

The structure of this document is as follows: nest section presents related work on the SL environment (and associated development platform). The following section is dedicated to the architectural issues of the SL development platform and, after that, recommended prototyping issues are studied. Conclusions include best practices, strong points and limitations of the usage of the SL platform as multi purpose game development tool.

II. RELATED WORK

While some private regions in Second Life are completely dedicated to work as a game, they can also be available in the top of a simple table or panel board.

Darklife [5] is an example of a complete region dedicated to a multiplayer role playing game, where players can work solo or as a team to defeat monsters, gather gold to buy items and receive experience points to develop their character. Beyond the fact that it uses items, like swords and shields, to increase the attributes of the character, it uses two very special items that are essential to play the game. One is the backpack and the other is the Heads Up Display (HUD). The backpack is a wearable object that controls the interaction between the player and the environment present in the game. Without it, it is not possible to play or interact with the other objects. On the other hand, the HUD provides the statistics data for the game, like the health points, magic points, gold, etc, and also some command operations like create a team, level up and define options.

The Pot Healer Adventure [6] is also a game implemented inside an entire region, but in this case is aimed specially for single player use. This is an adventure game where the player must explore a strange world and solve some riddles to progress in the story. It has a narrative that is essentially described in a written form, available in most cases through

note cards. It is also used a wearable item, in the form of a notebook, to save the game progress.

The previous case studies have the primary goal of entertain the player, but probably Second Life is a more adequate platform to the goals of learning and training. It is often used by universities and schools for teaching and research, as educators are constantly looking for the potential of games in virtual worlds to foster experimental and constructivist learning methodologies [7]. Ohio University within the Science and Technology Enrichment for Appalachian Middle-schoolers (STEAM) project developed a series of educational games for science classrooms [8]. Some examples are the Fruit Fly Genetics and Weather Challenge where both use extensively the HUD for the player interaction with the game.

In many situations of game development under the Second Life environment, it's common practice the usage of wearable objects that controls aspects of the game and HUD´s for present game information and offer additional interaction options. Nevertheless, these implementations are not mandatory for the creation of a game in Second Life.

III. GAME PRODUCTION PARTS

Independently of the platform and tools, any project to develop a game is essentially segmented in four distinct parts during production: a design part that encloses the game concept and game mechanics; a coding part that is related to programming and integrates the game engine and the game logic; an art part that includes all the assets needed to the game like sprites, 3D models, animations, audio, etc; and finally a management part that is specially important in large projects [9]. Although the design and management parts are not immensely affected by the delivery platform, the code development and art parts could be heavily conditioned.

A. Asset Creation for Second Life

In Second Life, the available types of content for creation are 3D models, raster images, sounds and animations. 3D models can be developed directly in the virtual world, through the use of the Build tool, allowing the creation of tridimensional primitives, usually called prims, like cubes, spheres, pyramids, torus, etc. These are prebuilt models limited to a minimum dimension of 1cm x 1cm x 1cm, and a maximum of 10m x 10m x 10m. The user can alter some of the object parameters like size, rotation, color/texture, and other features related with the behavior of the object within the virtual world. There are also some limited support for the modification of the primitives form, including hollowing, path cutting, and some rudimentary implementations of twist and other functions. A large part of the existing models in Second Life are simply constructed with this tool, for although the available prims are only fifteen, the most practical approach its to group several prim instances together (like building blocks) to create more intricate models.

There is currently no support for mesh files in Second Life, although it's aimed to be implemented in a future version of the player and a new beta mesh viewer is available for testing with COLLADA file support. The closest approach to mesh creation available at the moment is through the use of sculpted prims, also designated sculpties. A sculpty is a Prim-type polygon mesh with a fixed count of vertices (usually 1024) that is applied to a single prim. In reality, the predefined shapes can only be sculpted and bended, but the mesh itself must be kept with the same number of polygons. This mesh data is stored in a RGB image (usually 32x32) where each red, green and blue color value of each pixel represents a coordinate in space (x, y, z). This shape-defining texture is called the sculpt-texture or sculpt-map and in this way the base model becomes deformed by these color stored values. Nevertheless, this is still highly limited and due to their graphical nature they might load slowly. It should be used lossless compression or the pixel colors would become defaced and the model will have a different representation relatively to the original design, usually a complete deformed and misrecognised one.

There are several modeling packages specifically designed for the production of sculpted prims, like Sculptypaint, Rokuro, and Tokoroten [10]. But other 3D software packages can also be used for modeling sculpties, usually with plug-ins. Other practical approach is to export the model to an OBJ file and to convert to the sculpty image through a converter [11]. The most limiting factor is that the model must have a precise count of 1024 vertices at their higher resolution and therefore no geometry can be added or removed.

Prims and Sculpties support the appliance of textures with some parameters related to tilling, transparency, bumpiness and a few more. For mapping, only the default and the planar projection can be used, meaning that in cases where detailed texturing is needed, baking the unwrapped render result in a proper software is often used to workaround this situation, especially where pixel density across the sculpty surface is not uniform. Character cloth texturing is also available and common templates are used for mapping.

In Second Life, animations files are only applicable for player characters (avatars). There are several built-in animations available, but animations can be created in the Biovision Hierarchy (BVH) file format. The BVH is a non-XML hierarchical file format that specifies the angles between the joints of an avatar's body parts.

One drawback is the lack of a full definition of the basis pose [12] (this format has only translational offsets of child segments from their parent, no rotational offset is defined). It also lacks explicit information for how to draw the segments. While it is possible to create BVH files in a text editor, it's more practical to use third-party programs, such as Avimator [13]. Through the specified keyframes, eighteen body parts can be rotated in any axis and the root (hip) translated for character animation in Second Life. Other types of rigging are not possible and animating prims or sculpties is only possible by scripting.

Audio files are also a type of asset that can be used but have specific restrictions. They must be in Microsoft Wave format (wav), have a sample rate of 44100 Hz and a total playtime of less than 10 seconds.

B. *Programming for Second Life*

Second Life provides scripting capabilities trough Linden Scripting Language (LSL). LSL is a state based and event driven programming language. Usually a script is attached to an object content folder and used to program that object behaviour in the virtual world. A LSL file script supports variables, functions and states, but LSL is not an object oriented language. Classes and interfaces do not exist and concepts like inheritance and encapsulation are not supported. This is effectively a possible limitation for game programmers, as the majority of games use the OOP paradigm in its conception. Code implementation, testing and debugging could not be the most flexible either, eventually resulting into an increase of development time. Nevertheless, LSL has more than three hundred functions and over thirty two events. Many of the common operations used in game programming like collision detection, object instantiation, timers, etc, are present. Objects can be linked with LSL scripts, defining therefore their states, functions and listeners accordingly with event occurrences. LSL also supports server side communication through XML-RPC, although might be considered limited for certain tasks [14], and HTTP protocols. HTTP allows more flexibility and scalability in comparison with its XML-RPC counterpart. It also permits a fast secure object to object communication, enabling quick and easy communication between objects. In the context of game development, this is relevant as it allows instance data exchanges and game data storage.

IV. METHODOLOGIES / PROTOTYPING ANALYSIS

The authors have developed a prototype implementation derived from a project where the objective was to promote student integration in university. The goal of the game was multi-user experience building and team building issues. The user demographics were also clear: university students from both genders and with an age of about eighteen years old. To achieve this goal, it was needed that the design of the game promotes actions between the players that turn out to be susceptible integration activities. Therefore, the game design evolved around the concepts of communication and team dynamics.

A. *Game Concept*

The mentioned project was codenamed "FEUP Adventure" [15] and its concept was a multiplayer action game with three distinct levels. Each level needs to be played by two teams (acting and opposite team). The proposed virtual space has three game rooms and an additional free practice area. The game levels are based in controlling a player's character to avoid obstacles and eventually move objects. The goal is to score more points than the opposing team.

Each level is composed by a specific set of rules and objectives, designed to promote the creation of collaborative and adversarial experiences. Levels were designed to be simple but allowing different strategic approaches and promoting communication and interaction among colleagues, in an attempt to achieve the proposed goal of the game.

As the core of the game mechanics rely heavily on moving and pushing objects around the map level, the Second Life physics engine (Havok) was used to implement their behavior. Some trial and error was used to figure out proper configurations for moving objects with the player characters. The physics engine calculates the mass of an object based on its size (width, height and depth) and although it's possible to define a predefined material, this only affects friction, elasticity and collision sound. There is also the possibility to program the behavior in a non-physic object, but as setting a translation by script causes a 0.2 seconds pause over the routine, object movements can become very choppy. With the impossibility to define the mass of a physical object, some testing was conducted with size, material and shape settings to reach an acceptable result.

In the first two levels, teams gain points by carrying objects to a goal wall or platform. The approach was to make the objects harder to be carried by one player alone, trying to force students to play as a team and above all with coordination. Besides that, both teams are in a direct confrontation and different strategies can be adopted by each team, giving the possibility for the groups discuss the best approach. The third level was a little different, as evolved around jumping between platforms with obstacles to reach a finish line. The score was based in the time spent by an entire team to arrive there, focusing the objective in the group.

This level design has the intention that, during gameplay, students enforce colleague recognition, promote communication, strengthen relations and suggests leadership roles.

B. *Strategies*

The experience gained from the development of the 3 level game FEUP Adventure, led to the following strategies for development of similar type of gaming applications:

- It was found that ramps and balls are objects where it is safe to have interesting physics.
- It is also interesting to define carefully the sizes of the objects involved (only a limited number of object materials are available – materials define density, elasticity, friction and default sound)
- Creation of scripts for normal mechanics of the game (example – pushing the crate animation)
- Creation of the scripts for normal dynamics of the environment (example moving platforms, etc)
- Creation of other illusory animations (client side animations for visual interest only)
- Creation of special scripts (example for game creation / initialization / timings / collisions / special game rules – fouls, etc); in our game, the game initialization script was activated by the event associated with an administrator flipping a start switch
- Our strategy for team creation was to create the groups as available in second life platform which turned out to be lengthy; another probably better solution would be to allow wearable objects to define teams
- Scoring data is associated with a specific object like, in our case, our starter switch; storing data (scores, etc) was

achieved by communication with the server and storing data in the server itself (server side programming needed)
- Deployment of the objects was firstly tested by editing and finding best parameters for all game elements
- It is possible to implement some features by using resources based on the server but networking latency will likely generate slow responses in world interaction, something not interesting for the FEUP Adventure
- Mechanics similar to a platform genre game may not be ideal to implement in Second Life. Avatars on the top of translating and rotating platforms will be left in the same position by the physics engine and the need of some precision during jumps can be seriously affected by the system response.

The presented strategies are not the only ones possible as the development platform is very flexible. The presented strategies were, however, found to be useful and interesting.

V. CONCLUSIONS

One immediate conclusion of developing games in virtual worlds is that it can be very different in comparison with a common game SDK. Independently of technical limitations and arguably distinct workflows, the option of a virtual world as a delivery platform can be a valid option, especially when the game goal and design is defined to take advantage of the persistent 3D world with user selected avatars. Although many kinds of implementation can be used, it probably offers more advantages and opportunities to serious games due to the platform concept nature, despite the identified drawbacks.

The popularity of Second Life may have contributed to be an attractive target platform for game development, but building large and complex games might end in very long development times. The overall system performance must also be taken in account. Fast paced action games could not be well suited under this platform and precise physic simulations may be very hard to implement (if not altogether impossible).

But probably some of the major Second Life limitations into game development concerns is it lack of flexibility to create and edit game assets, the use of a non-object-oriented programming language and very restricted processes of interaction for the user.

Other very common practices in game development like cinematic scenes and non-player character animation are also very limited in this pre-established environment. Therefore, NPCs tend to be static objects that open message boxes when players interact with them.

Target end-users may be actual residents, but for new users, the new virtual world might present a lengthier learning curve, as they must became familiar with the Second Life interface first. External conditions may also affect the game experience as for example network congestion or excess of users inside the same region that may slowdown performance and provoke latency thus degrading the quality of the experience of the user.

Second life can be a very attractive platform to develop some types of games due to its easy of use, built-in functionalities and accessibility. Some of the genres more suited for this system could be board games, simple puzzle games, adventure games that focused in exploration and riddles, role-play games and some sort of action games that do not require high reflexes, precision or accuracy.

ACKNOWLEDGMENTS

This work is partially supported by the Portuguese government, through the National Foundation for Science and Technology - FCT (Fundação para a Ciência e a Tecnologia) and the European Union (COMPETE, QREN and FEDER) through the project PTDC/EIA- EIA/108982/2008 entitled "3DWikiU – 3D Wiki for Urban Environments".

REFERENCES

[1] J. Novak, "Game Development Essentials" Thomson Delmar Learning, pp. 65–72, 2005.

[2] J. Novak, "Game Development Essentials" Thomson Delmar Learning, pp. 85–83, 2005.

[3] N. Iuppa and T. Borst, "End-to-End Game Development" Focal Press, pp. 225–229, 2010.

[4] M. Dickey, "Brave (new) interactive worlds: A review of the design affordances and constraints of two 3D virtual worlds as interactive learning environments. *InteractiveLearning Environments, 13*(1-2), 121-137, 2005.

[5] DarkLife – Second Life's MMORPG. Available online at: http://darklifehq.wordpress.com/ (last consulted on April 2011)

[6] The Pot Healer Adventure. Available online at: http://slurl.com/secondlife/Numbakulla/214/17/2/ (last consulted on April 2011)

[7] C. Inman, "Use of Second Life in K-12 and Higher Education: A Review of Research", Spring 2010, [Journal of Interactive Online Learning], pp. 44-63, 1982.

[8] T. Couper, S. Carroll, C. Liu, T. Franklin and D. Chelber, "Using the Virtual World of Second Life to Create Educational Games for Real World Middle School Science Classrooms", 2008.

[9] E. Bethke, "Game Delopment And Production" Wordware Publishing Inc., pp. 39–62, 2003.

[10] J. Heaton, "Introduction to Textures, Animation Audio and Sculpting in Second Life", Heaton Research Inc, 2007.

[11] Sculpty Maker. Available online at: http://www.shiny-life.com/sculpty-maker/ (last consulted on April 2011)

[12] M. Ciragan, "Realistic Human Motion Animation with .bvh Files", CSCI 6350 Topics in Computer Graphics, 2009.

[13] Avimator - Second Life Avatar Animation Editor. Available online at: http://www.avimator.com/ (last consulted on April 2011)

[14] J. García-Zubia, J. Irurzun, I. Angulo, P. Orduña, J. Ruiz-de-Garibay, U. Hernández and M. Castro, "Developing a Second-Life-based Remote Lab over the WebLab-Deusto architecture", 2010.

[15] A. Cruz, A. Sousa, A. Coelho, "Using games to promote student integration in universities through the use of virtual worlds", SLACTIONS 2010.