3D MODELLING AND VISUALIZATION BASED ON THE UNITY GAME ENGINE – ADVANTAGES AND CHALLENGES

Ismail Buyuksalih[a], Serdar Bayburt[a], Gurcan Buyuksalih[a], A.P. Baskaraca[a],
Hairi Karim[b] and Alias Abdul Rahman[b]

[a]Bogazici Insaat Musavirlik A.S., Eski TUYAP Binasi
No. 50 Beyoglu, Istanbul, Turkey
{ismail.buyuksalih, serdar.bayburt, gurcan.buyuksalih,
apeyami.baskaraca}@bimtas.istanbul

[b]3D GIS Research Group,
Faculty of Geoinformation and Real Estate,
Universiti Teknologi Malaysia,
81310, Johor Bahru, Johor, Malaysia.
wmhairigis@gmail.com and alias@utm.my

KEY WORDS: 3D city modelling, 3D visualization, and Unity 3D game engine.

ABSTRACT

3D City modelling is increasingly popular and becoming valuable tools in managing big cities. Urban and energy planning, landscape, noise-sewage modelling, underground mapping and navigation are among the applications/fields which really depend on 3D modelling for their effectiveness operations. Several research areas and implementation projects had been carried out to provide the most reliable 3D data format for sharing and functionalities as well as visualization platform and analysis. For instance, BIMTAS company has recently completed a project to estimate potential solar energy on 3D buildings for the whole Istanbul and now focussing on 3D utility underground mapping for a pilot case study. The research and implementation standard on 3D City Model domain (3D data sharing and visualization schema) is based on CityGML schema version 2.0. However, there are some limitations and issues in implementation phase for large dataset. Most of the limitations were due to the visualization, database integration and analysis platform (Unity3D game engine) as highlighted in this paper.

1. INTRODUCTION

Research progress on 3D technologies for city modelling standards (e.g. CityGML) and platforms (e.g. Unity3D game engine) resulting a variety of 3D GIS applications in bigger scale implementation, for example the estimation of potential solar energy from building rooftops and façades of Istanbul city (Buyuksalih et al., 2017), another example of 3D city model for large city – city of New York by Kolbe et al. (2015).

3D applications have been developed with the capability to employ more specific 3D analyses (Kurakula and Kuffer 2008) such as noise distribution and solar energy estimation. Utilization of 3D city models for business or authority operations such as in urban planning are directly highlights the benefits of 3D city models (Moser et al., 2010).

Similarly, BIMTAS company has carried out a few projects for estimating potential solar energy on buildings and 3D underground utility mapping for Istanbul city.

The paper has been organized with a brief introduction on 3D GIS modelling standards, description of project area and Unity3D game engine as project implementation platform. Second section describes briefly on projects implementation for each project implementation process. Approach in developing 3D city modelling using Unity3D game engine platform will be described in section 3. Section 4 presents result of each project/application, advantages and limitation of using Unity platform. Lastly section 5 highlights the conclusions of the paper and future outlook.

1.1 3D Modelling - CityGML

3D CityGML models are commonly used for applications which require 3D mapping, realistic simulation, better visualization and analysis. The model mostly consists of polygons as the surface either façade, roof or more details building assessor such as balcony, doors and furniture.

CityGML version 2.0 offers five levels of detail (LoDs) to be incorporated in any 3D model (see Figure 1). A detailed level of CityGML model produces a better accuracy and precision of the measurement, visualization and analysis.

Figure 5 illustrates an example of different LoDs (LoD1 and LoD2) with respective analysis of shadow results. However, it consumed higher storage, graphic and processing of the devices (modelling hardware) compared to less detailed level. Thus, appropriate LoD should be carefully selected for any potential applications and desired accuracy on modelling or analysis.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-IV-4-W4-161-2017 | © Authors 2017. CC BY 4.0 License.

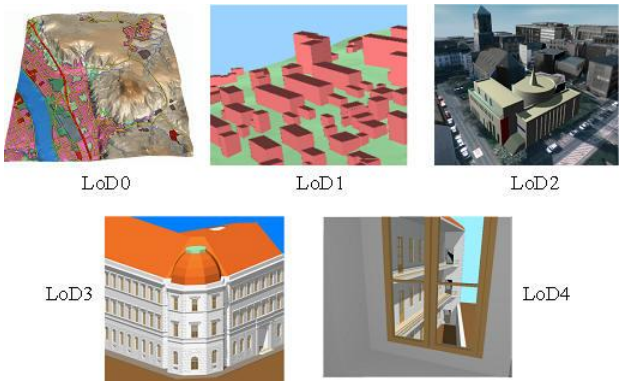


Figure 1. The LoDs concept in CityGML v2.0 standard (Kolbe and Bacharach, 2006).

1.2 Unity3D Game Engine

Unity is a cross-platform game engine designed to support and develops 2D and 3D video games, simulations for computers (e.g. Figure 2), virtual reality, consoles and mobile devices platform (Unity, 2017). It is commercial software developed by Unity Technologies. Unity 3D provides three dimensional manipulation and simulation through functions defined using programming languages.

The engine capable of making high quality games, design and develop 3D social network gaming platforms (Bae and Kim, 2014). In addition, 3D visualization, functions and attributes, intelligence and metric measurements can be done with Unity 3D encoding.

Figure 2. Examples of 3D scenes from Unity's sample projects (Unity, 2017)

Unity has three important components (Unity, 2017);

- A game engine: This allows the games to be created, tested and played in different environments.
- An application where the design or the user interface is put together with a graphics preview option and control play function.
- A code editor: The IDE provides a text editor to write code. However, a separate text editor is often used to avoid confusion.

Unity is an excellent platform to start game development and definitely recommended for developers who want to jump start game development. Unity provides numerous tutorials based video examples that make a developer familiar with the development process.

1.2.1 Built-in Game & Map Functions

Unity3D features such as flight, rotation, pan, zoom in and zoom out can be used. The camera, light and other types of objects are offered to the user in the scene. Standard mapping functions such as projection system, information, and on-off layer are built based on the Unity platform. Specific function such as query, shadow calculation, solar radiation and sun locator and angle are being embedded in the equations and internal codes (not in the viewer).

1.2.2 Built-in Analysis

Unity game engine provides a platform either to use their libraries or our own codes for supporting application analysis.
Built-in analysis such as estimating solar potential project uses several available libraries and localizes algorithms including effectiveness of roof-mounted and façade-mounted system, using photovoltaic (PV) and thermal. The project also implemented some related methods on solar analysis available in others solar estimation application - shadow calculation, sunlight location and angle, direct or indirect reflection and others.

1.3 Project Implementation Area

Building models can be extracted from both aerial images and LiDAR data (Wang et. al, 2008). Digital Surface Model (DSM) and Digital Elevation Model (DEM) for the whole Istanbul were acquired by the previous project (data collection using Airborne LiDAR), two years ago.

The first project is to estimate potential solar energy buildings, covers the whole Istanbul (5,400 square kilometres, around 1.5 million buildings) as illustrate see Figure 3. The project uses LoD2 and LOD3 CityGML standards as 3D city model schema.

The second project focuses on customization of 3D underground utility mapping and tools to support several 3D underground analyses. The Figure 4 shows the utility network over the city.

Figure 3. Istanbul province as the first project (estimating solar energy) study area.

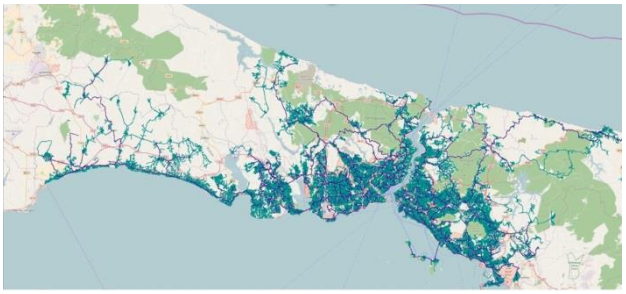


Figure 4. 3D underground utility mapping.

2. PROJECTS IMPLEMENTATION

This section is divided into two subsections - estimating potential solar energy 3D city modelling and 3D underground utility mapping.

2.1 Estimating Potential Solar Energy – 3D Istanbul

Solar energy modelling is increasingly popular, important, and economic significant in solving energy crisis for big cities. It is a clean and renewable resource of energy that can be utilized and solving power crisis to accommodate expansion of urban and the growth of population. However, as the spaces for solar panel installation in cities are getting limited nowadays, the available strategic options are only at the rooftop and façade of the building. Thus, accurate information and selecting building with the highest potential solar energy amount collected is essential in energy planning - installing photovoltaic (PV) system and panels.

Sunlight comes to the earth as a form of electromagnetic radiation. Solar energy is radiant light and heat from the sun which is very beneficial and crucial as renewal energy. Solar energy radiation at earth's surface varies due to absorption, scattering, reflection, change in spectral content, diffuse component, water vapour, clouds pollution, shadow (Figure 5) and others (Alam et. al.,2013).

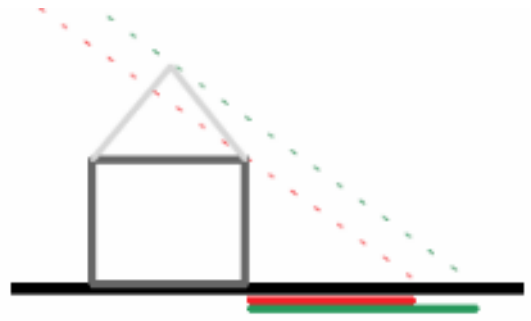


Figure 5. Effect of LoD1 and LoD2 in shadow calculation. (Biljecki et. al., 2013).

Solar potential analysis can be used to localize the most suited areas on a building in its urban (Good et. al., 2014). Several method of solar analysis was proposed in Good et. al (2014). The project also implemented some related methods on solar analysis available in others solar estimation application. They are shadow calculation, sunlight location and angle, direct or indirect reflection and others.

The implementation of this solar estimation project for Istanbul uses CityGML LoD2-LoD3. The model and analyses were carried out using Unity3D game engine with development of several customized tools and functionalities.

2.2 3D Underground Utility Mapping

The aim of the project is to provide the 3D modelling of the natural gas network with integration of the pipelines on 3D City Modelled Map. Natural gas network can be built by integrating 3D models metric and geometrically with high precision and 3D city model created in Unity 3D software (Windows application).

3. 3D GIS MODELLING IN UNITY

As industry experts and connoisseurs of the tool, they explained that one of the biggest beneficiaries of using Unity3D is Android. Being, in the market with more users and more devices in circulation has many advantages. This is especially reflected when analysed from a social point of view.

Unity is an excellent platform to start game development and definitely recommended for developers who want to jump start game development. Unity provides numerous tutorials based video examples that make a developer familiar with the development process.

- **Rendering**: The Unity graphics engines use OpenGL, Direct3D, OpenGl ES for mobile platform (iOS, Android) and various APIs. There is also support of reflection, parallax and bump mapping. It provides features to render text and use of shadow maps for dynamic shadows. Various file formats of different software are supported. For instance, Adobe Photoshop, Blender and 3ds Max are supported.
- **Scripting**: Scripting is built on Mono, the open source platform for.NET Framework. Programmers write the UnityScript similar to JavaScript, C sharp and Boo.
- **Physics**: The Unity engine provides built-in support for PhysX physics engine with real time cloth simulation on skinned meshes, collision layers and thick ray casts.

The project was completed in two years; starting from converting CityGML data LoD2 – LoD3 (with photos) into Unity supported formats (e.g. in this project mostly use .fbx format). The final results (estimated potential solar energy for each building façade and rooftop) are acquired via implementation flow as illustrated in Figure 6.

3.1 Pre-Processing Flow (Preparing input data)

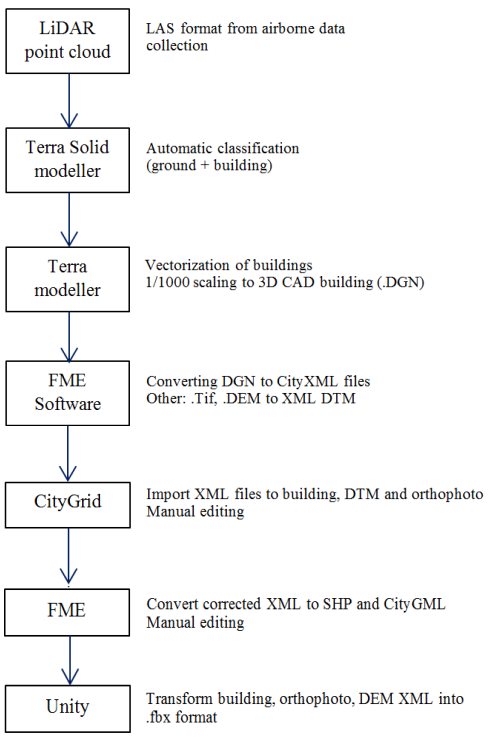


Figure 6. Pre-processing phase for unity input model.

3.2 Customized GUI and Functionalities

3.2.1 Estimating Solar Energy Project

The custmized Unity application was named as *Istanbul Gunes Enerjisi Verimliligi* as default view in Figure 7.

Figure 7. Default GUI view of Istanbul Solar Estimation.

3.2.2 3D Underground Utility Mapping

Conversion of CityGML data to Unity supported format (.fbx), projection, orthophoto images and basic mapping tools (zoom-in, pan, rotate, selection of object and others) has been completed. However the customized functions (see Figure 8) for supporting underground utility is still in progress.

3.3 Challenges

Estimating the solar energy/radiation from rooftops and facades has some limitations – e.g. the shadows from other neighbouring buildings. One of the biggest issues in solar radiation calculation in urban area is shadow casting. It affects the accuracy of the potential radiation results as for high storeys buildings. The size and position of the shadow caster directly impact on energy collection (Alam et. al., 2013). Solar irradiance can be estimated while incorporate shading effects of surrounding buildings with generated lines pointing towards the direction of the sun at the specific points in time (Wieland et. al., 2015). Alam et. al (2013) uses potential building surfaces of the city model (CityGML) and sub-divided them into triangles again and again into smaller triangle like octree hierarchical process. The coverage of the triangles (every level) will determine the solar irradiance of particular CityGML building surfaces.

For the 3D underground utility mapping, data interoperability of the utilities layer (pipeline, gas and others) is one of the highlighted challenges. Layers need to be prepared in valid topological elements (e.g. not intersect 3D, connected network) and it was extracted from shapefile format (.shp). Converting format 2D shp to 3D fbx format is one of the difficulties of this project.

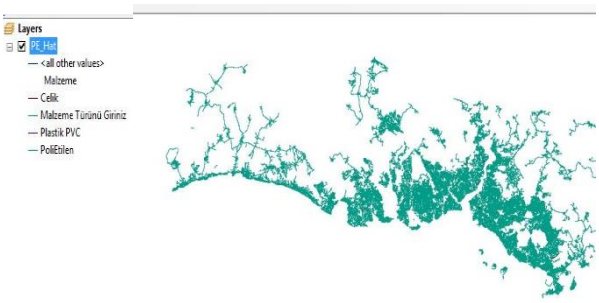


Figure 8. 2D utility layer in shapefile format.

4. MODELLING OUTCOMES, ADVANTAGES AND LIMITATAIONS OF USING 3D UNITY GAME ENGINE

Final outcomes are presented in section 4.1, while advantages and limitations are discussed in the sections 4.2 and section 4.3 respectively.

4.1 Outcomes

4.1.1 Estimating Solar Energy Project

The outcomes show the estimation of potential solar energy received for the whole area by day, week, month and year, thus the decision for installing the solar panel. Calculation of the energy will be made based on the rooftops or/and façades of a selected building. Figure 9 and Figure 10 show a selected building with rooftop as solar panel just before and after the calculation take place. While Figure 11 shows the calculation (estimation of potential solar energy received by) on the selected building façades.

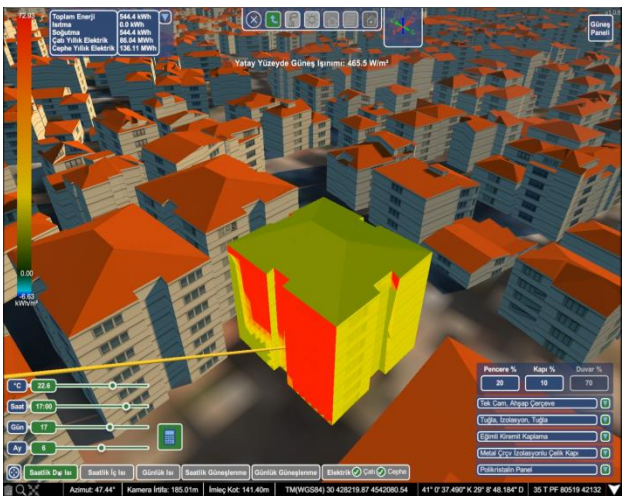


Figure 9. Selection of a particular building to be calculated.

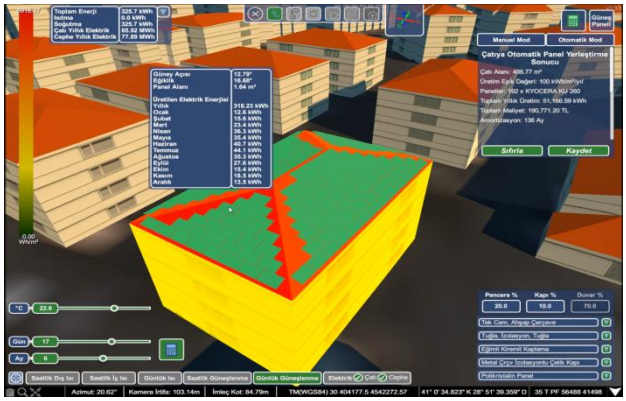


Figure 10. Solar estimation for the selected building rooftop.

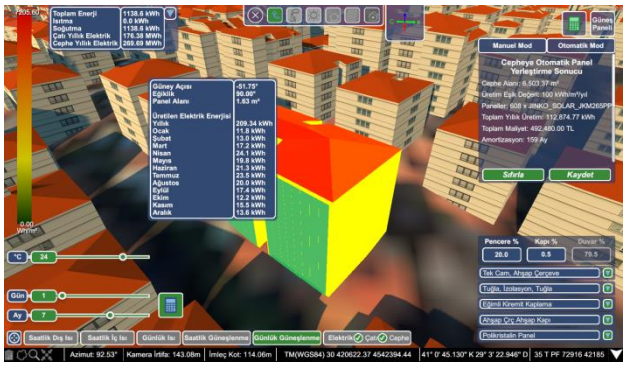


Figure 11. Solar estimation for the selected building façade (potential solar panel).

4.1.2 3D Underground Utility Mapping Project

This is a ongoing project - visualization of 3D buildings and data modelling (orthophoto-terrain and underground cables) are in progress as shown in Figure 12, 13 and 14 respectively.

Figure 12. 3D visualization of 3D terrain information.

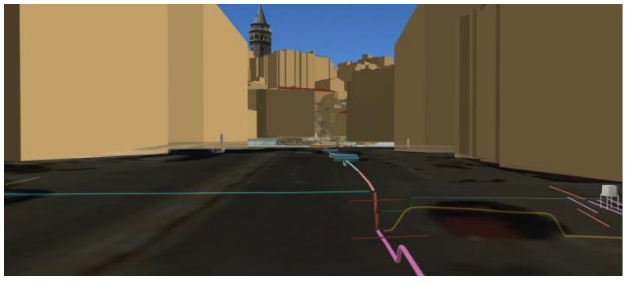


Figure 13. 3D visualization of underground utility (gas).

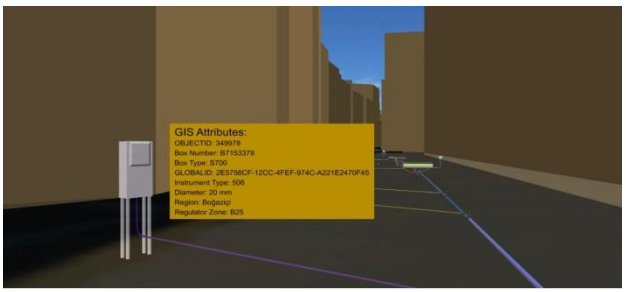


Figure 14. 3D visualization of underground utility (gas).

4.2 Advantages of 3D modelling in Unity

The main advantage of Unity3D is that it can run on multi-platform with online and offline modes on Windows, Mac, Linux and mobile platform (Ruzinoor et. al., 2014). Most of Unity 3D development projects are based on Android platform due to integration simplicity. Unity also allows user to publish the developed games on other platforms such as iPhone and Windows phones or tablets with large number of users. The game engine also capable of importing other 3D models in FBX, SBX and OBJ formats.

Unity 3D game engine provides good visualization and light rendering of 3D objects, thus major advantage for modelling 3D spatial geometry. It provides variety of functions and modules library for multi specifications of the gaming development (Bae and Kim, 2014).

For these two projects, Unity 3D provide a flexible platform especially to customize appropriate functions and modules for application needs. As compared subscribing commercial software such as AutoCAD Map and ESRI City Engine, only default functions are available.

4.3 Limitations of 3D modelling in Unity

Some general limitations for Unity 3D are as listed below:

- It does not allow developer to start from a foundation or template. Any game or project needs to be implemented from scratch to the detailed functions with GUI.
- From a graphical point of view, Unity game platform performs slower than other engines such as UDK (Unreal Development Kit).
- High costs.

Other specific limitions on 3D modelling (mapping context) will be highlighted in the following sub-sections.

4.3.1 No Coordinate Transformation Projections

Most of the 3D modelling software could not cope with spatial data is because of the projection system. Unity also did not support coordinate system as for their libraries/modules. There are no transformation tools or libraries for coordinate's conversions.

4.3.2 Format Interoperability Problems (GIS)

The Unity 3D game engine supports very limited input formats e.g. .fbx, .sbx and .obj which focuses on geometry; vertex and faces ordered. Thus, creates limited interoperability during conversion process such as loss of semantic and attributes (information in the column fields).

4.3.3 Single / stand-alone file and Limited Databases Support

Unity uses only a single file (or single file directory) to store all the codes, functions, GUI properties as well as the datasets (2D and 3D) of a project, thus consumes huge memories (RAM and ROM) of the computer system. Other disadvantage – it is a stand-alone system which hardly able to integrate with other software such as database.

4.3.4 Less Users, Documents and Supports for 3D Mapping

Unity is designed for game development where most of the manuals, tutorials and sample projects are associated with the gaming specifications and developments. The game engine can be considered as very specialized tool for 3D visualization of spatial objects. It has various limitations and other drawbacks, thus very limited users, documents and supports.

4.4 3D Game Engine for 3D GIS Visualization

This section describes the potential of 3D Unity visualization and game engine for 3D GIS visualization. There are several visualization packages meant for GIS, however most of those packages have very limited functionality for 3D GIS modelling and visualization, e.g. ArcScene, Bentley Map, MapInfo Discover 3D, AutoCAD Map 3D as reported during the workshop (TU Delft, 2011) as well as by Abdul Rahman (2016). The development of 3D visualization from those packages is still in progress and working toward solutions for 'real' 3D GIS visualization requirements.

3D spatial database component is quite essential in any spatial information system. We anticipate those two projects could be extended for a bigger scope in 3D spatial database component, i.e. providing answers for spatial and non-spatial queries of 3D object. We also recommend a transformation of 3D objects (CityGML) into GIS-based layers where more spatial or/and spatio analysis could be carried out on the layers.

All these layers from two different projects (solar energy estimation and underground utility) would be useful for future GIS-based smart city – planning and management. Later, new 3D layers or objects could be added or integrated to provide multi-functional applications within a single smart city model, in this case - 3D Unity.

5. CONCLUSIONS

We have described the 3D modelling for estimating potential solar energy on buildings surfaces and underground utility mapping using Unity 3D game engine platform for Istanbul. These applications provide precise measurements (e.g. solar energy estimation and location of underground utilities), analysis and better visualization especially for city planners.

Some advantages and limitations of the game engine also highlighted. The Unity 3D game engine itself is the most high quality game engine available in the market, easy to use and shorter rendering duration are mostly suite for game developers and companies. We strongly believe this 3D game engine could be extended for better visualization, format interoperability, database and other spatial related functions or applications in the near future.

REFERENCES

Abdul Rahman, A., (2016). Recent Research Works on 3D GIS in Malaysia. Keynote paper, International Conference on Geomatic and Geospatial Technology (GGT) 2016. 5 – 6 October 2016. Kuala Lumpur, Malaysia.

Alam N., V. Coors, and S. Zlatanova, 2013. Detecting shadow for direct radiation using CityGML models for photovoltaic potentiality analysis. UDMS Annual 2013. Publisher: Taylor & Francis Group, London, UK, Editors: Claire Ellul, Sisi Zlatanova, Massimo Rumor, Robert Laurini, pp.191-210

Bae J.H and A.H. Kim, 2014. Design and Development of Unity 3D Game Engine-Based Smart SNG (Social Network Game). International Journal of Multimedia and Ubiquitous Engineering. Vol. 9, No. 8 (2014), p.p 261-266.

Biljecki, F., J. Stoter, 2013. The Concept of LoD in 3D city models. In: Proceedings of Workshop CityGML in national mapping, 21- 22 January, 2013, Paris, France).

Buyuksalih G., S. Bayburt, A.P. Baskaraca, Hairi Karim and Alias Abdul Rahman. 2017. International Geomatic and Geoinformatic Technology, ISPRS Achieve. 03-04 October 2017, Kuala Lumpur. [In Press].

Good C.S., G. Lobaccaro, and S. Hårklau, 2014. Optimization of solar energy potential for buildings in urban areas – a Norwegian case study. Renewable Energy Research Conference, RERC 2014. ScienceDirect, Energy Procedia 58 (2014) 166 – 171.

Kolbe T., B. Burger., and B. Cantzler. 2015. CityGML goes to Broadway. In: D. Fritsch. Photogrametric Week'15. Wichmann, Stuttgart, Germany. Pp 343-355.

Kolbe T. and Bacharach S., 2006. CityGML: An Open Standard for 3D City Models. In Direction Magazine (Online). http://www.directionsmag.com/entry/citygml-an-open-standard-for-3d-city-models/123103 [Cited: June 15, 2017].

Kurakula V., Kuffer, M., 2008. 3D Noise Modeling for Urban Environmental Planning and Management. Real Corp 2008 Proceedings, Vienna.

Moser J., F. Albrecht and B. Kosar, 2010. Beyond Visualisation – 3D Gis Analyses for Virtual City Models, ISPRS Archives, Volume XXXVIII-4/W15. 5th International 3D GeoInfo Conference, November 3-4, 2010, Berlin, Germany.

Ruzinoor C. M., M. Shariff A. R., A. N. Zulkipli., M. Rahim M. Shafry, and M. H. Mahayudin, 2014. Using game engine for 3D terrain visualisation of GIS data: A review. 7th IGRSM International Remote Sensing & GIS Conference and Exhibition, Kuala Lumpur. IOP Conf. Series: Earth and Environmental Science 20 (2014) 012037 doi:10.1088/1755-1315/20/1/012037

Šúri M. and J. Hofierka, 2004. A New GIS - based Solar Radiation Model and Its Application to Photovoltaic Assessments, Transactions in GIS, vol. 8, no. 2, pp.175 –190, 2004.

TU Delft, 2011. 2nd International Workshop on 3D Cadastres. Organized by FIG, EuroSDR and TU Delft, 16-18 November 2011, Delft, The Netherlands.

Unity, 2017. Unity Documentation - 2D or 3D projects. https://docs.unity3d.com/ [Cited: June 10, 2017].

Wang Y., S. Schultz, F. Giuffrida, 2008. Pictometry's Proprietary Airborne Digital Imaging System and Its Application in 3D City Modelling. Commission I, ThS-2, ISPRS Vol. XXXVII. Part B1. Beijing, 2008.

Wieland M., A. Nichersu, S. M. Murshed, and J. Wendel. 2015. Computing Solar Radiation on CityGML Building Data. AGILE 2015 – Lisbon, June 9-12, 2015

3D GAME DEVELOPMENT USING UNITY GAME ENGINE

Pa.Megha [1] L.Nachammai [1],T.M.Senthil Ganesan[2]

U.G Students, Department Of Computer Engineering, Velammal College Of Engineering And Technology, Madurai, India[1]

Associate professor, Department Of Computer Engineering, Velammal College Of Engineering And Technology, Madurai, India[2]

***Abstract*: In this paper, we present the design and implementation of the game called Cognitive Arenas. It is a very common game and was developed keeping the Mac OS X and Windows Operating Systems in mind. The aim behind the project was to promote education via innovation. The students playing our game can learn their given material in one of our modules and answer the questions asked in the game where simultaneously he play the shooter game. The player can move to three levels of scenes depicted in Unity3D Game Engine and 3ds Max to play the game interestingly.**

Keywords: Unity Game Engine, 3ds Max, Cognitive Arenas, Education

1. INTRODUCTION

Game development has always been a controversial academic topic. The computer gaming industry has grown by leaps and bounds, becoming a mainstream software development sector, and earning billions of dollars in revenue each year. The gaming industry has also been one of the main driving forces behind the development of advanced modern hardware such as multi-core, hyperthreaded processors, high-performance graphics processing units (GPUs), advanced sound processing devices, and extraordinary human-computer interface devices such as virtual reality helmets and brain sensor caps. Computer games are like a book, a movie, or a museum.

In this paper we present our experience in creating a game based on the Unity game engine and promoting education via innovation. We want to use this more visual, more easily accepted medium to depict the world through which we can educate the people. Our game is a shooter game which also includes quiz based on the materials provided for learning. Shooter games are a subgenre of action game, which often test player's speed and reaction time. It includes many subgenres that have the commonality of focusing on the actions of the avatar using some sort of weapons.

Unity3D Game Engine is an integrated development tool used to develop interactive contents like video games, architectural visualization and real-time 3D animations. Its editor runs on Windows and Mac OS X platforms. though it runs on only two platforms, it has the ability of developing applications for multiple platforms which are mentioned as follows – Windows, Windows Phone, Mac OS X, iOS, Linux, Android, Web Player, etc. All we need for it to develop the application on respective platforms is the software development kit (SDK) for it.

2. RELATED STUDIES

2.1 Introduction to Unity3D

The Unity game engine was introduced by Unity Technologies in 2005, and has since become one of the most popular platforms for developing 2D and 3D games. It has been embraced both by small, third party developers and by large commercial game development companies. The functions thet are supported by Unity3D are very abundant. Unity3D produces the applications based on JavaScript and/or C#. These are used to assign the animation or real-time transition of the Game-Objects defined in the application. GUI of Unity3D helps a new developer to approach easily and script and program the transition of the Game-Object. The latest version, Unity 5, is a fully functional game engine, with all advanced features enabled and freely available to developers.

Unity is an integrated environment, which combines a number of sophisticated components such as the PhysX physics engine, the Mechanim animation system, a self-contained terrain editor and much more. It is also seamlessly integrated with the Monodevelop code editor, so that any changes made in Monodevelop are transparently compiled by Unity's C# or Javascript compilers, and inserted into game. Compilation errors are displayedin the Unity console window.

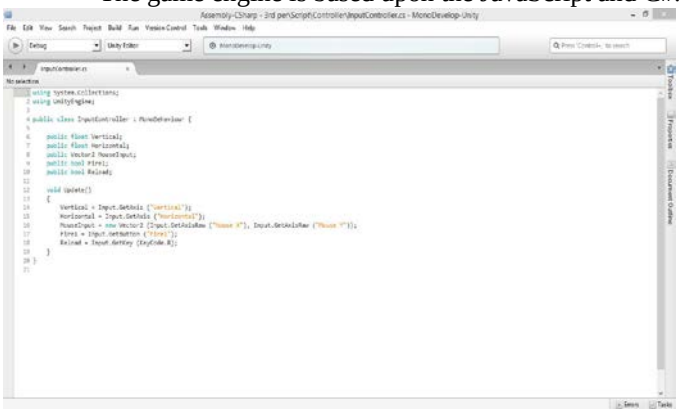
>> Rendering:

Unity supports art assets and file formats from 3ds Max, Maya, Softimage, Blender, Modo, ZBrush, Cinema 4D, Cheetah 3D, Adobe Photoshop, Adobe Fireworks and Algorithmic Substance. These assets can be added to the game projct, and managed through Unity's graphical user interface. Unity also has built-in support for Nvidia's (formerly Ageia's) PhysX physics engine, (as of Unity 3.0) with added support for real-time cloth simulation on arbitrary and skinned meshes, thick ray casts and collision layers. 3D objects in made in 3dsMax, can be saved as .max files directly into the project or export them into Unity using the Autodesk .FBX or other generic formats. Unity imports meshes

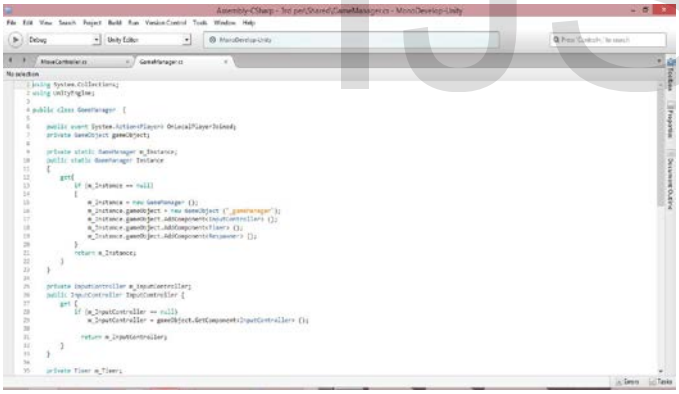
from 3ds Max. Export 3D file formats such as .fbx or .obj from 3D modeling software in generic formats that can be imported and edited by a wide variety of different software.

>> Scripting:

The game engine is based upon the JavaScript and C#.



Unity's scripting is built on Mono, an open-source implementation of .NET Framework. MonoDevelop is an open-source IDE for Linux, MacOS X and Windows. It supports Boo, C, C++, C#, CIL, D, F#, Java, Oxygene, Python, Vala and VB .NET. Scripts are used to respond to input from the player and arrange events in the gameplay to happen when they should. They can also be used to create graphical effects, control the physical behavior of objects or even implement a custom AI system for characters in the game.



>> Asset Tracking:

Unity also includes the Unity Asset Server – a version control solution for the developer's game assets and scripts. It uses PostgreSQL as a backend, an audio system built on the FMOD library (with ability to playback Ogg Vorbis compressed audio), video playback using the Theora codec, a terrain and vegetation engine (which supports tree bill boarding, Occlusion Culling with Umbra), built-in light mapping and global illumination with Beast, multiplayer networking using RakNet, and built-in pathfying navigation meshes.

>> Platforms:

Unity supports deployment to multiple platforms. Within a project, developers control over delivery to mobile devices, web browsers, desktops and consoles. Unity also allows specification of texture compression and resolution settings for each platform the game supports.

Currently supported platforms include for Windows, Mac, Linux/Steam OS, Unity Web player, Android, Ios, Blakberry 10, Windows Phone 8, Tizen, Windows Store apps, WebGL, PlayStation 3, PlayStation 4, PlayStation Vita, Wii U. Xbox One, Xbox 360, Android TV, Samsung Smart TV, Oculus Rift and Gear VR.

2.2 Autodesk 3ds Max

Autodesk 3ds Max is a professional 3D computer graphics program for making 3D animations, models, games and images. It is developed and produced by Autodesk Media and Entertainment. It is frequently used by video game developers, many TV commercial studios and aechitectural visualization studios. It is also used for movie effects and movie pre-visualization. In addition to its modeling and animation tools, the latest version of 3ds Max also features shaders (such as ambient occlusion and subsurface scattering), dynamic simulation, particle systems, radiosity, normal map creation and rendering, global illumination, a customizable user interface, and its own scripting language.

Modeler and texture artist could increase the speed of working and efficiency because of extensive polygon modeling and texture mapping tool set. CAT (Character Animation Toolkit) that is completely integrated provides advanced rigging and animation system that can be used immediately. Artist could create very realistic or great image by using mental ray renderer and integrated 3ds Max scan line including mental ray network that can finish rendering faster. Since it supports C++ and .NET, model with Autodesk 3ds Max, send it to Autodesk FBX exchange technology file after animating it and then bringing it to Unreal Editor Contents production software.

2.3 Importing from 3ds Max to Unity

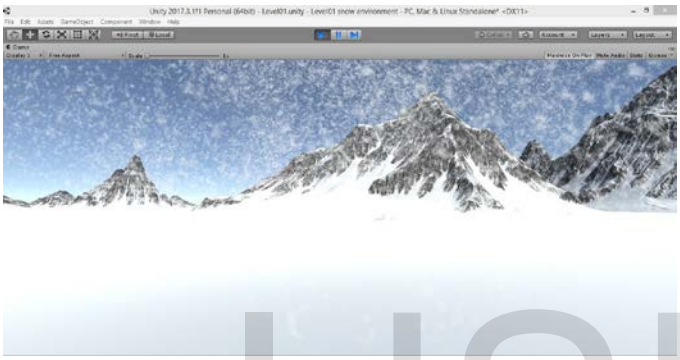
The things that are imported from 3DS MAX from Unity3D can be largely divided into five parts. First, it is every node that has position, rotation, scale. Second part includes pivot point, names, vertex and meshes with colors. Third, they are normal and one or two UV sets. Fourth, they are diffuse texture and materials with colors. Fifth, they are many materials per mesh, animation, and bone based animations. Tus, these five parts are the ones that Unity recognizes from 3DS MAX. Also, Unity3D could import FBX file. FBX provides supports for software and applications that are both independent and from certain companies. FBX is used within e tertainment pipeline or as a part of design production. Files can be sent more smoothly, more data is maintained, and the work flow is more efficient. To send a file from 3DS MAX to Unity3D, first convert to FBX file with export from 3DS MAX, and send it. Then the file can be imported from Unity3D. Only, if the name of FBX file is in Korean, Unity3D cannot recognize it.

2.4 Summary of the Game Development

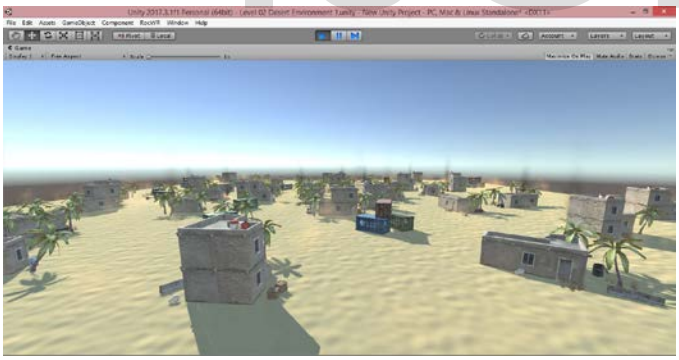
In this research paper

1) **Game Name:** Cognitive Arenas
2) **Game Genre:** Quiz, Shooter
3) **Platform:** Supports Multi-platform (Wiindows PC, Android, etc.)
4) **Game Characteristics:** Our game has three levels for the player to travel through. The various scenes depicted in those levels are

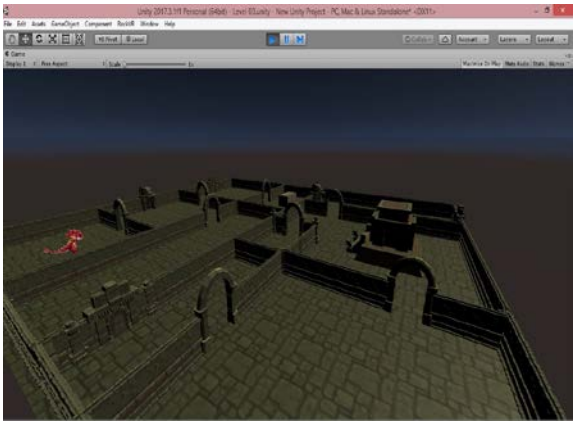
Level 1>> The Snowies:
This level contains snowy mountains and sereny place for the player to travel.



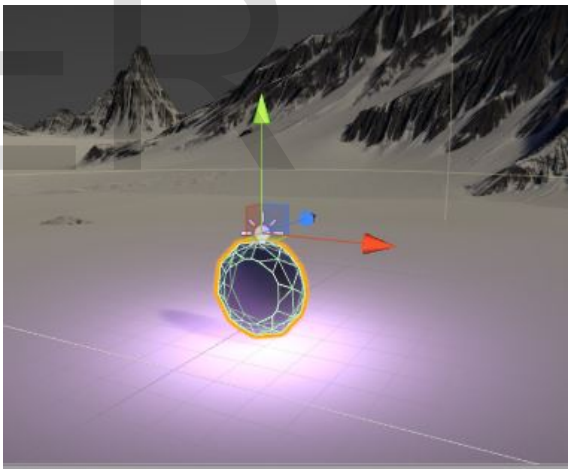
Level 2>> Deserted Area:
This level scene looks like a desert that also contains sand houses.



Level 3>> Mayan area:
The building in this level looks like Mayan buildings which will be in old-fashioned style.



The player will travel through all these levels to complete the game. The player can learn the material given in the game for answering the questions. The learning material will be available in the LEARN module of the game. The rules and controls of the game will be given in the modules of the game for the player to follow. The PLAY module of the game will direct the player to the game scenes to play. The player has to answer the questions asked in the game by exploring the answers from the given environment and has to undergo offending and defending activities agains the enemies in the game.



The object that helps the player to score points.

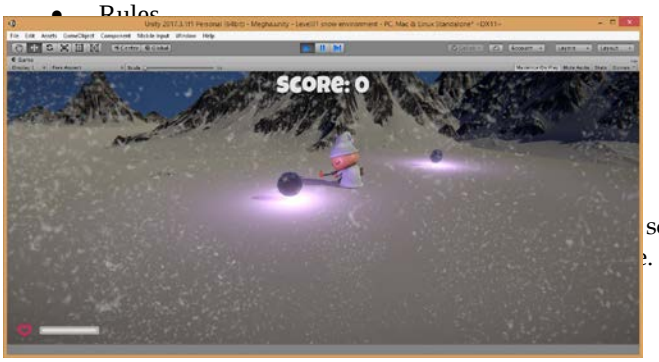
5) **Core Idea of the Game:**
The objective of our game is to promote education via innovation. We want to use this more visual, more easily accepted medium to depict the world through which we can educate the people. Our game is not just entertainment, it's a channel that is responsible for promoting education. It gives the feel of intangible treasure through the environment provided. We found there is a need of a medium that will provide education through entertainment. And this may be a solution for lack of interest as well as getting bored in learning.

3 MODULES

3.1 Game UI screen

>>Modules:

- Play
- Settings
- Rules



scene
. The

The player in the first level.

>>Settings:

This module can be used to modify the volume for the game.

>>Rules:

This module shows the rules and controls required for the movement and exploration by the player in the game and constraints need to be followed by the player.

>>Learn:

This module provides the materials published for the game to answer the given questions. The materials given can be read by the player and use those information to answer the questions asked in the game.

>>Quit:

This module is used to close the game window that is to exit from the game application.

4 CONCLUSION

This study aims to design and develop a gaming application using Unity Game Engine. Through this cornerstone for a new concentration on Game Development, we have conveyed our views and ideas on promoting education via innovation. The functions that Unity3D supports autonomously are very abundant. All game developments are possible such as shader, physics engine, network, terrain manipulation, audio, video, and animation, and it is enabled so that it is possible to revise, meeting demand of user according to the need.

REFERENCES

- http://seriousgamesnet.eu/mod/elgg_segan_framework/static/conferences/2013/presentations/10_Game_Development_Using_Unity_David_Gouveia.pdf
- http://ds.nashobmen.org/fo/get/2655976/Packt_Unity_Game_Development_Essentials_2009-wapdisk_ru.pdf
- http://trcf52.okstate.edu/GPU/u3d-tut-1.pdf
- https://www.theseus.fi/bitstream/handle/10024/68068/Dansie_Jason.pdf?sequence=1&isAllowed=y
- https://gamifique.files.wordpress.com/2011/11/5-game-design-theory-and-practice.pdf
- https://en.wikipedia.org/wiki/Subnautica
- https://knowledge.autodesk.com/support/3ds-max/getting-started/caas/CloudHelp/cloudhelp/2017/ENU/3DSMax-Tutorial/files/GUID-C99064E7-7E14-4F98-9A41-796BFC5613EF-htm.html

International Journal of Advanced Science and Technology
Vol. 29, No. 7s, (2020), pp. 1821-1830

A REVIEW ON DEVELOPMENT OF 3D ADVENTUROUS SERIOUS GAME: THE SEASONAL RUN

Nithiyaa Muniandy
Politeknik Metro Kuala Lumpur, Malaysia

Sathya Manoharan
Politeknik Metro Kuala Lumpur, Malaysia

Kohilah Miundy
Politeknik Metro Kuala Lumpur, Malaysia

Abstract

Gaming industry is a boosting phenomenon and the focus of games has grown from entertainment to serious learning. Previously, tremendous amount of games were produced with the focus on adventure, action-adventure, role-play, racing, multiplayer, casual play and many more. However, current trend in gaming world has transformed the action or adventure game to an immersive educational game, which include pedagogical aspect to enable learning process during the play. This paper discusses on the development of three-dimensional (3D) adventurous serious game called Seasonal Run using Unity Game Engine by a group of students. The game was developed based on phases available on the traditional Game Development Life Cycle (GDLC) namely, initiation, pre-production, production, alpha testing and beta testing and finally to release but each development phases has been refined with detailed context to simplify the game production. This guideline method able to produce a wholesome game with interesting features of adventure game such as exciting storyline, fun and realism, unceasing challenge, immediate and useful rewards, target to achieve, re-playability and also include some serious game elements such as engaged learning, knowledge through experience and problem solving skills. After the deployment, a prototype framework was developed to differentiate educational and entertainment module.

Keywords— *Adventure game, Serious game, Game Development Life Cycle (GDLC).*

I. INTRODUCTION

Gaming industry is rapid growing field in 21st century (Krotoski, (2004). Game is vastly used in many applications because it enhance experiences (Luimula & Trygg, 2016) and provide high degree of re-playability (Medendorp & Semwal, 2018). Traditional games were took over by computer games (Connolly et al., 2012) as a leisure activity among teenagers (Aleem et al., 2016) within 40 years of period. Everyone plays game since computer games are played globally in various languages (Ahmad et al., 2014). Many reports specify that playing digital games promotes aggressive thoughts, physiological arousal (Anderson, 2010), inappropriate time management (Ogletree & Drake, 2007), addiction (Griffiths & Davies, 2002) loneliness and anti-social (Merhi et al., 2007) especially when playing violent entertainment. Conversely, (Subrahmanyam & Greenfield, 1994) state that digital games also can develop valuable skills and provides an attractive and beneficial approach on learning if it is emphasise in positive way (Connolly et al., 2012). Hence, it is crucial to focus on educational entertainment games rather than violence-based entertainment games. Nowadays, games assimilated into education to establish a pioneering educational prototype (Tan et al., 2007). The process of calibrating educational element in an entertainment construct a new term called edutainment game (Muda & Basiron, 2005) widely known as serious game (Perez-Colado et al., 2017).

ISSN: 2005-4238 IJAST

II. SERIOUS GAME CONCEPT AND APPROACH

Serious games are games that implanted pedagogical and entertainment aspects equally in digital games (Ahmad et al., 2014). A great extend emphasis on pedagogical element will lead to a course material which has very minimal engagement, whereas too little emphasis on pedagogy will lead to full entertainment with controlled learning (Perez-Colado et al., 2017). Thus, serious games should include both education and entertainment aspects to enhance the learning process through gaming (Ahmad et al., 2014). Therefore, development process of serious game is not only in the hands of developer and designers but also incorporate with educators, pedagogues, psychologist and also students (Ávila-Pesántez, 2017). A serious game design process must include a number of processes before it make up to be a complete game. The process includes storyboarding, analysis, design, animation refinement, video production, scenarios, sound, technological and functional requirement, programming, testing and evaluation (Aleem et al., 2016). Serious game convey the learning message efficiently since it provides immersive learning experiences through the game play (Avila-Pesántez et al., 2017). Educational games generally drives motivation (Papadimitriou & Virvou, 2018), (Amani & Yuly, 2019) of a player to learn and adapt with the message given in the game very easily. When serious game elements embedded in the adventurous game, the learning process becomes very fast and efficient.

III. INSIGHT TO ADVENTURE GAMES

Adventure game is a type of video game in the form of exploration or puzzle-solving to proceed to next level where the player plays the role of hero or leading character to complete the mission of the game (Papadimitriou & Virvou, 2018). Adventure games are the type of games that have powerful narration (Dickey, 2006), (Lin et al., 2018). Adventure games are designed for single player since the primary focus of the genre is the story and the character. Adventure game is very popular entertainment among all the children, youth and even adults. The evolution of adventure game starts in the year of 1970s to date from simple two dimensional (2D) to a challenging three dimensional (3D), afterwards to Augmented Reality and Virtual adventure games (Luimula & Trygg, 2016). Evolution of technology and the appealing game presentation are the prime factors for many youngsters to play the adventure games nowadays. Every day the release of adventure games in the google play or online platform keep rising (Muda & Basiron, 2005).

IV. OBJECTIVE AND SCOPE

The objective of this study is to provide an expanded genre of Game Development Life Cycle (GDLC) model with detail description to ease the development of 3D adventurous game from the scratch. Thus, to provide a framework model of game prototype to differentiate learning and entertainment module in the game. Further, the new GDLC would help other game developers to follow as guideline to create or develop a game and also include a framework model to clearly see the edutainment approach to make the learning fun, alive and active. The framework model is design in assisting children learning season from five (5) years to twelve (12) years old in the communication medium is in English.

V. DEVELOPMENT METHODOLOGY

The development methodology for this research paper has been divided into four sections. First section discusses about the summary of Seasonal Run game. Generally, this section discusses on the background setting of the game, role-playing character, the mission of the game, intended user and the objective of the game. Second section talks about the conceptual model used to develop Seasonal Run game. This section gives a very brief detail about each phases in a modified GDLC and how it is used to develop Seasonal Run game. Third part is about a framework model which was developed to show the interconnections between menu, learning and knowledge modules of Seasonal Run game. The final section is speaking about prototype interface and design. This section discusses on the timeframe to complete Seasonal Run game, includes few examples of interface design and motivating factors for the re-playability of the game.

ISSN: 2005-4238 IJAST

A. Overview of Seasonal Run

Seasonal Run is an educational adventure game which is also considered as adventurous serious game created by students of Metro Polytechnic Kuala Lumpur, Malaysia. Seasonal Run focuses on the people who loves exploration. Basically, this game is about a boy, Rudy who explores four seasons on earth in his dream. The world season includes summer, winter, autumn and spring. He learns about four seasons on earth in his school. After coming back from school he brought his learning mind to bed and dreamt about four seasons he have learnt in school. He really wants to experience all the four seasons on earth and would like to enjoy his stay in all the four seasons. Hence this game is about Rudy's dream where he explore and have entertainment in all these four seasons as well as get knowledge all through his exploration.

So the player cast the role of Rudy in this game. The player needs to unlock all the levels in order to help Rudy to get out from his dream. There are four levels in this game from level 1 up to level 4. All the levels give various seasonal experiences with different obstacles at each level. Rudy's mission in this game is to complete his exploration in all the four seasons by collecting items along the path based on instructions. Once he complete each level, he will be awarded badges based on the finishing time. Besides that, this game has embedded educational values, where at the end of each level the player will get exposure about the suitable attires to wear during the particular seasons.

Basically this game is an educational game to give early exposure for the pre-school kids regarding four seasons and the suitable attires to wear throughout the four seasons. There are four levels or modules in this game. Module 1 is about spring, module 2 is about summer, module 3 is about autumn and the final module is about winter. After each mission the kids or the player will have a short quiz before proceed to next module. In each module the player need to collect the attires to wear during particular season. Once complete with the module, the player need to recall the attires during gameplay to answer a short quiz. If the player manage to pick all the related attire then the character will appeal to the player with the same attire he collected as in Figure 1. A selection on the right answer in the quiz will unlock their play for the next level.

Figure 1: Character appeal in various seasons

The objective of this game is to create an exploratory game that implement all the four seasons on earth to teach various seasons, to promote educational values in adventurous game for the children, to entertain the children with fun learning and to develop an adventurous game which is playable without boring feeling.

B. Conceptual Model

Seasonal Run was successfully implemented with the guidance of a conceptual model which arise from existing Game Development Life Cycle (GDLC) model. Some modifications has been done based on the need of edutainment digital game. In this model there are six phases namely, initiation, pre-production, production, alpha testing, beta testing, and final release (Ramadan & Widyani, 2013), (Amani & Yuly, 2019) as shown in Figure 2 to complete the production of game. Students of Metro Polytechnic Kuala Lumpur, Malaysia use this conceptual model to complete Seasonal Run game.

ISSN: 2005-4238 IJAST

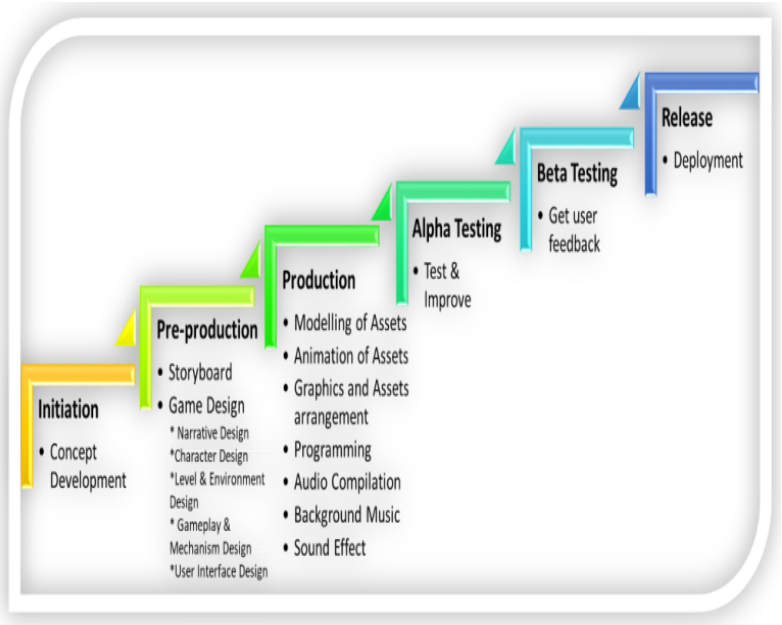


Figure 2: Modified Game Development Life Cycle (GDLC) model

1. Initiation

The production of Seasonal Run starts with the minute concept. Metro Polytechnic Kuala Lumpur, Malaysia students are still considered as freshman in exploring games. Creating 3D games from the scratch is not an easy task. After the brainstorming session, students come up with a concept of creating an adventurous game. Students' primary motive is to produce an exploratory adventurous game. Yet, a game for the solitary purpose of entertainment is not going to benefit the player or mankind. Thus, students implanted a concept of education in the adventurous game.

They picked a small topic from pre-school syllabus where kids learn different seasons on earth. Students of Polytechnic Metro Kuala Lumpur, Malaysia think of it as a good idea and took the challenge to create a concept to introduce seasons for the pre-school students in the form of game to give early exposure about four different seasons exists on earth since Malaysia only experience two monsoon seasons, namely Northeast Monsoon and Southwest Monsoon. Automatically, they have set their target players to pre-school students. Yet, this game do not restrict others from playing this game. While thinking of how to connect the idea of seasons they come up with an idea of introducing attires that links particular season, and thence organize their script to bring a beautiful game flow.

2. Pre-Production

Next step is, pre-production. This is an important stage in game development. Pre-production is the first and foremost step of production phase and the initial seed for the final output of a game. Pre-production works on defining production pipeline. During pre-production phase students identify and gather sufficient amount of information, tools and software required to develop a game. Then, jobs were segregated among members such as making storyboard and designing game. Storyboarding is a sequence of sketches of game scenes with appropriate camera view types and descriptions about every scene. Basically, the storyboard tells us the story of a game in the form of drawing. Figure 3 shows the storyboard of Seasonal Run.

ISSN: 2005-4238 IJAST

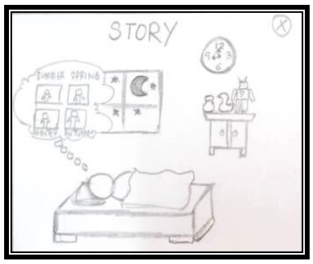
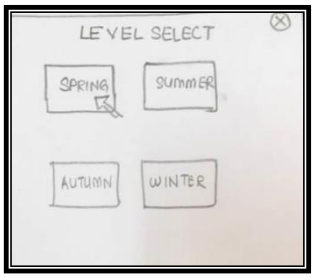
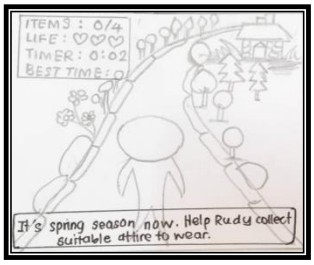
	
Scene: 1 **Camera type:** Fixed camera system **Description:** Interface of game to provide options to play or exit.	**Scene:** 2 **Camera type:** Fixed Camera System **Description**: Rudy is dreaming of exploring all the four seasons while sleeping on the bed in his room at night.
	
Scene: 3 **Camera type:** Fixed Camera System **Description:** Displaying all four seasons as image type. First level is active and the rest are locked till mission is complete in first level.	**Scene:** 4 **Camera type:** Tracking camera system **Description:** Along this scene, character will run and collect the items along his path. Camera will follow the character's movement.

Figure 3: Some Storyboard design for Seasonal Run

However, game design is the mould for final output of a game. It has five primary elements to be measured before proceed to production phase. Game design elements involves narrative design, character design, level or environment design, gameplay and mechanic design and also user interface design (Ramadan & Widyani, 2013). During narrative design, the game plot and the mission of the game will be delineated. Once the plot is outlined, the next step is to design characters that play a role in the game. There are two types of characters when comes to gaming world, main character(s) and non-player character(s) or also known as computer-controlled character(s). The developer should sketch both player and non-player characters during character design. Once the plot is ready with characters, the subsequent step will be level and environment design. During this phase, the developer will decide suitable background for each scene, amount of lighting requires to create dynamic environment, perfect camera angle, appropriate weather for particular level and prop needed in each level. Next is the gameplay and mechanics design.

ISSN: 2005-4238 IJAST

While designing game mechanics, significant things such as challenge in the game, activity for the player, rewards for every successful attempt, game progression in level increment, game rules to be followed and the required skills to win the game will be outlined to make the gameplay durable. The last thing to consider in game design is user interface design. During interface design the developer has to take into consideration on the control of input keys. Input keys varies for different platforms, for instance, console uses controller, mobile uses touch, computer uses mouse or keyboard input and virtual reality applications uses headset. Apart from that, output devices to view the game should also have to define in the design phase. The last one is viewpoint. In user interface design it is important to state the game view whether it is perspective, side view or top view. All sort of designs mentioned above should be documented systematically in Game Development Document (GDD) at the end of pre-production phase for further revision.

Seasonal Run has undergone all these game designs before enters production phase. Game setting has been set to four different seasons with the mission of collecting suitable attire to wear during particular season. Main character in the game is Rudy, a small boy whereas there are some non-player characters which moves back-and-forth to restrict Rudy from moving forward along his path. The user interface of Seasonal Run is set with a perspective view and intent to deploy in personal computer which accepts keyboard and mouse input. This game also has rules to follow, challenges to move to next level, rewards after completion of a level and many more.

3. Production

Next step is production. Production phase uses the GDD which was developed during pre-production to transform the game designs to realism. Modelling of characters and assets, making animation of assets, graphics arrangements, physics simulation, scenes development, programming or implementation, audio-compilation, background music, sound effect and interactivity is done during this production phase.

Modelling of game objects for Seasonal Run was done in Autodesk Maya. Game objects are fundamental elements which create entire game (Raguman et al., 2019). The main character, Rudy was initially sketched and developed using Autodesk Maya. Then texturing has been done for the Rudy character using Adobe Photoshop as shown in Figure 4. Environment for each scene also developed using Autodesk Maya and Unity Game Engine was used to create some terrain functions like trees and hills as shown in Figure 5. The character and entire environment was developed using low poly model to reduce rendering time and to make the game less computational-intensive.

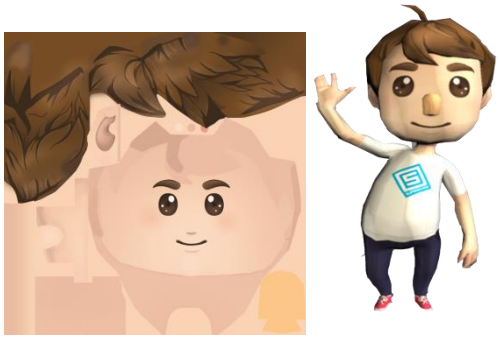


Figure 4: Texture and prototype of Rudy character

ISSN: 2005-4238 IJAST

Figure 5: Low poly assets and obstacles used in Seasonal Run scenes

Then, the arrangement of assets is done in Unity Game Engine according to the camera view. Barriers and colliders were applied to the environments to prevent the character passing through the assets. Then, the level design which involves various seasonal environment was set up. Separate scenes were developed for each season, spring has a greenery effect with many colourful flowers, summer is about sunny days, autumn season shows fall of flowers and the last phase is winter to show the cold weather. In each season, player will collect suitable attires for the particular season. Indication panel is given to show the number of items to collect in the particular scene, number of lives left to complete the level, player's completion time and the best time from total number of play. After exploring each level scene, the player have to take a quiz to answer and the achievement will be rewarded with stars. The stars visibility is based on the number of collectable items within a particular timing. Programming has been applied to activate the game mechanics and to make interaction between game assets as well as to combine all game assets according to game need. Finally, the individual scenes rendered with suitable background music. The production phase takes a long process to make as a complete game with all game mechanics working.

4. Testing

After the production, the game will go for alpha testing for quality assurance. During this phase, the game will be tested at developer's site where it involves in-house developers, quality assurance team and customers. Alpha testing is done to check for defects, bugs, deficiencies, errors and incompatibilities. If the primary customer is unsatisfied with the product, the product will undergo the pre-production stage again to revamp the unsatisfactory elements in the game. Then, the game will reiterate production phase to make it complete and perfect for beta testing and thereafter to release to the public. Seasonal Run was tested among few pre-school teachers and also development team. Some improvement were made such as changing background music for the game and reconnecting the finishing storyline with initial storyline. Changes were made at the end of the alpha testing.

5. Beta

Second revision or known as beta testing is conducted by the group of real end users after the alpha testing to get end user's feedback and ensure readiness of product for the release. During this phase ideas will be thrown to enhance usability, compatibility and functionality.

6. Release

After undergoing all the testing phases, now it is time to release the game. A complete game is sent to the publisher to launch and sell in the market to get revenue. Presently, gaming is used in classrooms to convey contents (Kafai et al., 2016) to students in an effective way to improve their intellectuality (Muda, 2006). Seasonal Run was sold to the pre-school teachers as their teaching-kit to demonstrate the season topic at school.

C. Framework Model

After Seasonal Run has been deployed, a framework model was designed to show overall flow of Seasonal Run game as shown in Figure 6. The framework model has been categorized into three major components (Muda et al., 2004) called Human Computer Interaction (HCI), Learning Module and Knowledge-Based. HCI represented by menu component, Learning Module as learning and Knowledge-Based as record. Menu module contains the functional keys to play, game information

ISSN: 2005-4238 IJAST

and exit. Learning module starts with the story of the Seasonal Run. Later, the story leads to game scenes and quizzes after every gameplay scene and finally to end story. Record module holds the score of the player, achievement and best time for the each scene in the game. All the modules are interrelated to one another.

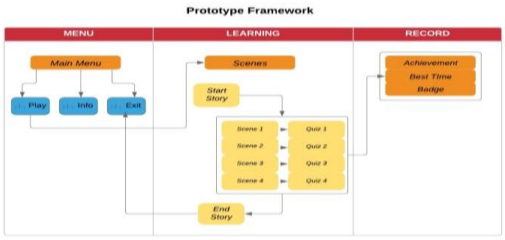


Figure 6: Framework Model of Seasonal Run

Figure 7: Sample of interface designs of Seasonal Run game

D. Prototype interface and design

The prototype design of Seasonal Run approximately devour a month to complete. Fig 7 shows the examples of Seasonal Run interface design in various seasons. All scenes are installed with lively 3D assets and vibrant graphics, catchy audios and fantastic animations. Various triggering functions with suitable sound effect also applied to handle interactivity when the character hit with obstacles and pick up the collectable items. Clear display of total lives, indication of items collected for the entire scene, well-noticeable text, simple key usage to move the character along the scene, coherent flow of story, refreshing quiz after every gameplay are truly some motivating factors for the player to drive their interest to play the game repeatedly.

VI. CONCLUSION AND RECOMMENDATION

In conclusion, this paper gives a review on the production of adventurous serious game, Seasonal Run from the scratch by using a conceptual model which is derived from GDLC model with six phases called initiation, pre-production, production, alpha-testing, beta-testing and release. The designed conceptual model is believed to produce a beneficial end products to the end users. Successful implementation of Seasonal Run game using the conceptual model verifies this statement. We recommend the usage of this conceptual model in the future as a reference to produce more adventurous serious games to produce valuable end product to the mankind. A recommendation is also given to deploy the end product, Seasonal Run game in Virtual Reality platform to give real time experience in various seasons. In addition, Seasonal Run can outspread the scope to learn the countries experiencing particular season by bringing alive the real environment to give real-time experience for the end users.

ISSN: 2005-4238 IJAST

REFERENCES

[1] Ahmad, M., Rahim, L. A., & Arshad, N. I. (2014, June). A review of educational games design frameworks: An analysis from software engineering. In 2014 International Conference on Computer and Information Sciences (ICCOINS) (pp. 1-6). IEEE.

[2] Aleem, S., Capretz, L. F., & Ahmed, F. (2016). Game development software engineering process life cycle: a systematic review. Journal of Software Engineering Research and Development, 4(1), 6.

[3] Amani, N., & Yuly, A. R. (2019, April). 3D modeling and animating of characters in educational game. In Journal of Physics: Conference Series (Vol. 1193, No. 1, p. 012025). IOP Publishing.

[4] Anderson, C. A., Shibuya, A., Ihori, N., Swing, E. L., Bushman, B. J., Sakamoto, A., & Saleem, M. (2010). Violent video game effects on aggression, empathy, and prosocial behavior in Eastern and Western countries: A meta-analytic review. Psychological bulletin, 136(2), 151.

[5] Ávila-Pesántez, D., Rivera, L. A., & Alban, M. S. (2017). Approaches for Serious Game Design: A Systematic Literature Review. The ASEE Computers in Education (CoED) Journal, 8(3).

[6] Connolly, T. M., Boyle, E. A., MacArthur, E., Hainey, T., & Boyle, J. M. (2012). A systematic literature review of empirical evidence on computer games and serious games. Computers & education, 59(2), 661-686.

[7] Dickey, M. D. (2006). Game design narrative for learning: Appropriating adventure game design narrative devices and techniques for the design of interactive learning environments. Educational Technology Research and Development, 54(3), 245-263.

[8] Griffiths, M. D., & Davies, M. N. (2002). Research noteExcessive online computer gaming: implications for education. Journal of Computer Assisted Learning, 18(3), 379-380.

[9] Ch'en, H. C., Jiang, C. R., Lin, C. H., Sung, W. Y., & Liao, Y. C. (2016, July). An Adventure Game: Kido Senki Valhalla. In International Conference on Frontier Computing (pp. 999-1006). Springer, Singapore.

[10] Kafai, Y. B., & Burke, Q. (2016). Connected gaming: What making video games can teach us about learning and literacy. Mit Press.

[11] Krotoski, A. (2004). White Paper: Chicks and Joysticks: An Exploration of Women and Gaming. Entertainment & Leisure Software Publishers Association (ELSPA).

[12] Luimula, M., & Trygg, M. N. B. (2016, October). Cultural heritage in a pocket: Case study "Turku castle in your hand". In 2016 7th IEEE International Conference on Cognitive Infocommunications (CogInfoCom) (pp. 000055-000058). IEEE.

[13] Medendorp, A., & Semwal, S. K. (2018, November). Procedural 3D Tile Generation for Level Design. In Proceedings of the Future Technologies Conference (pp. 941-949). Springer, Cham.

[14] Merhi, O., Faugloire, E., Flanagan, M., & Stoffregen, T. A. (2007). Motion sickness, console video games, and head-mounted displays. Human factors, 49(5), 920-934.

[15] Muda, Z. (2006, April). Storytelling approach in multimedia courseware: An introduction to science for preschool education. In 2006 2nd International Conference on Information & Communication Technologies (Vol. 2, pp. 2991-2993). IEEE.

[16] Muda, Z., & Basiron, I. S. (2005, November). Multimedia adventure game as edutainment application. In International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06) (Vol. 2, pp. 497-500). IEEE.

[17] Muda, Z., Mat Noor, S.F., Ismail, A., Mohd Noah, S.A., Tengku Wook, T.S.M., Badioze Zaman, H., Mohd Ali, N. & Mohamad, N.H. (2004). Multimedia Approach in Education Courseware Architecture and Development. In L. Cantoni & C. McLoughlin (Eds.), Proceedings of ED-MEDIA 2004--World Conference on Educational Multimedia, Hypermedia & Telecommunications (pp. 1883-1887). Lugano, Switzerland: Association for the Advancement of Computing in Education (AACE).

ISSN: 2005-4238 IJAST

[18] Ogletree, S. M., & Drake, R. (2007). College students' video game participation and perceptions: Gender differences and implications. Sex Roles, 56(7-8), 537-542.

[19] Papadimitriou, S., & Virvou, M. (2017, August). Adaptivity in scenarios in an educational adventure game. In 2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA) (pp. 1-6). IEEE.

[20] Perez-Colado, I. J., Perez-Colado, V. M., Martínez-Ortiz, I., Freire-Moran, M., & Fernández-Manjón, B. (2017, April). UAdventure: The eAdventure reboot: Combining the experience of commercial gaming tools and tailored educational tools. In 2017 IEEE Global Engineering Education Conference (EDUCON) (pp. 1755-1762). IEEE.

[21] Raguman, R., Santhakumar, M., Thomas, X. P., & Revathi, M. (2019). 3D Adventure Game Using Unity. Bonfring International Journal of Software Engineering and Soft Computing, 9(2), 16-20.

[22] Ramadan, R., & Widyani, Y. (2013, September). Game development life cycle guidelines. In 2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS) (pp. 95-100). IEEE.

[23] Ramdan R and Widayani Y2013. Game Development Life Cycle Guidelines. ICACSIS2013. pp.95-100.

[24] Subrahmanyam, K., & Greenfield, P. M. (1994). Effect of video game practice on spatial skills in girls and boys. Journal of applied developmental psychology, 15(1), 13-32.

[25] Tan, P. H., Ling, S. W., & Ting, C. Y. (2007, September). Adaptive digital game-based learning framework. In Proceedings of the 2nd international conference on Digital interactive media in entertainment and arts (pp. 142-146). ACM.

ISSN: 2005-4238 IJAST

2014 IEEE 14th International Conference on Advanced Learning Technologies

A Survey of Frameworks and Game Engines for Serious Game Development

Brent Cowan and Bill Kapralos
Faculty of Business and Information Technology
University of Ontario Institute of Technology
Oshawa, Ontario, Canada. L1H 7K4.

e-mail: brent.cowan@uoit.ca bill.kapralos@uoit.ca

***Abstract*—Given the sparsity of standard game engines and frameworks for serious game development, developers of serious games typically rely on entertainment-based game development tools. However, given the large number of game engines and frameworks dedicated to entertainment game development, deciding on which tool to employ may be difficult. A literature review that examined the frameworks and game engines used to develop serious games was recently conducted. Here, a list of the most commonly identified frameworks and game engines and a summary of their features is provided. The results presented provide insight to those seeking tools to develop serious games.**

Keywords-serious games; game engine; framework; review;

I. INTRODUCTION

Given the ubiquity of video game use across a large demographic (i.e., male, female, from the very young to the very old), and the ability of video games to engage and motivate learners, the popularity of serious gaming (video games whose primary purpose is education and training) has seen a recent surge across all areas of education and training. Serious games "leverage the power of computer games to captivate and engage players for a specific purpose such as to develop new knowledge or skills" [1] and greater engagement within an educational setting leads to higher academic achievement [2] (greater details regarding the benefits of serious games are available by Squire [3] and Bergeron [4]). Despite the popularity of serious games and the benefits they afford, the development of effective serious games is a difficult and complicated process requiring an appropriate balance between game design and instructional design (a lack of proper instructional design can lead to an ineffective serious game) [5,6].

Further complicating matters, there are currently very few development tools (game engines and frameworks in particular), designed specifically for serious game development; serious game developers have generally relied on game engines and frameworks designed for entertainment games despite the inherent differences between serious game and entertainment game development [7]. More specifically, serious games are often produced by small teams in contrast to many of the commercial entertainment games where the development team can include dozens of developers.

There are many commercial game engines that provide advanced rendering technologies, and simple tools for content creation allowing game developers to reuse code thus decreasing development time and costs [8]. Sherrod defines a game engine as "a framework comprised of a collection of different tools, utilities, and interfaces that hide the low-level details of the various tasks that make up a video game" [9]. The terms game engine and framework are often used interchangeably. For the purpose of this paper, the term game engine refers to the functionality and features that become part of the completed game. A framework includes a game engine in addition to external tools and resources that simplify the process of game development.

Here, we present the results of a literature review that was conducted across three databases (*Google Scholar*, the *Institute of Electrical and Electronics Engineers* (IEEE) Xplore digital library, and the *Association for Computing Machinery* (ACM) digital library), to examine the tools commonly used to develop serious games, essentially compiling a list of tools used by serious game developers. This list will allow us to investigate the game engine and framework features that are most important to developers. This may prove to be useful to the developers seeking such tools.

II. BACKGROUND

A list of the primary features provided by the majority of most modern game engines is provided below. The code responsible for providing this functionality becomes a part of the finished game:

- **Scripting**: Simple code that can be written to control game objects and events.
- **Rendering**: The speed and accuracy that a 3D scene is generated, and the visual effects provided.
- **Animation**: The movement and deformation of objects such as characters.
- **Artificial Intelligence**: Steering behaviors such as pursuing, dodging, and fleeing, are combined with path finding.
- **Physics**: Objects respond accurately when collided with or in response to force or pressure applied to them.
- **Audio**: Spatial rendering of audio allows sounds to have a location in the environment. Filtering can add variation and environmental cues such as reverberation.
- **Networking**: Allows players to interact with other players within the game by sharing data through a network.

978-1-4799-4038-7/14 $31.00 © 2014 IEEE
DOI 10.1109/ICALT.2014.194

Game creation frameworks generally provide a graphical user interface (GUI) which often ties together several editors. Listed below are some of the editing tools commonly included with game creation frameworks:

- **Level Editor**: Also known as a world editor, this tool aids in the creation of virtual 2D or 3D environments (game levels, maps, etc.).
- **Script Editor**: Scripts can be attached to objects selected in the level editor to customize their behavior.
- **Material Editor**: Shader code is edited and combined with images to form the surface of objects or to create visual effects.
- **Sound Editor**: Volume, attenuation and other settings can be combined with filter effects provided by the sound engine.

Level and script editors can also be part of a game engine and at times, these components may be purposely included with a finished (shipped) game to encourage modification ("modding") of the game itself by the users.

III. Results: The Surveys

A. Survey Methods

Two surveys were conducted. In the first survey (Survey 1) a search of game engines and frameworks that are popular among serious game developers was conducted by searching three databases: Google Scholar, ACM digital library, and the IEEE Xplore digital library. The search terms "serious game", "educational game", and "simulator" were used to reveal approximately 200 academic publications related to serious games and game development. These publications were then scanned for the names of game engines without regard to the context in which the engine was mentioned. Twenty game engines and frameworks that were mentioned in more than one publication were deemed worthy of further investigation. For each of these 20 game engines, the same three databases were used to survey the engine's popularity based on the number of search results returned. In each case the engine name was combined with the search terms described above ("educational game", "serious game", or "simulator"). Many game engines including Unity, Unreal, Torque, and Olive, have names that are common words in the English language whose meaning may not necessarily have any relation to game engines or frameworks. To separate the search results specific to game engines and frameworks from those that did not relate to game development, the first thirty documents from each search term were reviewed. Dividing the number of documents found that referred to the engine by the total number of documents reviewed provides an approximate percentage of the search results relating to the game engine. The results were then normalized to ensure that each of the three search terms were equally weighted.

After completion of Survey 1, Survey 2 was conducted to determine the engines or frameworks serious game developers are actually using. This survey consisted of conducting a search within the Google Scholar, IEEE digital library, and ACM digital library, with the search term "serious game". The search revealed several thousand papers across each of the three databases. However, the papers were manually reviewed sequentially to determine the first thirty papers from each database that met the requirements. Those papers that did not meet the requirements were disqualified and not considered further. For example, with respect to the search of the ACM digital library, over 200 papers were manually reviewed to find the 30 papers that met the requirements. The papers that did meet the requirements were then analyzed (manually reviewed) to determine which game engine was used. The majority of these papers (approximately 80%) were disqualified since they failed to describe the game engine used, described more than one serious game, or described a serious game that was developed from "scratch" without the use of a specific game engine or framework. Papers were also removed from the list if they featured a game that had been discussed in a previous paper, or if the first author matched that of a previous paper. The first thirty papers for each database that satisfied the above requirements were collected to determine the most popular engines utilized by serious game developers.

B. Results

The results of Survey 1 are summarized in Fig. 1 where the *x*-axis represents the search engine/framework and the *y*-axis represents the normalized occurrence (the number of search results) for each of the three search terms ("educational game", "serious game", and "simulator"). The results of Survey 2 are summarized in Table 1 where the top ten most utilized engines and frameworks are listed in order from left to right. 23.3% of the serious games found in the search were developed with Unity, 13.3% with Unreal, and 8.9% were developed using Torque. In addition, Table 1 compares the features (important to developers of serious games) provided by the top frameworks.

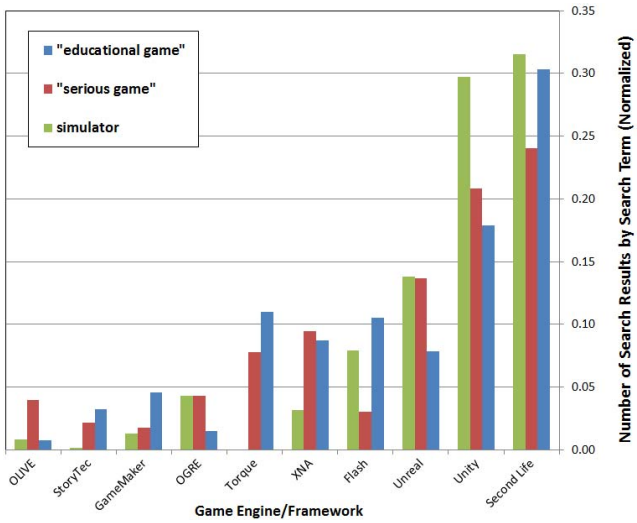


Figure 1. Game engine/framework vs. normalized occurrence (number of search results) by search term. The *x*-axis represents the search engine/framework and the *y*-axis represents the normalized occurrence (the number of search results) for each of the three search terms.

	Unity	Unreal	Torque	HTML5	Flash	Source Engine	Ogre	Second Life	GameMaker	XNA
Level editor	■	■	■		■	■		■	■	
Scripting	■	■	■	■	■			■	■	
C++		■	■			■	■			
Networking	■	■	■	■	■	■		■	■	■
3D Graphics	■	■	■			■	■	■		■
Shader effects	■	■	■			■	■		■	■
Dynamic shadows	■	■	■			■	■			■
Physics	■	■	■			■		■	■	■
Artificial Intelligence	■	■	■			■			■	
Free non-commercial	■	■	■	■		■	■	■		■
Free for commercial				■			■			■
Mobile Devices	■	■		■	■		■		■	■
Web player	■			■	■				■	■

Table 1. Game engines and frameworks and their features.

IV. Discussion and Conclusions

Two surveys were conducted using three academic databases. The first survey (Survey 1) compared the number of papers mentioning the name of each engine in relation to the search terms "serious game", "educational game", and "simulator". The frameworks and engines that scored well in this survey are those that are often discussed in academic writings. Second life was by far the "most talked about" even though Second Life is not specific to serious games. The serious games created with Second Life are essentially modifications ("mods") that exist as a location within the Second Life world. After Second Life, Unity, Unreal, Flash, XNA Game Studio, and the Torque game engine had the greatest number of search results. The second survey attempted to discover which game engines and frameworks were used most often to develop serious games. A search was performed using the search term "serious game". Papers detailing the creation of a serious game were downloaded and read in order to determine which engine was used. Five of the top ten engines are full featured game creations frameworks and eight of the top ten are commercial products. However, six of these commercial game engines are free for non-commercial use. Although there are a few game engines designed specifically for serious game development (e.g., the Delta3D simulation and gaming engine [10]), they do not appear our list of top ten most utilized game engines.

The results presented here suggest that serious game developers are primarily using game engines and frameworks that were designed specifically for the creation of leisure or entertainment games. Given the disparity of available resources to serious game developers in comparison to commercial (entertainment) game developers, it is peculiar that they chose the same tools. This suggests that the currently available serious game engines may be lacking many of the features found in commercial engines. The results may also indicate that game engines designed specifically for serious games do not simplify the process beyond that of commercial entertainment focused engines and frameworks. Each game engine/framework has a number of features that may lend itself to a specific serious game development project. Table I summarizes ten of the most utilized game engines and frameworks along with common features that are important to developers of serious games.

Although a significant amount of time and effort was placed to conduct the searches and review the hundreds of resulting papers, there are limitations to our results. More specifically, our results are specific to papers within three academic databases (Google Scholar, IEEE digital library, and ACM digital library), and as a result, game engines and frameworks that are not described in any research-based publications may have been missed altogether. Our results (Survey 1 and 2) are also subject to any bias Google Scholar, IEEE, or ACM may have relating to the ordering of the search results. Furthermore, although access to the ACM and IEEE digital libraries was available, 46% of the articles revealed in the Google Scholar search were not freely accessible and thus not considered. Despite these limitations, the above surveys do help to remove any personal bias on the part of the authors in selecting, and ranking the frameworks. We are confident that the results presented here will be useful to designers and developers of serious games and may serve as a first step to their development projects.

Acknowledgment

The financial support of the *Natural Sciences and Engineering Research Council of Canada*, the *Social Sciences and Humanities Research Council of Canada* IMMERSe initiative, and the *Canadian Network of Centres of Excellence*, GRAND initiative is appreciated.

References

[1] K. Corti, "Game-based learning; a serious business application," PIXELearning. Coventry, UK, 2006.

[2] V. J. Shute, M. Ventura, M. Bauer, and D.Zapata-Rivera, "Melding the power of serious games and embedded assessment to monitor and foster learning," In U. Ritterfeld, M. Cody, and P. Vorderer (Eds), Serious Games. Mechanisms and Effects.Routedle Publishers, New York, NY. USA, 2009.

[3] K, Squire, and H. Jenkins, "Harnessing the power of games in education," *Insight*, vol. 3, no. 1, pp. 5-33, 2003.

[4] B. Bergeron, "Developing serious games," Thomson Delmar Learning, Hingham, MA. USA, 2006.

[5] K. Becker, and J. Parker, "The Guide to Computer Simulations and Games," John Wiley and Sons, Inc., Indianapolis, IN USA, 2011.

[6] N. Iuppa, and T. Borst, "End-to-End Game Development: Creating Independent Serious Games and Simulations from Start to Finish," Focal Press, Oxford, UK, 2010.

[7] P. Petridis, I. Dunwell, D. Panzoli, S. Arnab, A. Protopsaltis, M. Hendrix, and S. de Freitas, "Game engines selection framework for high-fidelity serious applications," *International Journal of Interactive Worlds*, vol. 2012 (2012), Article ID 418638.

[8] M. Lewis, and J. Jacobson, "Game Engines," *Communications of the ACM*, vol. 45, no. 1, pp. 27, 2002.

[9] A. Sherrod, "Ultimate 3D Game Engine Design & Architecture," Charles River Media, 2007. Boston, MA.

[10] R. Darken, P. McDowell, and E. Johnson, "Projects in VR: the Delta3D open source game engine," *IEEE Computer Graphics and Applications*, vol. 25, no. 3, pp. 10-12, 2005.

Content Creation for a 3D Game with Maya and Unity 3D

Matthias Labschütz[*] Katharina Krösl[†]
In alphabetical order: Mariebeth Aquino[‡] Florian Grashäftl[§] Stephanie Kohl[¶]
Supervised by: Reinhold Preiner

Institute of Computer Graphics and Algorithms
Vienna University of Technology
Vienna / Austria

Abstract

'Dynamite Pete'[1] is a 3D game we developed with Autodesk Maya and Unity 3D in a team of 26 computer science students with varying skills and expertise in content creation. A game development pipeline explaining the production of the game from concept to release is presented. In addition, this paper explores the challenges faced by the project staff and outlines the experiences gained during the project's implementation.

Keywords: content creation, content creation pipeline, game production pipeline, Autodesk Maya, Unity 3D

1 Introduction

Nowadays, the quality of graphics and realism of games is constantly increasing, since consumers are always demanding a more realistic look and feel in their games. This means that improvement of renderings, outstanding content, more believable animations and more authentic behavior of artificial intelligence are needed. Therefore the work of artists and animators is crucial for the success of a game and the prosperity of a game development studio.

In 2010 a group of 26 students attending 'Maya-Course 2' at the Vienna University of Technology performed a game production process in a game development studio. The ultimate goal of this exercise was to produce a video game. Unity 3.1 [10] was chosen as game engine, but the focus was placed on 3D content creation for real time application, using, besides other tools, Autodesk Maya 2011. The game development team brought in different sets of skills which ranged from beginners with hardly any knowledge about content creation to students with particular profession. Every member of the team was given at least one role and had to fulfill tasks corresponding to their job title. The assigned positions covered Project Management, Technical Direction, Art Direction and different Artists. The main areas of these artists were Modeling and Sculpting, UV-Layout, Texturing, Lighting, Rendering, Rigging, Animation, Level-design and Sound. The resulting game 'Dynamite Pete' is a comic-style Western-Adventure where the player plays the role of the antihero named Pete and has to escape from a canyon. Figure 1 shows an example screenshot of the finished game.

Figure 1: In-game screenshot of the game 'Dynamite Pete'.

This paper explains a basic strategy for the professional development of a content-intensive video game in a large team and shares experiences from our project. In the following, we will shortly introduce the tools that were used (Section 2), schematize the workflow of our game development process (Section 3) and finally explain some selected challenges that had to be overcome during the creation of 'Dynamite Pete' (Section 4).

2 Tools

The main tools used throughout the project were Maya for modeling, animating and rendering and the Unity game

[*]Technical Director Workflow: e8971103@student.tuwien.ac.at
[†]Artist: e0325089@student.tuwien.ac.at
[‡]Producer: e0326746@student.tuwien.ac.at
[§]Art Director: e0300310@student.tuwien.ac.at
[¶]Technical Director Unity: e0626088@student.tuwien.ac.at
[1]http://www.cg.tuwien.ac.at/maya/

engine for implementation. In addition, image, audio and video editing tools as well as drawing, sculpting and communication tools were used.

2.1 3D Animation and Modeling Software (Autodesk Maya)

Autodesk Maya (freely available for educational purposes[2]) was chosen as 3D modeling, animation and rendering tool in this project. Maya provides artists with an end-to-end creative workflow [4]. As a professional tool it is very complex and offers a great number of features. Despite a steep learning curve, using Maya in its whole complexity requires a long-winded learning process. Another drawback is the lack of forward compatibility, which means all the artists had to work with the same version of Maya irrespective of their preferences.

2.2 Game Engine (Unity 3D)

In this project, the free version of the game engine Unity 3.1 was chosen for the production of 'Dynamite Pete'. There are different export options in this game engine, each one dedicated to another platform (e.g.: Web Player, PC and Mac Standalone, iOs, Android, Xbox 360, PS3), which eases the development for different consoles or devices. Unity attracts especially small and middle-sized development studios who hardly invest in expensive high-end rendering engines. Furthermore prototyping and game development is very quick due to the WYSIWYG ("what you see is what you get") editor which allows instant changes and live editing. In addition, Unity provides multiple built-in shaders and effects as well as a physics engine and collision detection.

For a students' project the most important reason for choosing Unity is the fact that it is very easy to use and to learn. Developing with Unity is mainly based on drag and drop with occasional adapting of scripts rather than writing code. Apart from shaders and effects which can simply be turned on in some game settings, Unity provides numerous scripts which can be dragged onto 3D models. These scripts act for example as character controllers, follow up cameras or other important features. However, Unity lacks of integrated modeling abilities, which is the reason for using Maya as external modeling tool.

Another major drawback of the free version of Unity is its lack of SVN support. This makes it difficult for multiple programmers to work concurrently on one single project, which will be discussed in detail in section 4. Like Autodesk Maya, Unity is not forward compatible, but overall the advantages outweigh the disadvantages. However, the use of Unity Pro is advisable, due the previously mentioned restrictions in the free version.

[2] http://usa.autodesk.com/maya/trial/

3 General Game Production Pipeline

A game production pipeline is basically a concept of workflow management for use in the game development process. The phases of this pipeline are certain tasks that need to be fulfilled until the release of a video game.

Figure 2 gives a detailed overview of the development process of this project, as it will be explained in the following chapters, showing the main five tasks in color.

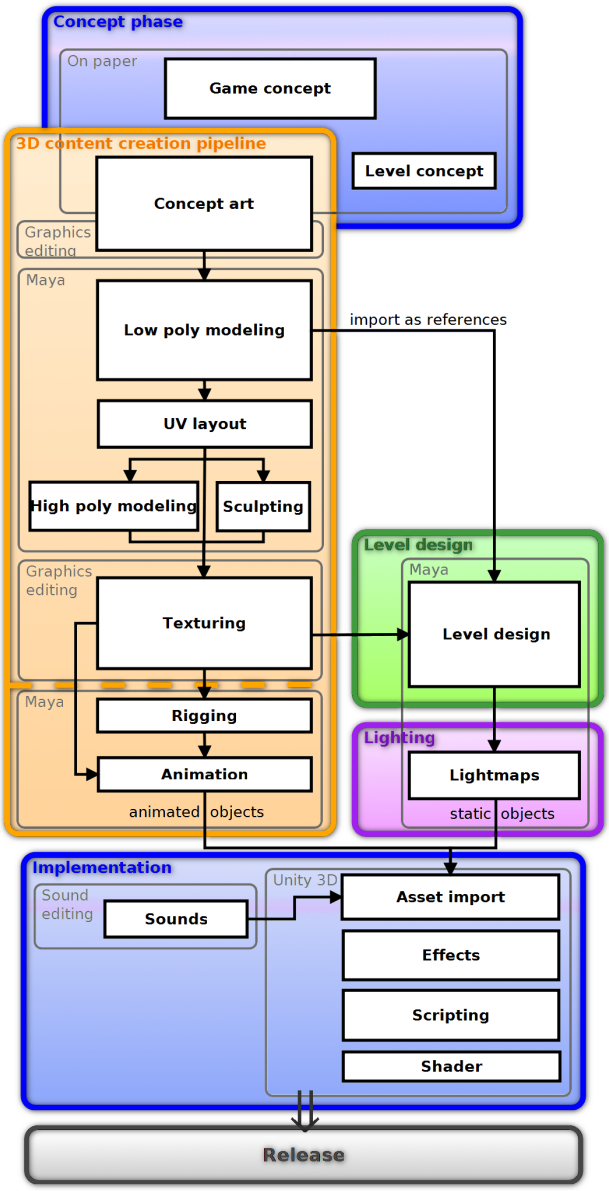


Figure 2: Game production pipeline overview: Concept, content creation pipeline, level design, lighting and implementation as organized in this particular project.

Some of these tasks require to be carried out sequentially as presented in the following. Nevertheless, to increase productivity, it should be a goal to break up this sequential approach, to allow people to work in parallel on different tasks.

Since this very project was an attempt to implement the workflow in a game development studio at a larger scale with very limited production time, it was necessary to follow a pipeline approach without iterations. The scope of this project was 3D modeling, with focus on content creation. Other aspects, like game-design, were of minor interest. The following paragraphs describe the production stages of the project in detail.

3.1 Concept Phase

During the concept phase a small team drew an outline of the plot, the setting and the game mechanics. The game's environment was chosen to be a wild-west desert. A comic style was preferred over a photorealistic style and the goal of the game was set to collecting dynamite for breaking out of a hostile canyon.

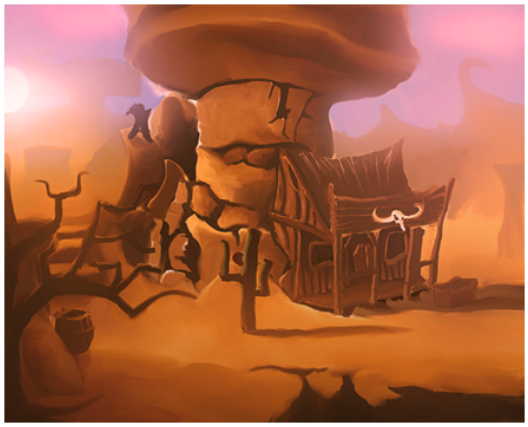


Figure 3: Early painted concept art

After this stage, the look and feel of the game content had to be worked out, requiring multiple revisions of concept arts. An example is shown in Figure 3. Inspiration was coming from all different sorts of media, starting from similar computer games and movies to comics and music. Color and proportions play an important part in the game's visual style. A dirty textured comic look was chosen, inspired by 'Star Wars - the Clone Wars Series' [9] and Woody, the Cowboy in 'Disney Pixar's Toy Story' [7].

3.2 3D Content Creation Pipeline

The orange box in Figure 2 illustrates the 3D concept creation pipeline which contains the stages every 3D model has to pass from concept art to the final model. The following paragraphs describe these stages in detail.

3.2.1 Concept Art

Figure 4: Character concept art of a sheriff and a barmaid model.

Concept artists worked primarily on the main assets of the game (characters and environment), gradually enhancing their work guided by feedback of the art direction and other artists. After the initial sketches on paper, some artists moved on to graphics editing tools, using tablets as input devices. As final step of the concept phase, they created colored front and side views of the assets, which modeling and texturing artists used as guides for their geometry and coloring. Figure 4 shows a small selection of colored character concepts. Most of the concepts were hand drawn and can be found on the development blog.[3]

3.2.2 Low Polygon Modeling

The first step from concept to 3D model is to create a low polygon model. Since Unity can handle triangles and quads, there was no restriction to use a certain modeling technique only. Nevertheless, most modeling artists preferred quads over triangles. Low polygon modeling also includes smoothing or hardening normals.

To avoid models with very differing level of detail, it is highly recommended to define a maximum polygon count in advance, depending on a model's size and importance. However, multiple revisions were necessary during this phase to achieve a consistent style for all the low polygon models.

For houses, a modular system was used to create numerous different house types out of individual basic parts. This tile based approach allowed more variations with less time for modeling.

3.2.3 UV Layout

Before texturing or high polygon modeling could start, UV layouts had to be done either by UV layout artists or by the modeling artists themselves. A UV set in Maya consists of

[3] http://twoday.tuwien.ac.at/mayakurs22010/

a single UV layout. An object with multiple texture types can have multiple UV layouts. For this project a maximum of three UV sets per object was defined:

- Color map UV set: For color textures. Faces can reuse texture space, meaning that a certain space in texture space can be used by multiple faces.
- Normal map UV set (optional): Only used if a separate normal map was required.
- Light map UV set: Can not include overlapping faces in texture space. Since each face can have different lighting information, each face has to have its own amount of space in the UV layout.

3.2.4 High Polygon Modeling

After the UV layouts were done, some of the low polygon models were improved to high-resolution models either by a smooth-operation provided by Maya (adding additional subdivision polygons), or by artistic sculpting using Autodesk Mudbox [5]. The result of this high polygon modeling step was a normal map.

3.2.5 Texturing

Texturing started right after the UV layout was done. A color table (Figure 5) was used as reference for texture artists. A screenshot of the UV layout in texture space was exported to digital image processing tools. The texture coordinates (UVs) served as orientation for texture artists, who painted their textures on these coordinates, using only colors from the given color table.

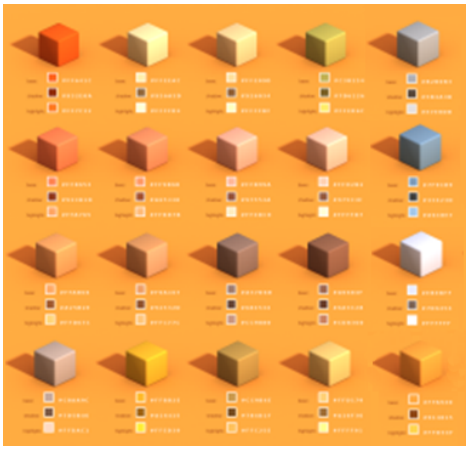


Figure 5: Color table for texture artists

For frequently used textures (such as wood), texture templates were created. For each contained material, these libraries provided bump maps, normal maps and differently colored layers which artist could switch on or off, to create new textures fast and easy.

3.2.6 Rigging and Animation

To enable artists to easily create naturally looking animations of characters, a technique called 'rigging' is used, which creates a virtual bone structure for each character. This structure can then be used to control the movements of the characters for animation in a natural way. After the bone structure is finished, the character's mesh is bound to the skeleton and weighted between influencing bones. Usually, inverse kinematics are used to specify the bones position when moving handles of the skeleton.

In this project, each game character went through an individual rigging process in Maya, offering different controls for the animators. Since this was an educational project, most of the participants decided not to use premanufactured rigs (such as the built-in Full Body IK [3]). The animation artists then produced key-frame based animations for some movements (e.g.: walkcycle, idle, attack) for each game character, which were done in Maya.

3.3 Level Design

3.3.1 The Level Design Process

The previous section described how single pieces of static and dynamic content have been created. The following sections will explain how all these parts are combined to form a virtual environment and finally a playable game. The first step on this path is level design which can be seen as an assembly process that incorporates or creates content to shape the environment of a game. Depending on the genre of a game, the scope of level design may lie on the content creation, the technical assembly, or the game-play tuning aspect.

Scheduling: In general, the crucial amount of work in level design is carried out at a late stage in the creation process. Not only because level design requires parts of the engine to be finished but also assets to work with. In order to allow the level design process start early, it is good practice to prioritize the assets depending on their importance for the level designer.

Concurrent working: Concurrent working on game levels is difficult to be done efficiently and can easily result in a high organization and merging overhead [11]. For small game environments, one person working on the level design is advisable to reduce the overhead. For larger projects, concurrent work of several designers on levels of high complexity can be achieved by splitting the levels along certain regions.

Asset placement: Open terrain game levels often require a lot of environmental objects to be placed into them to produce a rich and exciting game feeling. Automatic placement can help speeding up the process, while keeping a human designer in charge of where objects are placed. For

large projects, more advanced level-editing tools can be used, which allow modifying the layout in a more abstract and faster way (e.g. drawing a map of the area, while the content is placed automatically).

Figure 6 shows the level design approach in relation to other tasks during the creation of a game. The solid lines describe the path of content as it is passed through the team during the game development process. The dashed lines denote requests and tools which will not be included directly in the final release. As shown in Figure 6, level design is a step between content creation and programming. Therefore, changes in level design require interactivity or fast compiling mechanisms to see the impacts on the game.

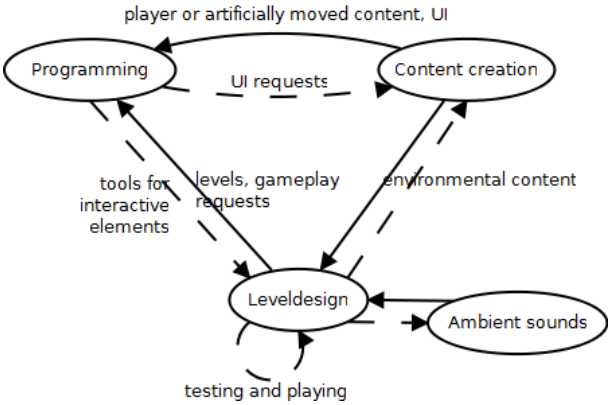


Figure 6: Level design in context to other game development tasks.

3.3.2 Level design in this project

The level design started right from the beginning of the project with concept plans of the game area [8] followed by modeling and texturing of the terrain. After this initial phase a level file was created and assembled in Maya, using references, so that unfinished content (e.g. low polygon models) could already be placed in the scene and be updated on the fly. In the end the whole scene was imported into Unity.

Figure 7 shows the stages of development of the level from the concept art over the terrain model to the final level rendering with buildings and other assets. The level design process took the following considerations into account:

- During the creation of the terrain, conceptual areas were planned for later object placement. (e.g. graveyard area, scenic overview, spots for buildings along the road).
- The level design was based on static geometry. To keep the player interested in exploring the level, many different floors and long paths along the terrain were created.

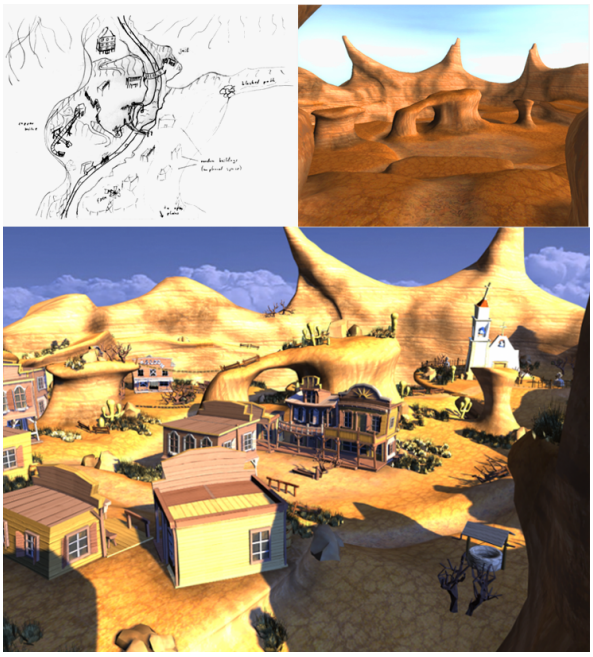


Figure 7: Three stages of level design: concept, terrain model, final scene.

- To underline the escape scenario of the game, a certain degree of hostility was added to the terrain design (e.g. in the shape of spiky rock formations).
- To keep the level simple for lighting, no caves were created.

3.4 Lighting

3D Production packages include powerful renderers to simulate light propagation through a scene. To this point, physically correct simulations of light are impossible because of measuring uncertainty and the inability to solve the rendering equation exactly. In addition, approaches close to physical models are often very costly and cannot be done in real-time yet.

Since computer games are real-time applications, preprocessing for global illumination is often used ('Light Baking') to achieve realistic lighting at runtime. During the process of Light Baking, the whole scene is pre-lit and the final color distribution is stored in textures. These textures can be either applied directly to a polygonal surface or saved as vertex colors. The advantage of Light Baking is that the artists can use high quality rendering algorithms to pre-calculate the lighting of a scene. Disadvantages are the added complexity in the production workflow and the lack of dynamic lights. Combining pre-calculated and real-time lighting effects can be a challenge, because numerous customized shaders and materials need to achieve acceptable color- blending effects.

3.4.1 Lighting in this project

The scene was meant to be illuminated by the sunset with dominating pink and purple colors. Dark areas were avoided to keep the lighting simple. Color textures had to be desaturated, so the scene could be lit with enough power to produce a believable result. Autodesk Mental Ray [6] was used to render the scene. Its fast final gathering algorithm in combination with a light emitting sky-dome was chosen to achieve the effect of global illumination for the game's content.

The sun consisted of two simple directional light sources not affecting the sky-dome. One light-source was used for controlling the light color and intensity. The other light source was used for controlling the shapes, intensities and colors of the shadows.

It was important to know the scene's look under final lighting conditions at an early stage of the production process to define appropriate texture color guidelines. Therefore, this light setup was developed at an early stage to test different texture colors and shaders in the chosen environment setting.

The final step in lighting was light map texture baking. Diffuse color bouncing required emitting and reflecting colored objects in the scene during bake-time. These objects had to be prepared for baking. First, they were given a second UV-set, which was not overlapping or tiled.

For light-mapping and export, objects were grouped into export groups, sharing one common light-map, and were then imported separately into Unity. A building and parts of its corresponding light-map are shown in Figure 8.

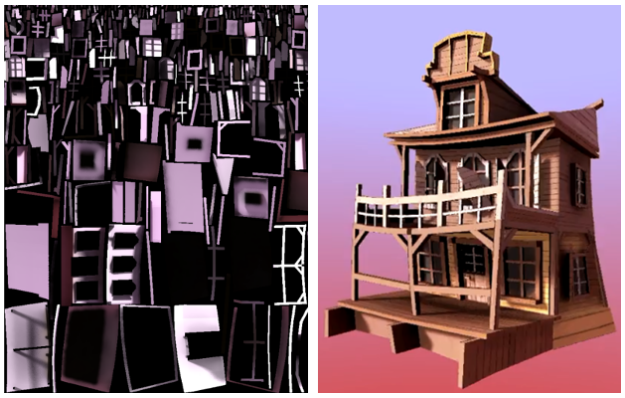


Figure 8: Building to the right and its corresponding light-map to the left.

3.4.2 Concept and Presentation Rendering

In addition to the scene's lighting, rendering was also used for many different tasks during the development process. Concept renders were created in order to communicate the aesthetics of the game environment. A game trailer was produced showing some animation sequences and renders were created by artists to share their concepts and models through the team's development blog. Since rendering is a time consuming process, some additional tricks were used to improve quality in these renderings. Depth of field effects were achieved through depth-map controlled blur filters using compositing. Additionally, image post processing was used to achieve the desired quality.

3.5 Implementation

During the implementation phase, intensive work with the game engine started, as all the models had to be imported into Unity. Unity generally uses *.fbx files, but allows the import of *.ma files as well. These Maya files were directly imported into Unity to avoid problems with animated objects exported by Maya 2011 as *.fbx and to skip an additional step in the workflow. The files were then converted into *.fbx by Unity. This procedure makes it easy to re-import changed Maya files but it takes more time for Unity to convert all files.

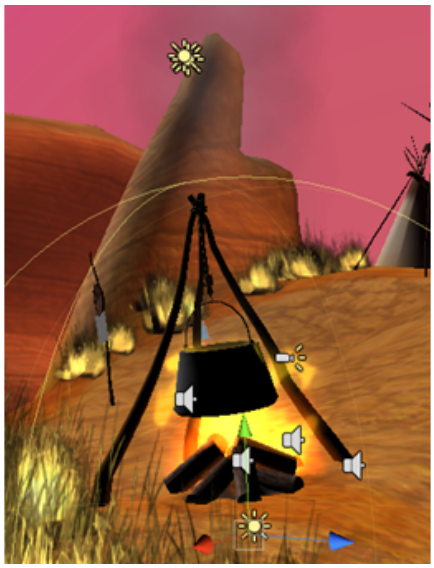


Figure 9: Fire, smoke effects and sound sources.

3.5.1 Object Integration in Unity

In the first step of the object integration process the level, which had been created in Maya 2011, was imported into Unity. It was structured into different display layers, each one dedicated to a different type of models (e.g. houses, grasses, cacti). When the level was completed, the different layers were saved in separate Maya files and put together in Unity. This made it easier and faster to re-import changed layers into Unity. The different characters as well as various pickup items (e.g. dynamite) used in the game story, were directly placed into the level in Unity.

3.5.2 Effects, Shader Programming

As soon as the main part of the game was finished, effects like fire and smoke (Figure 9), fog, water, dust, lens flare and explosions were implemented. All these effects, except lens flare and fog, were created with an Ellipsoid Particle Emitter or a Mesh Particle Emitter provided by Unity. Fog was simply activated in Unity's render settings.

3.5.3 Sound

In the final phase background music and sound effects were added to the game. To add a sound file to an object in Unity, an 'Audio Source Component' had to be added to the specified object. In Unity's audio source options, the audio clip had to be defined and some other options like the volume were individually adapted.

3.6 Release

The final result of the whole production process was a game prototype with one level. Overall, it took a production team of 26 students about 1550 person hours to develop the game 'Dynamite Pete'. Table 1 gives a detailed overview of the working time spent on the individual phases of the project.

Task	Time in person hours
Concept and references	130
Low polygon modeling	500
UV layout	150
High polygon modeling	50
Texturing	190
Rigging	20
Animation	50
Level design	50
Lighting	60
Implementation	200
Miscellaneous (presentation)	50
Organisation and feedback	100
Total	1550

Table 1: Development time overview

After the end of the project, the final game was presented in a release event. The artists created a trailer video, character sheets, posters and character turntables for this event.

4 Further Challenges

4.1 Terrain Multi-Texturing

The usual approach for texturing 3D models is to apply a single texture per object. Since the terrain is a very large object, tile-able textures were used to define its surface properties. These textures are repeated seamlessly across a geometric surface representation. However, since terrain usually consists of different material, more than one tile-able texture had to be used.

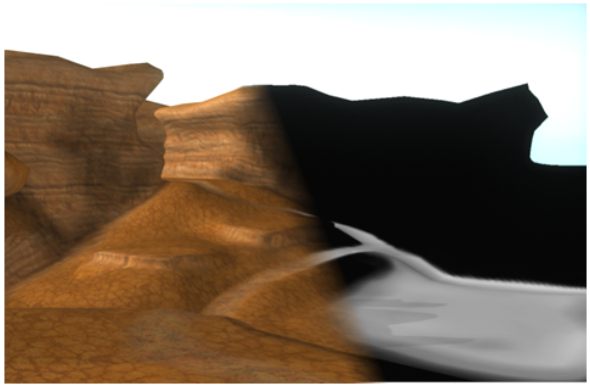


Figure 10: Left side: textured terrain. Right side: alpha map for the path layer.

Multiple layers of tile-able textures were used in combination with an alpha texture to define the global occurrence of a material. Figure 10 shows the textured terrain on the left side and the alpha map for one layer on the right side. In Maya, the material that supports this kind of texturing method is called 'Layered Texture'. In Unity, this method of texturing is only provided for terrains created from height-maps. Therefore, a custom fragment shader had to be written for the terrain in this project.

The following shows the essential part of this fragment shader code. First, the albedos of the individual overlay layers are consecutively accumulated. Finally, the calculated albedo is multiplied with the light intensity to obtain the output color.

```
//'c', 'c1', 'c2', 'c3' ... color textures
//'a' ... alpha values in rgb channels
//'lightmap' ... light map texture

//overlay 1st layer
o.Albedo = (1.0f-a.r)*c.rgb + a.r*c1.rgb;
//overlay 2nd
o.Albedo = (1.0f-a.g)*o.Albedo + a.g*c2.rgb;
//overlay 3rd
o.Albedo = (1.0f-a.b)*o.Albedo + a.b*c3.rgb;

//1.5f to control light map brightness manually
o.Albedo = o.Albedo * lightmap * 1.5f;
```

4.2 Concurrent working with Unity

As mentioned earlier, there are some differences between Unity Free and Unity Pro. Because the game was developed with Unity Free, a solution for the SVN problem had to be found. Unity Pro saves all asset metadata and import settings for each asset in a corresponding metafile. These files need to be versioned along with the associated asset

[2]. In Unity Free all these metadata are saved in the library directory [1]. To share a project without Unity Pro, the library directory has to be compressed and uploaded via SVN. After downloading the project from the SVN server the library directory has to be uncompressed into the Unity project folder. One negative effect of this solution is that it is not possible to work concurrently on the Unity project file. Because the whole library folder is uploaded, it is not possible to merge the different metadata, only the whole folder can be overwritten. Therefore, for multiuser projects it is recommended to use Unity Pro.

5 Summary

Together with our colleagues, we deployed a professional game development studio workflow in the context of a 3D content creation project. The utilization of 3D content in a 3D game put theoretical knowledge into practical use. Participating students improved not only their technical skills, but also learned valuable lessons in team work. The major challenges were different theoretical and practical understanding of the subject, unalike working methods and variable personal availability. A blog was used by most of the students to post their work and to get feedback in a constructive way. The final stages in the process of game development were rushed in about one or two weeks. At the time of the release deadline, the result was a decreased quality in game-play and lighting as well as the game mechanics, which had not been tested before. More concurrent work and earlier deadlines would have been needed to speed up the production process. Nevertheless the project was finished within the given time schedule, resulting in a nice game containing rich and detailed content.

6 Acknowledgements

The authors would like to thank all team members who participated in this valuable exercise and dedicated their time to this project. Thanks to Rainer Angerer, Martin Brunnhuber, Damir Dizdarevic, Brigit Faber, Markus Fellner, Roman Gurbat, Andreas Himmetzberger, Rosemarie Hochreiter, Roman Hochstöger, Bernhard Holzer, Peter Houska, Albert Kavelar, Desiree Lavaulx-Vrecourt, Andreas Lenzhofer, Thomas Mayr, Johannes Sorger, Stefan Stangl, Nicolas Swoboda, Markus Tragust and Ulrike Zauner. Our special thanks to our lecturer Markus Weilguny, who guided us through the whole production process. Last but not least we would also like to thank Associate Professor Michael Wimmer for making this lecture possible and for giving us the great opportunity to present our efforts and findings in this paper.

References

[1] Unity 3D. Unity: Behind the scenes. http://unity3d.com/support/documentation/Manual/Behind%20the%20Scenes.html, 2010.

[2] Unity 3D. Unity: Using external version control systems with unity. http://unity3d.com/support/documentation/Manual/ExternalVersionControlSystemSupport.html, 2010.

[3] Autodesk. Autodesk maya online help: Fbik (full-body ik). http://download.autodesk.com/us/maya/2010help/index.html?url=Glossary_F_FBIK_fullbody_IK.htm, topicNumber=d0e188540, 2000-2009.

[4] Autodesk. Autodesk maya 2011: Features. http://usa.autodesk.com/maya/features, 2011.

[5] Autodesk. Autodesk mudbox 2011: Features. http://area.autodesk.com/mudbox2011/features, 2011.

[6] Autodesk. Mental ray. http://usa.autodesk.com/adsk/servlet/pc/index?siteID=123112&id=13566140, 2011.

[7] Disney/Pixar. Toy story. http://www.pixar.com/featurefilms/ts/, 1995-2011.

[8] Michael Stuart Licht. Gamasutra: An architect's perspective on level design pre-production. http://www.gamasutra.com/view/feature/2848/an_architects_perspective_on_, 2003.

[9] Lucasfilm Ltd. Star wars. the clone wars. http://www.starwars.com/theclonewars/, 2011.

[10] Unity Technologies. Unity 3d. http://unity3d.com/, 2011.

[11] Mick West. Gamasutra: Collaborate game editing. http://www.gamasutra.com/view/feature/3991/collaborative_game_editing, 2009.

2016 Eighth International Conference on Measuring Technology and Mechatronics Automation

Design and implementation of 3D Virtual Digital Campus -- Based on Unity3D

XiaoJing

（College of Electronic Information Engineering, Wuhan City Vocational College, Wuhan 430064, China）

a519755597@qq.com

***Abstract*- Based on the Unity3D multi platform to establish a three-dimensional virtual digital campus system, has a strong practical. Using the Unity3D engine to realize the virtual digital campus scene design, access and interactive roaming and other functions, the need to achieve a combination of various simulation effects. The 3D Virtual Digital Campus Based on Unity3D development system has high efficiency and low maintenance cost, which has obvious advantages. This paper will mainly elaborate the principle of system design, design of virtual scene and the practice of the system.**

Keywords- 3D virtual, Digital campus, Unity3D multi platform

Further application of digital technology in the campus to build digital campus provides the possibility of implementation, with the further development of the Internet information technology, the digital campus will learn from traditional network office for the development of realistic virtual roaming, learning, and office. The use of Unity3D multi-platform technology to achieve efficient development of virtual digital campus system provides implementation way, is a kind of stable operation, friendly interface of high development tools. It is based on an open source component of the game engine (Mono.net), which can realize the 3D virtual scene and good interaction of digital campus system, and realize the online office and to learn.

Ⅰ.THE DESIGN PRINCIPLE OF 3D VIRTUAL DIGITAL CAMPUS

In the process of designing 3D virtual digital campus, the following principles should be followed, which is the step of the whole design plan.

First, to obtain the overall geographic information data on the campus, which mainly includes the CAD Auto and other buildings.

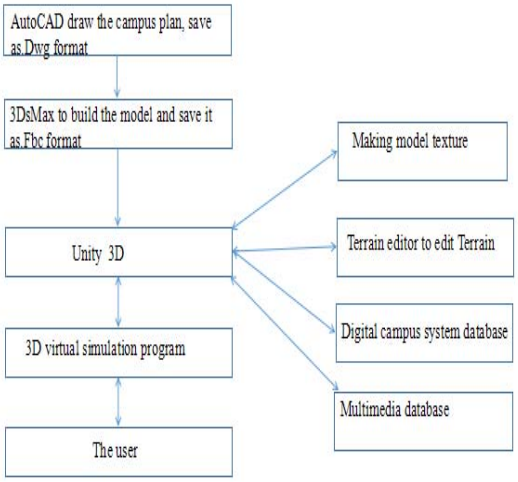
Second. In the 3D modeling software, the 3D digital model is produced, which is based on PS and 3DMax. The rough model is processed and rendered, and the 3D virtual model with high degree of simulation is obtained.

Third, import the production of good three-dimensional model, and according to the geographic coordinates of the model into the Unity3D platform, the development of virtual technology to prepare.

Fourth, the Unity3D software platform using Terrain tool to create specific terrain, and the design of the user interface, the existing digital campus system database to achieve shared data connection. Figure 1 is a framework for the design of 3D virtual digital campus system.

2157-1481/15 $31.00 © 2015 IEEE
DOI 10.1109/ICMTMA.2016.55

Figure 1.3D virtual digital campus system architecture diagram



Ⅱ.VIRTUAL SCENE DESIGN

A.Create and import the model

Derived from 3DsMax will create good model, its storage format for fang binxing. Assets at the same time, the Unity3D project file directory of the new name of the file named Object, and in the folder instead continue to two new folder to store the virtual model respectively the material of the ball (Materials) and map file (Textures). Then fang binxing model file into the Object folder, the software system can automatically identify the folder under the different types of storage folder, create the material of the ball on the save Unity3D and map files, can automatically identify.

B.Texture and texture

Texture map can guarantee the quality of the model, in Unity3D software environment, design also need to ensure the quality of the texture and texture, virtual to achieve good results. At the time of making modeling, should also be pre-determined all sorts of style of model material, texture and physical properties, to smooth the model imported into Unity3D software in the texture map and script Settings (Shader). Unity3D software platform has the formidable material written and audio-visual language tools, equipped with tools and CgFX and Direct3D syntax is similar, not only can record the basic point of endpoints and impression (vertex/pixel) properties, also can describe the material has all the attributes. With glass material programming code below as an example to illustrate:

Sub Shader{Pass{Tags{ " Queue" = " Transparent" " Ignore Projector"

= " True" " Light Mode" = "Always" }

ZW rite On Cull back

Color mask 0}

Tags { " Queue" = " Transparent" " Ignore Projector"

= " True" " Render Type" = " Transparent" }

ZWrite Off Blend One One Fog { Color (0，0，0，0) }}}

C.Using the Terrain editor to edit Terrain

In the development of 3Dvirtual system in the process of digital campus, Terrain editor is an extension of the Unity3D platform tools, based on the editor that designers can efficiently produce 3Dvirtual Terrain map, and the editor USES a unified standard, to the mountain, can achieve a higher degree of urban and rural scene simulation. Secondly, can also be friendly GUI interface design, the whole terrain texture on the shop decoration virtual environment, make 3D virtual interface can be more realistic. By mixing or merging terrain texture can achieve a smooth transition, from one to another in a map, can produce diversified environment. The editor that allows including trees, rocks, trees and bushes, grass, houses, walls of any object, such as a quick layout, realize flexible combination. Will also be able to apply the image as a blueprint in each object, also can change its color attribute. In place of every detail, can choose the rotation of the scale, location, location, etc., and able to implement all the unity of the same object

color, such as trees, etc.

Ⅲ .VIRTUAL DIGITAL CAMPUS SYSTEM IMPLEMENTATION

A.Roaming technology

Similar to the game scene, 3D virtual digital campus system, can be operated through the keyboard and mouse control role in moving in the scene (roaming), can choose any direction after left before and after the walk. You can also select the walking style such as: running, jumping, walking, etc. Roaming in the implementation. Mainly through the following aspects.

First, in the 3DsMax software to make character images of various motion states such as: walking, standing, jumping and running, etc. Again after completed the animation model export and saved as a set (fang binxing) format of the document. Finally the model imported into Unity3D platform project folder (Assets).

Second, based on Unity3D platform Project column options , you can import all kinds of animation model, and in the properties pane to set the start of the different actions of different model frame and end frame. In dragging animation model on the stage, choose the menu bar command Component/Character/Character Motor, can increase a Character for animation controller model. And then to adjust the task in the attributes panel controller, the outline of the height of the task, and so on.

Third, for animation model add Controller and rotating Controller can be realized after any movement of the virtual objects. For example: "Mouse Look" wheel can very good implementation object rotate up and down or so. By designing the function of the rollout of "Axes" for the "Mouse X", and then set the attribute value is 0, Y can achieve only within about rotation. And add the Controller in the camera object "Smooth Follow", through the "Target" set, pointing to the animation model can finish the camera and animation model of binding targets.

Fourth, in athletic result established in implementation, can deal with all kinds of sports Animation JS script, named by "move Animation Control" script specify various Animation model. The control code expressed as: static initial state of the animation is: **function Start ()**

Is the state of play set to cycle: **{animation "jump" layer = 1; / /**

Beating screen showing setup mode to Clamp, in the flash to the last frame, if the program does not change, will have been playing the animation of the last frame. Generally program is to modify the animation of wrap Mode. Not to introduce one of animation program setup code. The following figure 2 for the role roaming in the 3D rendering of virtual digital campus.

Figure 2.3D virtual digital campus roaming rendering

B.Collision detection and switch scene

In the process of improve the interactivity of virtual environment, collision detection is the need to first solve the problem. By setting the model can effectively solve the physical properties of a wall, in Unity3D platform, a collision detection component Mesh starts the component calculation model can effectively act collision Mesh automatic generation, if the grid is more and will affect the execution efficiency of the system. Therefore, general meeting to grid model to add a more basic model as the father of this model, it is set to become an apply colours to a drawing object. This is very effective to solve the collision problem.

The scenes when the character's perspective into the switch that is a particular area, after switching the

behavior of the scene, here also need to first select the collision, and then according to the result of the collision detection switch to the next scene. After entering the room, for example, you can use event On the Trigger Enter judgment through a Trigger, but with two functions to Load a new scene at this time (**Application. The Load Level (lv)**) and keep the objects in the scene in the process of the switch all the properties of the reserve (**Don 't Destroy On the Load (object)**).

C.Database access

In Unity3D platform, the software System can smooth access various database, which use a dynamic file (System. Data. DLL and System. Enterprise Services. DLLZH) create a database connection and very efficient. In the process of realization of 3D virtual digital campus, to ensure the safety and reliability of the existing data, the system also needs to implement the existing database query function, and text data query (Sql) My connections. Among them, to establish or disconnect from the database code can be expressed as:

Connection code:

Private static void open Sql Connection(string connection String){ / /。Db Connection = new My Sql Connection (connection String);Db Connection．Open( ) ;}

Disconnect the code：**dbConnection.Close( ) ; / /**

D.Platform to build more

Unity3D4. Version 2.2 supports a variety of operating systems and server platforms, such as: the PC, the Web, IOS, Flash, Android, Black Berry, Apps, etc., will also be able to better support the WAMP (Windows + PHP + Apache + My SQL), LAMP (Linux + PHP + Apache + My SQL) and the IIS (Internet Information Services), and other common Web application platform. The latter can also be very good with the host operating system to write, in the aspect of management and control is very convenient. Based on Unity3D multi-platform 3D virtual campus digital system implementation process combines a variety of software configuration and database system. To be one to one correspondence file extensions of each system, in order to more efficient implementation system development. Vivid display effect of 3D virtual digital campus system, under the stable operation, intuitive user experience can be implemented and roaming effect, also can provide rich multimedia audio, its sense of reality will be very strong and intuitive.

REFERENCE

[1] Wang-Xingjie, Li-Chunhua. Research and development of 3D virtual city based on Unity3D platform [J]. 2013 (04).

[2] Li-Qiang, Ouyang-Pan, Lu-Xiuhui. Research and implementation of virtual campus development based on Unity3D [J]. modern electronic technology. 2013 (04).

[3] Li-Junfeng. The research and practice of virtual reality technology and Virtual Campus -- Taking Weifang University as an example of the construction of virtual campus of as a case study on the Journal of engineering graphics, 2011 (03).

[4]He-Xianpeng. The thought of building a new generation of digital campus [J]. intelligence. 2010(08).

[5]Wu-Wenwu, Zheng-De. The construction of University Information Technology campus [J]. computer knowledge and technology. 2010(34).

Development of a BCI Simulated Application System Based on Unity3D

Banghua Yang, Tao Zhang, Kaiwen Duan
college of mechatronic engineering and automation
Shanghai University
Shanghai, China
e-mail: yangbanghua@shu.edu.cn

***Abstract*—This paper develops a BCI simulated application system based on Unity3D, called going to the cinema. Firstly, a 3D car model and some scene models built in 3ds Max are imported into Unity3D .Then, in Unity3D, the virtual reality scene layout is designed with its own and imported models .The audio effects and collision detection are added to enhance realistic immersion. After that, the behaviors of the 3D car are defined. The communication and timer functions are configured. Finally, the system can receive the commands through the TCP/IP protocol to control the 3D car movement to go to the destination- cinema. The simulation results show that the system is feasible and has good performance. The designed system can be used as not only a simulated application system, but also a feedback training system providing effective feedback of vision and hearing, which can lay a foundation for BCI application and feedback.**

Keywords- brain computer interface; application system; Unity3D; virtual reality; TCP/IP

I. INTRODUCTION

A BCI (Brain Computer Interface) is a kind of interface to establish a direct communication channel between brain and external devices. Due to the high temporal resolution and portability of EEG (Electroencephalogram), the BCI system based on it is easy to put into practice [1]. The BCI system transforms the EEG signals to corresponding commands to control external devices [2]. Nowadays, BCI systems are widely used in the fields of rehabilitation, auxiliary control, entertainments and so on. However, on one hand, before the BCI systems are applied in practice, the hardware equipment needs to be configured to test their performance, which will increase the cost. On the other hand, most subjects can control their EEG independently only after numerous trainings [3]. But conventional feedback training systems are so boring that subjects lose motivation, which make the training effects are unsatisfactory [4][5]. Therefore, it is necessary to develop a BCI simulated application or feedback training system. It can be used to test the performance of the practical system without hardware devices. At the same time, it can provide effective visual and auditory feedback to improve the training effects, which will lay a solid foundation for the BCI application [6].

The VR (Virtual Reality) technology has the capability of creating an immersive and interactive environment [4]. Meanwhile, it can configure the ideal environment of the BCI practical system at a lower cost. So, the VR technology can be used to achieve the high quality simulated system. Unity3D, as an integrated game engine and editor, is one of the VR technologies. It is known that components are used to develop games in Unity3D, which include mesh component, physics component, audio component, rendering component and script component, etc. What's more, Unity3D provides rich resource packages, such as terrain creation tools, common scripts, collision detection, sky boxes and so on. High-level programing languages like c# and JavaScript are used for implementing the script functions [7]. In short, all of these are beneficial to create high-performance multimedia applications or games efficiently.

In this paper, a BCI simulated application system, called going to the cinema, is designed in Unity3D. Receiving commands from BCI signal acquisition and processing system through the TCP/IP protocol, the designed system can control the 3D car to go the cinema in real time. The visual and auditory VR scene in the system is more immersive and fun, which can inspire subjects' participating enthusiasm. In the meantime, the real-time feedback of vision and hearing makes subjects adjust their motor imagery, which improves the adaptability of human brains to computers. The timer function is designed to test subjects' control effects and the classification accuracy of the EEG signals. Above all, the environment configured is consistent with the practical one, which can effectively test the performance of BCI practical application system.

II. BCI SYSTEM OVERVIEW

The whole BCI system consists of two parts, which is shown in Fig. 1. One is the signal acquisition and processing system and the other is the Unity3D simulated application system. In the signal acquisition and processing system, the EEG signal, generated by subjects imaging left hand or right hand movement, is acquired by the electrode cap. Then the signal is processed and transformed to a real number in [-1, 1]. In order to control easily, the real number is further transformed to the command with the predefined format, which is used to control the car to perform turning left, turning right or moving forward. After that, the command is sent to the simulated application system to control the movement of the 3D car through the TCP/IP protocol.

The real number is transformed to the command to control the movement of the 3D car. So, the relationships

978-1-4673-7673-0/15 $31.00 © 2015 IEEE
DOI 10.1109/ICVRV.2015.30

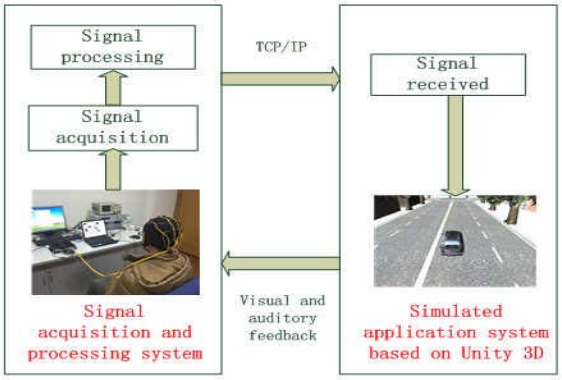


Figure 1. Whole BCI system

between the real number x and the movement of the car are described as

$$\begin{cases} v = 2*(0.5-|x|), x \in [-0.5, 0.5] \\ r = x*0.4, x \in [-1,-0.5) \cup (0.5,1] \end{cases}. \quad (1)$$

where the real number x in [-0.5.0.5] represents 3D car moving forward, and the relationships between the real number x and move speed v (m/s) can be described as a mathematics equation, v=2*(0.5-|x|). Similarly, the real number x in [-1,-0.5) and in (0.5, 1] represent 3D car turning left and turning right, respectively. The relationships between the real number x and rotation speed r (rad/s) is described as a mathematics equation, r=0.4*x. Among the predefined format, the "MD" is the rotating identifier while the "MS" is the moving identifier. For example, if the system receives the command of "MDR0.3", the car will turn right at a rotation speed of 0.3rad/s. If the system receives the command of "MDL0.3", the car will turn left at a rotation speed of 0.3rad/s. If the system receives the command of "MS+1", the car will move forward at a move speed of 1m/s.

III. SIMULATED APPLICATION SYSTEM DESIGN

The simulated application system is built in Unity3D with the models built, the visual and auditory VR scene constructed and the car behavior defined. Then the TCP/IP communication function is configured, through which the application system can receive real-time commands to control the 3D car. The timer function is also added to reflect subjects' control effects. The following steps to design the system are described in detail.

A. Building Models

In order to develop the system rapidly, the models are offered by 3ds Max and Unity3D own resource packages. The models include 2D road models and 3D models, such as a 3D car, buildings, trees, a cube and a sphere. Among them, the cube and sphere are used to replace the 3D car and the cinema in the mini-map respectively. Other models make up the VR scene.

As a tool for 3D modeling, animation making and rendering, 3ds Max is widely used in game development, character animation or other fields. More important, it cooperates with Unity3D smoothly. Therefore, the 3D car and the building models are built in 3ds Max with the method of polygon modeling, resulting in creating 3D mesh models [8]. The mesh models are mapped textures to increase their sense of reality and then are exported as the FBX file format. Finally, the model files are imported into Unity3D, which implements the resource delivery from 3ds Max to Unity3D.

In addition, because the mesh models of trees, roads, a square, a sphere are directly derived from Unity3D resource packages, they just need to be set up their textures and materials in mesh render component [9].

B. Constructing the VR Scene

At first, the scene layout is designed with the models built in Section 3.1. Then the sky, the collision detection and the audio effects are added to the scene. The mini-map is made to help subjects conduct the experiment. The steps of the VR scene construction are as follows.

1) Layout design: It is known that Unity3D uses the left-hand rectangular coordinate system. In other words, all the models are located in x-z plane. In Unity3D, any object appearing in the scene is bound to add the transform component, which contains three attributes: position, rotation and scaling. Hence, with the appropriate position values being set, the models built in Section 3.1 are placed in the reasonable positions, contributing to designing the scene layout conveniently.

2) Skybox: The sky is an important part of VR scene, which is used to make the scene more realistic. Thus, this system employs the skybox technique to render the sky. The skybox consists of six surfaces with sky textures.

3) Collision detection: Collision detection is indispensable in the VR scene. Otherwise, objects can pass into others, which violates physics rules of our real world and reduces the sense of reality [10]. Unity3D has abundant physics engines, which can simulate the real collision effects and draw a more vivid picture. The 3D car, as a special controlled object, is suitable to add the character controller, which can make the car detect collision and achieve the optimal collision effects. Other mesh objects just need to detect collision. So, they are added with the mesh colliders [11].

4) Audio effects: The audio effects can not only make the system more immersive and interesting, but also can serve as auditory feedback to remind subjects or adjust subjects' moods. So, four kinds of audio are added to the system. Among them, the audio of the car starting, the car driving, and the car colliding with trees or buildings can improve the scene of reality. The colliding audio can also reminder subjects to adjust their motor imagery. The audio of the fireworks displaying can encourage subjects to conduct a better motor imagery. The audio effects are controlled by the script, in which the audio is played by calling the function music.Play() and is stopped by calling the function music.Stop().

Figure 2. VR scene

5) Mini-map: Mini-map shows the positions of the 3D car and the cinema clearly. According to the mini-map, subjects adjust their motor imagery in time to control the 3D car to go to the cinema. Actually, the mini-map mainly depends on the vice camera, which always overlooks the VR scene. In the view of the vice camera, a red sphere and a blue cube respectively become the substitutions of the 3D car and the cinema. In order to follow the movements of the 3D car in real time, the sphere is affiliated to the car [12].

Fig. 2 shows the whole VR scene: the 3D car and the cinema are circled in red. In the lower left corner of the VR scene is the mini-map.

C. Defining Behavior of the 3D Car

The character controller component controls the character behavior more flexibly than the rigid-body component. Thus, the 3D car is added character controller component to control its movement. The character controller component of 3D car simulates a third-person perspective. That is to say that the main camera follows the 3D car forever to show the 3D car current movement clearly. Furthermore, a script is used to control the component. The script needs to inherit the MonoBehaviour class and then the function Start() and the function OnGUI() in it are overridden. The procedures of the definition are as follows:

(1) The move and rotation speeds of the car is defined, which are from the received commands.

(2)In the function Start(), the function GetComponent <CharacterController>() is called to obtain the component object ,which is used to make sure that the controlled object is the car.

(3)In the function OnGUI(), the function Rotate() is called to make 3D car turn left or right. When the car moves forward, firstly, the function transformDirection() is called to get the current direction that the car faces. Then the function Move() is called to realize moving forward of the car.

D. Configuring the TCP/IP Communication and the Timer Functions

1) The TCP/IP communication function: In order to avoid the thread blocking [13], the application system communicates with the signal acquisition and processing system through the asynchronous TCP/IP protocol [14]. The application system acts as the server, while the signal acquisition and processing system acts as the client. The specific communication procedures of the server are as follows: (1) Call the function Socket() to create a listening socket. (2) Call the function Bind() to bind the listening socket with an address, which consists of IP address and port number. (3) Call the function Listen() to monitor connection request of the client. (4) Call the function BeginAccept() to receive the connection request and respond to the request, Then the connection is established. (5) Call the function BeginReceive() to receive data from the client or call the function BeginSend() to send data to the client, completing data transmission with the client. (6)Call the function Close() to close the listening socket when the experiment is over.

2) The timer function: The timer function can not only measure subjects' control effects, but also reflect the classification accuracy of the EEG signals, which contributes to the further research and improvement of the signal acquisition and processing system. The realization of the timer function mainly depends on the function WaitForSeconds(), which makes the system main thread reach a wait state. Once the start button is pressed, the timer is triggered to count down. If the car fails to arrive at the cinema within the given time or reaches in advance, the timer is triggered to stop timing. The figure of the timer changes in real time until the experiment is over.

IV. EXPERIMENT RESULTS

During the experiment, the application system received commands through the TCP/IP protocol to control the car to turn left, turn right or move forward in real time. When the car reached the intersection on its way to the cinema, the subject imagined the movement of his right hand. So, the application system received the command of "MDR0.4", controlling the car to turn right at the rotation speed of 0.4rad/s, which is shown in Fig. 3. Moreover, according to feedback of vision and hearing, the subject adjusted motor imagery timely, thereby adjusting the car action. When the 3D car arrived at the cinema ahead of time, the timer stopped timing. Besides, the audio of the fireworks displaying and the caption-“Congratulations!” were used to encourage subjects. When the car failed to reach the cinema within the limited time, the timer stopped timing and the computer interface appeared a caption-“game over and fighting”.

Figure 3. Experiment result

In a word, the system is available. Meanwhile, the TCP/IP communication and the timer functions are normal. The effective multi-sensory feedback, urges subjects to conduct a better motor imagery, which improves the training effects.

V. CONCLUSION

In this paper, the system developed in Unity3D is a VR simulation application system providing effective visual and auditory feedback. Through the TCP/IP protocol, the system receives commands from signal acquisition and processing system to realize the movement of the 3D car. The VR scene built in the system is easy to arouse the subjects' effective EEG signals with strong immersion and interactivity. Besides, the development of the Unity3D simulated application system costs less money and less time. In a word, the designed system furnishes the BCI system with a good testing and training platform, which promotes the BCI application in daily life.

ACKNOWLEDGMENT

This project is supported by National Natural Science Foundation of China (60975079, 31100709), Innovation project of Shanghai Education Commission(11YZ19), Shanghai University, "11th Five-Year Plan" 211 Construction Project.

REFERENCES

[1] G. Pfurtscheller, C. Neuper, "Motor Imagery and Direct Brain-computer Communication," IEEE Journals and Magazines, Proceedings of the IEEE, vol. 89, Jul. 2001, pp. 1123-1134, doi:10.1109/5.939829.

[2] J. Tomas, S. Junst, and P. Willemsen, "Effectiveness of Commodity BCI Devices as Means to Control an Immersive Virtual Environment," SUI Proceeding of the Symposium on Spatial User Interaction, 2013, pp. 97, doi:10.1145/2491367.2491403.

[3] F. Benedetti, N. C. Volpi, L. Parisi, and G. Sartori, "Attention Training with an Easy–to–use Brain Computer Interface," Virtual, Augmented and Mixed Reality, Applications of Virtual and Augmented Reality, Eds.: Springer International Publishing, 2014, pp. 236-247, doi:10.1007/978-3-319-07464-1_22.

[4] R. Ron-Angevin, A. Díaz-Estrella, "Brain–computer Interface: Changes in Performance Using Virtual Reality Techniques," Neuroscience Letters, vol. 449, Jan. 2009, pp. 123-127, doi:10.1016/j.neulet.

[5] X. Bin, Y. Wenlu, D. Xiao, and C. Wang, "The Training Strategy in Brain-computer Interface," 2010 Sixth International Conference on Natural Computation, vol. 4, Aug. 2010, pp. 2190-2193, doi:10.1109/ICNC.2010.5583993.

[6] F. Lotte, J. Faller, C. Guger, and Y. Renard, G. Pfurtscheller, A. Lécuyer, and R. Leeb, "Combining BCI with Virtual Reality: towards New Applications and Improved BCI," Towards Practical Brain-Computer Interfaces, Eds.: Springer Berlin Heidelberg, Jul. 2012, pp. 197-220, doi:10.1007/978-3-642-29746-5_10.

[7] I. Aswin, M. Shinozaki, "The Investigation on Using Unity3D Game Engine in Urban Design Study," ITB Journal of Information and Communication Technology, vol. 3, 2009, pp. 1 – 18, doi:10.5614/itbj.ict.2009.3.1.1.

[8] J. H. Yu, X. M. Bai, and J. W. Lv, "Complex Object Modeling Method Based on 3ds MAX Platform," Applied Mechanics and Materials, Vol. 184-185, Jun. 2012, pp. 724-727, doi:10.4028/www.scientific.net/AMM.184-185.724.

[9] S. Wang, Z.L. Mao, and H.L. Gong, "A New Method of Virtual Reality Based on Unity3D," International Conference on Geoinformatics, Proceedings of the IEEE, Jun. 2010, pp. 1-5, doi:10.1109/GEOINFORMATICS.2010.5567608.

[10] B.H. Yang, Q. Wang, Z.J. Han, H. Wang, and L.F. He, "Development of a BCI Simulated Application System Based on DirectX," Practical Applications of Intelligent, Eds.: Springer Berlin Heidelberg, Jul. 2014, pp. 813-822, doi:10.1007/978-3-642-54927-4_77.

[11] R. Leeb, M. Lancelle, V. Kaiser, D. W. Fellner, and G. Pfurtscheller, "Thinking Penguin: Multimodal Brain-computer Interface Control of a VR Game," IEEE Transactions on Computational Intelligence and AI in Games, vol. 5, Jun. 2013, pp. 117-128, doi:10.1109/TCIAIG.2013.2242072.

[12] F. Sun, D. Hu, H. Liu, H. Zhang, J. Liang, Y. Liu, H. Wang, and L. Zhang, "An Iterative Method for Classifying Stroke Subjects' Motor Imagery EEG Data in the BCI-FES Rehabilitation Training System," Foundations and Practical Applications of Cognitive Systems and Information Processing, Eds.: Springer Berlin Heidelberg, 2014, pp. 363-373, doi:10.1007/978-3-642-37835-5_32.

[13] R. Emardson, P.O. hedekvist, M. Nilsson, and S.C. Ebenhag, " Time and Frequency Transfer in an Asynchornous TCP/IP over SDH-network Utilizing Passive Listening," Frequency Control Symposiun and Exposoyion, Proceedings of the IEEE, 2005, pp. 908-913, doi:10.1109/FREQ.2005.1574054.

[14] G. Carofiglio, L. Muscariello, "On the Impact of TCP and Per-flow Scheduling on Internet Performance," IEEE/ACM Transactions on Networking, vol. 20, Apr. 2012, pp. 620-633, doi:10.1109/TNET.2011.2164553.

Research on Key Technologies Base Unity3D Game Engine

Jingming XIE
Information Engineering Institute
Guangzhou Panyu Polytechnic College
Guangzhou, China
xjmadam@hotmail.com

***Abstract*—Game engine is the core of game development. Unity3D is a game engine that supports the development on multiple platforms including web, mobiles, etc. The main technology characters of Unity3D are introduced firstly. The component model, event-driven model and class relationships in Unity3D are analyzed. Finally, a generating NPCs algorithm and a shooting algorithm are respectively presented to show common key technologies in Unity3D.**

***Index Terms*—game engine, Unity3D, NPC algorithm, shooting algorithm**

I. UNITY3D GAME ENGINE

A game engine provides a main framework and common functions for developing games, which is the core of controlling games. Since the first advent of the Doom game engine in 1993, game engine technology has experienced nearly 20 years of evolution. Game engine initially only supported 2D, now fully supports 3D, lifelike images, massively multiplayer online game, artificial intelligence and mobile platforms. Some representative game engines are Quake [1], Unreal Tournament [2], Source [3], BigWorld [4] and CryENGINE [5], etc. The internal implementation techniques in Game engines have some differences, but their ultimate goal is the same to improve the efficiency of game development. To comprehensively grasp the general development ideas of a game, it is necessary for choosing a typical game engine to study deep.

Unity3D is a popular 3D game engine in recent years, which is particularly suitable for independent game developers and small teams. It mainly comprises eight products such as the Unity, Unity Pro, Asset Server, iOS, iOS Pro, Android and Android Pro [6]. Without writing complex codes, programmers can quickly develop a scene by using the visual integrated development environment of Unity3D. In the popular iPhone game list, the games developed by Unity3D take a large proportion, such as Plants vs. Zombies, Ravensword:The Fallen King. In particular, Unity3D also provides the Union and Asset Store selling platforms for game developers.

Unity3D has special advantages in easily programming a game. For example, platform-related operations are encapsulated in its internal, the complex game object-relations are managed by different visual views, and JavaScript, C # or Boo scripting languages are applied to program a game. A script program will be automatically compiled into a .NET DLL file, so the three scripting languages, in essence, have the same performance, their execution speed is 20 times faster than traditional JavaScript. These script languages have good cross-platform ability as well. That means developers can deploy games on different platforms such as Windows, Mac, Xbox 360, PlayStation 3, Wii, iPad, iPhone and Android. In addition, games can run on the Web by installing a plug-in.

Another feature of the Unity3D is that game resources and objects can be imported or exported in the form of a package, which can easily make different game projects share development works. Therefore, using package can greatly improve efficiency in game development. In addition to resource material files, specific functions can be packaged, such as AI, network operation, character control, etc.

II. UNITY3D GAME ENGINE INTERNAL ANALYSIS

A. The Component Model

The component model is applied in the Unity3D game development, which provides a scalable programming architecture. Game function modules can be reused conveniently in this component model. Each entity in a game scene is called as a GameObject. A GameObject represents a game object, which has the characteristics of a container. According to the needs of a game, many different components are added into a GameObject. A component can be taken as a collection with a group of related functions, which can be accessed through an interface. For example, a script is a component and its role is to offer a logical operation on a game object, and the Box Collider component in Unity3D specifically provides a support for game objects collision detection. Unity3D has many predefined components. Programmers can combine some of them to create a feature-rich GameObject. The Figure 1 shows the ideas of the component model in Unity3D.

The Figure 2 shows the tree hierarchy for organizing objects in Unity3D game project. A game is composed of one or more scenes, each scene includes one or more GameObjects, and moreover, every GameObject is composed of some components or child GameObjects.

In game development, besides directly using GameObjects predefined in Unity3D, programmers can create an empty GameObject with the information about position, rotation and scale of an object, and then add scripts or other components

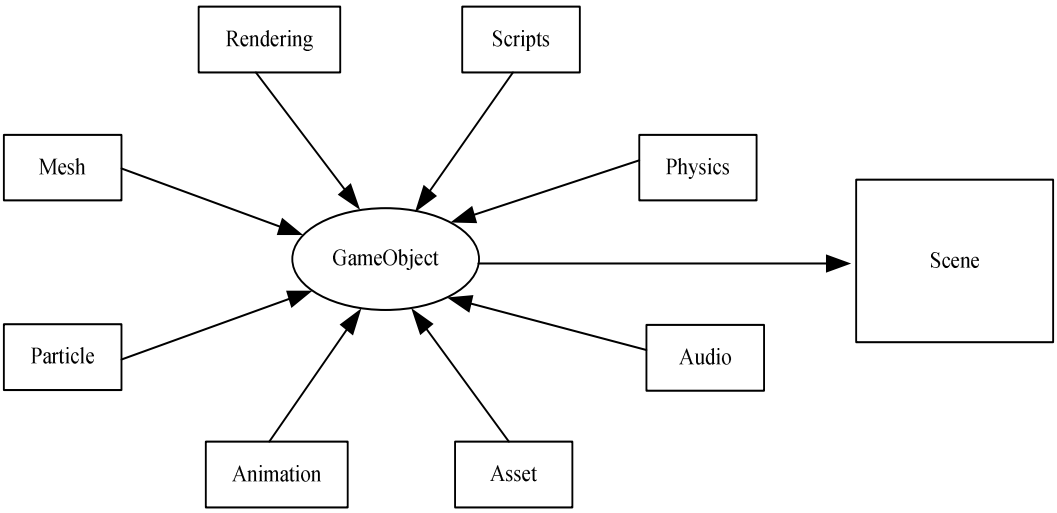


Figure 1 The Unity3D component model

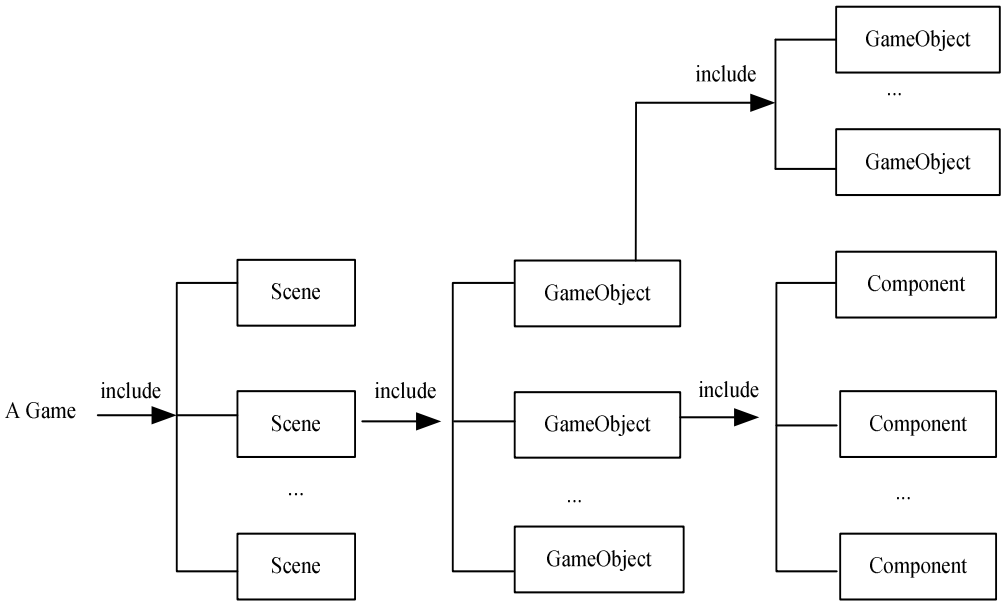


Figure 2 The Unity3D game project hierarchy

into it. In order to facilitate the same type of game object management, Unity3D affords the Prefab that is a technology like a template. A Prefab can contain both objects and game resources such as 3D models. When the same type of game objects need to be created, a Prefab can be used in this situation. All GameObjects will be updated simultaneously when its Prefab is changed. The above mechanism of Prefab can greatly improve the maintenance efficiency of a game.

To observe the impact of an object's state on a game, programmers can dynamically change the configuration parameters of a component when the game is running. After the game exits, all its initial parameters will be reset. In order to better grasp the impact of game elements, details of differences between game design and actual running affect can be discovered by using the game view and scene view at the same time.

B. The Event-Driven Programming

There are a number of scripts in a game. A script is a class controlling the behavior of a game, which should inherit a base class called MonoBehaviour. MonoBehaviour defines common event triggering methods. When a predefined event occurs, an appropriate method will be automatically executed. The Figure 3 lists the important event methods in Unity3D and their execution order relationships.

The Awake() method is executed only once in the life cycle of a script instance, when the script is called. The Awake() method can be used to initialize variables when a game starts. After all GameObjects are initialized, this method will be executed. The execution order of each Awake() method in GameObjects is random. Consequently, it is safe for referencing other GameObjects in the Awake() method without causing a null object reference.

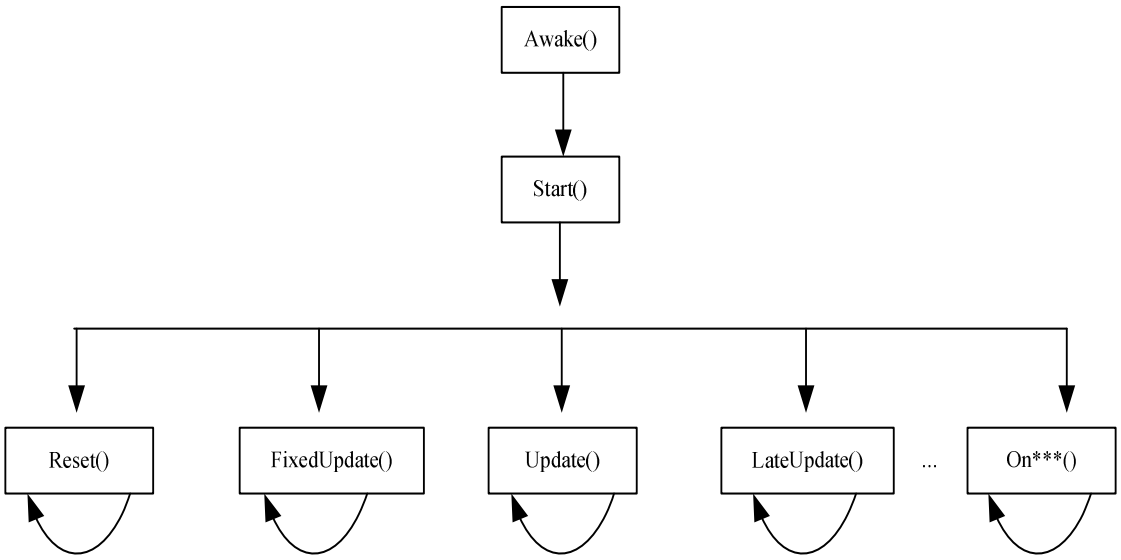


Figure 3 The event-driven model

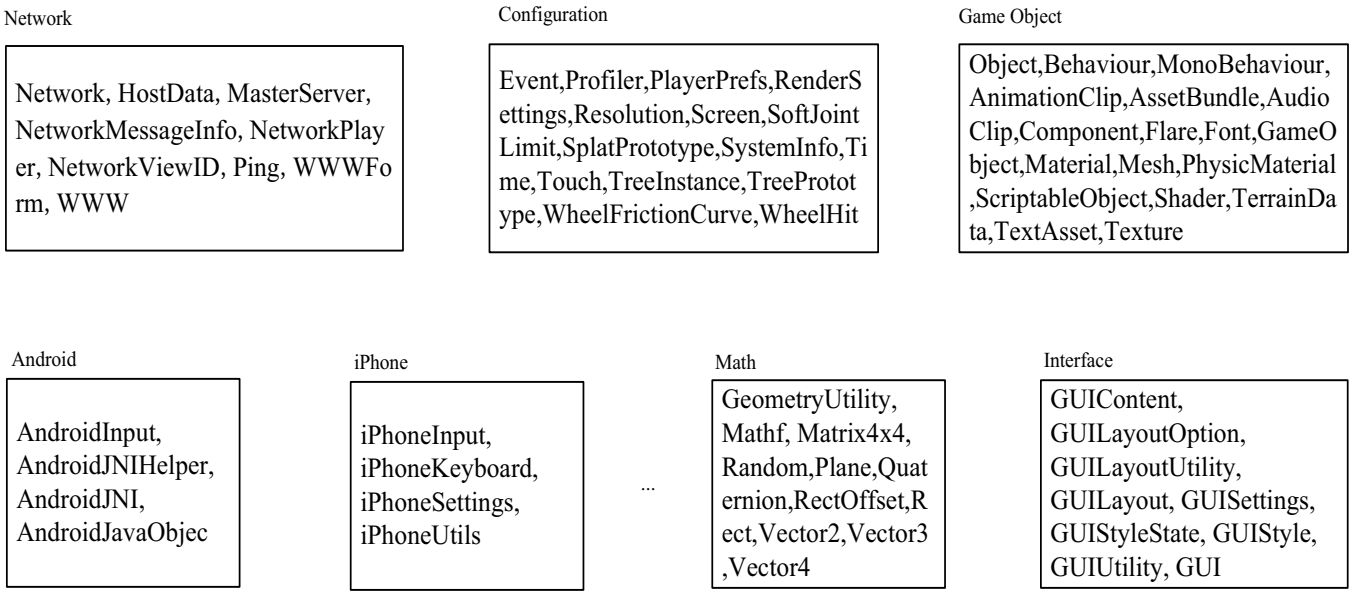


Figure 4 The distribution of main classes in Unity3D

The Start() method is executed only once like the Awake() method, and it can also be used to initialize variables. The difference between the two methods is that the Start() method is executed only when the script instance is enabled. When it needs to order initialization operations, the Start() method can be utilized to delay initializing codes.

After the execution of the Awake() and Start() methods, other event methods can be recycled to trigger in the life cycle of a game. Methods such as Reset(), FixedUpdate(), Update(), LateUpdat() can be executed in multiple times. For example, the Update() method will be called when each frame is rendered. Its implementation number depends on frames per second. It is recommended that logic operations should be put in this method.

C. The Summary of Classes

A game is controlled by programming scripts to access the internal of Unity3D. There are many system classes in Unity3D, which are useful in scripts. There are two kinds of classes. One is named as runtime classes that are applied to operate game objects, another is named as editor classes that are applied to modify Unity3D tools such as plug-ins and view information.

Although a class in Unity3D has small number of methods, it is not only easy to use and powerful to develop games. Understanding methods function is the key of using them in a game program.

Some classes represent visual game objects, which all inherit from the Object class. Ojbects from these classes can be directly showed in a game scene and players will see their effect. These objects can be created, updated and destroyed in game codes. These classes relate to the camera, sound, animation, particles, rigid, material, texture, color, font, GUI text, GUI image, light, sky boxes, etc.

Other classes are used as auxiliary operations, such as network processing, game configuration, math geometry, resource management and special classes for mobile platforms.

According to function of classes, nearly 200 runtime classes are categorized into the Figure 4.

III. THE RESEARCH ON KEY TECHNOLOGIES

A. A Dynamic Algorithm of Randomly Generating NPCs

NPC is the abbreviation of non-player character. That the number and location of NPCs are dynamically changed can keep players strong interesting on a game and extend the vitality of the game. NPCs can be designed by the difficulty and logic of a game. The following will propose a dynamic algorithm of randomly generating NPCs, which can be called when some NPCs need to be created in a game.

1. Select a kind of NPC that has not been yet handled, it is named as *t*;

2. *bCreate*=0;

3. Determine whether the kind of NPC *t* should appear:

If(*random.value*>*tRValue* && *NPCrule*==1){

//determine by a probabilistic method, *tRValue* is the appearing probability for the kind of NPC *t*.

bCreate=1;

}else if(*existNum*<*tEValue* && *NPCrule*==2){//Existing number of the kind of NPC *t* is below a certain threshold value *tEValue*.

bCreate=2;

}else if(*somethinghappen* && *NPCrule*==3){

//determine by an event in which some conditions are satisfied.

bCreate=3;

}

4. If (*bCreate*>0)

tNum=Random.Range(*tMinValue*, *tMaxValue*);//determine how many NPCs of the kind of *t* will be created.

5. while(*tNum*>0){

6. Set the location of a NPC of type *t*:

position=*reference.transform.position*+*Vector3*(*x,y,z*);//*x, y* and *z* are randomly generated;

7. Copy a NPC of type *t* from the template by using the instantiate method and set the behavior of its action;

8. *tNum*--;}

9. Determine whether it is necessary to create other kinds of NPCs, if it is necessary then goto step 1, else end this algorithm.

Now, let's analyze the algorithm. The random operations are applied in the step 3, 4 and 6 to control the number and location of new NPCs. The step 3 determines whether certain types of NPCs should be generated by a variety of rules. In the step 4, difficulty of a game can be adjusted by regulating *tMinValue* and *tMaxValue*. The numbers of NPCs are dynamic by the corresponding kind of NPC. In the step 6, the protagonist or the pre-set coordinates in a scene can be taken as a reference point. An empty GameObject with special deployment mark can be designed as a reference for the location of NPCs.

It often requires creating a number of NPCs in a game. We introduce a simple solution of enabling them to have perfect distribution.

1. Manually deploying a number of referencing coordinates for NPCs in the scene by special strategies. The distribution of references should have a well density.

2. Classifying these references by setting the tag properties in Unity 3D to prepare basic data for deployment. Programmers can quickly find a group of GameObjects with a certain tag by the *FindGameObjectsWithTag* method.

3. Deploying NPCs by these references and deployment strategies.

B. A Shooting Algorithm

There are mainly two ways of firing bullets in game development. One is that a true 3D object is modeled by decorating a bullet with texture and color. This approach needs to control the flight path of a bullet and destroy the bullet when it is out of bounds. Another method is to use ray tracing technology. That the light collides with an object means a bullet hits the object. In order to get more realistic effects, cartridge cases are pop-upped or sparks are twinkled at the gunpoint when bullets are fired. Moreover, drawing light between the launch point and target point can more realistically simulate the firing of bullets.

In a 3D shooting game, a sight is usually provided to players. Players can move a mouse to control it for firing. The principle of this technique is to place a graphic sight in the center of the screen. Moving the mouse is equal to move the camera, which will cause the illusion of the sight moving in the screen. In fact, the coordinates of the camera are real target coordinates in game codes.

In a first-person game, a main camera is set as a sub-object of the protagonist [7], so they can move together as a whole. There are four directions in the mouse movement, namely left, right, up and down movement. The following will discuss how to calculate them.

The mouse moving left or right is equivalent that the camera rotates the Y axis. The following formula is about the way of calculating a 3*3 rotation matrix $Ry(\theta)$. θ is the horizontal movement angle of the mouse, which can be got through the magnitude and direction of mouse horizontal movement. *Input.GetAxis*("*Mouse X*") method in Unity3D can help programmers do that in a simple way. This method will return a value between -1 to 1. The *sensitivityX* represents the sensitivity value for the mouse, which is usually assigned to 15. To the formula (1-3), the *Rotate* method in Unity3D can be directly applied to rotate an angle.

$$mouseX = Input.GetAxis("MouseX"); \quad (1-1)$$

$$\theta = mouseX * sensitivityX; \quad (1-2)$$

$$Ry(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \quad (1-3)$$

Finally, we discuss the calculation principle about moving the mouse up or down. A camera moving up or down is equivalent that the head of a person (parent node) rises up or down. Moving values are in a fixed angular range. For example, angles are between -60 degrees to 60 degrees. From the mathematical point of view, it relates to the rotation of the parent's transformation and actually it can be expressed by Euler angles [8]. Euler angles represent three angles around the *X*, *Y*, *Z* axis rotation, which are used to simulate the rotation of local coordinates. The advantage of applying Euler angles is more intuitive, easy to understand and control than a matrix. The initial value of *rotationY* in formula (2-1) is 0, later it cumulates the magnitude for mouse moving up and down. The *p* in formula (2-3) is the Euler angle rotating around the *X* axis. New Euler angles can be assigned to *transform.localEulerAngles* in Unity3D.

$$rotationY += Input.GetAxis("MouseY") * sensitivity; \quad (2-1)$$

$$rotationY = Mathf.Clamp(rotationY, minnumY, maxnumY); \quad (2-2)$$

$$p = -rotationY \quad (2-3)$$

REFERENCES

[1] Quake. http://www.idsoftware.com/games/quake/.

[2] Unreal Tournament. http://www.unrealtournament.com.

[3] Source. http://source.valvesoftware.com/.

[4] BigWorld. http://www.bigworldtech.com/index/index.php

[5] CryENGINE . http://www.crytek.com/cryengine.

[6] Unity. http://unity3d.com.

[7] Graham McAllister. Creating a First Person Shooter (FPS). http://download.unity3d.com/support/resources /files/FPS_Tutorial_1.pdf.

[8] Fletcher Dunn, lan Parberry, 3D Math Primer for Graphics and Game Development. Tsinghua University Press, 2005.

Toward Supporting Stories with Procedurally Generated Game Worlds

Ken Hartsook, Alexander Zook, Sauvik Das, and Mark O. Riedl

***Abstract*—Computer role playing games engage players through interleaved story and open-ended game play. We present an approach to procedurally generating, rendering, and making playable novel games based on *a priori* unknown story structures. These stories may be authored by humans or by computational story generation systems. Our approach couples player, designer, and algorithm to generate a novel game using preferences for game play style, general design aesthetics, and a novel story structure. Our approach is implemented in GAME FORGE, a system that uses search-based optimization to find and render a novel game world configuration that supports a sequence of plot points plus play style preferences. Additionally, GAME FORGE supports execution of the game through reactive control of game world logic and non-player character behavior.**

I. INTRODUCTION

Computer Role-Playing Games (CRPGs) are a genre of games in which a player enters a virtual game world in the role of a story world character that embarks on a number of quests or missions, ultimately achieving some overarching goal. This genre exemplifies the interleaving of story and open-ended adventuring in games. A canonical CRPG example is Nintendo's Zelda™. CRPGs are content-heavy games, requiring a large world, sophisticated story lines, numerous NPCs, and numerous side-quests. Due to their nature, CRPGs typically have low *replayability* – the replay value drops off significantly after the game play affordances of the story and world are exhausted.

As a consequence of their content-heavy nature, CRPGs are prime candidates for the application of *Procedural Content Generation* (PCG). Procedural content generation is any method that creates game content algorithmically, with or without the involvement of a human designer. There are two broad uses of PCG in games: design time assistance of content creation, and run-time adaptation of gameplay. On one hand, PCG may make the development of content-heavy games such as CRPGs cheaper and easier by semi-automating the construction of some game content. On the other hand, personal information about the player, such as the player's wants, desires, preferences, and abilities – information that cannot be known at design time – can be used to personalize the story and world of the game so as to maximize pleasure and minimize frustration and boredom.

In prior work (cf., [6]), we explored the generation of customized game plots irrespective of the world environment in which they would play out. CRPG game plots were automatically adapted to individual players based on a set of preferences over the types of tasks the player likes to perform in role playing games. However, without a game world, a story cannot be played. Will any world do? We assert that the game world must service the story by providing an environment with the right kinds of places in a reasonable configuration. Since CRPGs also involve adventuring in between significant story events, the world should additionally cater to the player's individual preferences for amount and style of adventuring. Prior work has not addressed the construction and coordination of procedurally generated story and world.

In this paper, we explore the question of how to procedurally generate a complete, executable CRPG, given an *a priori* unknown story and a set of player preferences. The challenge of generating a CRPG, which distinguishes it from many other game genres, resides in the central role of story. Story is a common element in many computer game genres used to establish context, motivate the player to perform certain tasks, and shape the player's overall experience inside the virtual world of the game. Storytelling in computer games can be distinguished from other forms of storytelling by the close relationship between the sequence of narrative events and the virtual world – in games, story is the progression of the player through space [1],[2],[5]. From the perspective of game design, level design shapes the story experience, although game players may not perceive the close relationship between story and space.

We present a technique for automatically generating a fully playable CRPG based on a story – created by a human or computational story generator – and information about the player's play style preferences. The contributions of our work are as follows:

1) A technique for generating and rendering a CRPG game world based on play style preferences and designer constraints.
2) Integration of game plot and game world.
3) A general game execution engine that can puppet non-player characters and other plot-relevant aspects of the game based on parameters of the story and specifics of the world.

Our procedural CRPG generation techniques are implemented in the GAME FORGE system, which is described in the remainder of this paper.

All authors are with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA (email: {khartsook3, azook3, sdas7, riedl}@gatech.edu).

978-1-4577-0011-8/11/$26.00 ©2011 IEEE

II. Background and Related Work

Computer role-playing games (CRPGs) employ a complex combination of story and game world in order to orchestrate player experience. The CRPG game playing experience, in its basic form, interleaves periods of time when the player is engaged in plot-centric activities and open-ended adventuring. Periods of engagement with the story typically involve interacting with non-player characters (NPCs), solving puzzles, and battling bosses. In between story plot points, the player is encouraged to engage in open-ended adventuring where he or she can explore the game world, fight randomly encountered enemies, collect items and treasure, advance his or her character's skills and abilities, and go on side-quests – quests that do not significantly progress the main plot line. The game world is integrated with the story to provide concrete experiences and a space for players to engage in adventures.

CRPG players often have particular preferences for both the story they experience in a game and the types of activities they engage in within the game world. Players may prefer to experience particular types of story content, such as rescuing a princess from a dragon or retrieving treasure out of a booby-trapped cave. In addition, players may have different attitudes toward how much time they would like to spend exploring the world, going on side-quests, engaging in combat with random encounters, and acquiring items. Current CRPGs typically handle these diverse interests by providing a "one size fits all" story and world that attempt to provide an optimal experience for all types of player preferences over both story and world. That is, game designers attempt to author a broad range of content that is all included within a single game world and story.

Research on procedural generation of computer game content has become popular in recent years. *Search-based procedural content generation* (SBPCG) involves the use of search-based algorithms to explore a space of game content for those meeting a set of evaluation criteria [13]. Examples of content generated using SBPCG include: decorations of virtual environments such as trees and other vegetation; landscapes and 3D environmental geometry; weapons; opponents; game levels; and quests [13],[16]. Work on SBPCG has focused mainly on non-story-driven game genres such as platformers, racing games, and shooters. Moreover, a majority of SBPCG work has focus on generation of a single aspect of the game, rather than integrating multiple aspects for a full game.

Game level generation is the procedural construction of an environment, or space, through which the player must navigate. Level generation is most commonly used in non-story-driven games where the goal is to reach the end of the environment (cf., [9],[10],[11],[12]). In games in which story is analogous to movement through space, level generation may result in the creation of story by constraining the way players move through an environment [2],[4]. These simulation-based methods, however, do not support the orchestration of a story as complicated as those in CRPGs and also typically do not incorporate player preferences into the world model.

Others have emphasized methods for incorporating given world elements into the generation process. Tutenel et al. [14] use semantic information and a constraint-based layout solver to create game worlds meeting designer aesthetics. We have employed elements similar to layout systems in our use of probabilistic distributions tied to particular types of environments while employing a greedy search algorithm for world layout. There is a trade-off between constraints, which require knowledge engineering effort, and algorithmic flexibility.

Dormans [3] has addressed the bipartite nature of story and space. This work uses a context-free shape grammar to arrange a cave to support an individual mission (story). In contrast, we employ an optimization process to avoid the problems of over-generation associated with context-free grammars and to incorporate player preferences.

Many efforts have recently aimed to incorporate player preferences in the field known as *experience-driven procedural content generation* [16]. Within this framework our system uses a subjective player experience model by directly asking about player preferences. Stories and worlds are evaluated directly, mapping given content and structures to quality values based on the player model. Our content representation involves multiple levels of abstraction and uses a non-exhaustive search process for generation. Currently Game Forge does not provide a closed loop where feedback at either the story or world levels influences subsequent rounds of content generation.

III. Story Representation

Game Forge takes story content produced by a user, computational story generation system, or other means and builds a world that supports the story. The story must be provided as a list of plot points. A *plot point* is a high-level specification of a period of time with a semantic and recognizable meaning. Examples include fighting and killing enemies, buying and selling objects, conversing with NPCs, engaging in social activities with NPCs (e.g., marrying a princess), and solving puzzles. The type of plot point is parameterized to reference NPCs and locations appropriate to the story. Each plot point type also includes a reactive script that provides reactive execution logic to control NPCs and modify the game world as appropriate to the plot point. See Section V for more information about reactive scripts.

Plot points reference places by name where the action is to occur. Named places are of a particular type, e.g., "palace" might refer to a particular instantiation of a location of type "castle". As with plot point types, place types are pre-defined. It is important to note that, although plot points reference named places, the story structure itself does not contain information about the spatial layout of these locations; it is the responsibility of Game Forge to automatically determine the spatial layout of the world.

In addition to a sequence of plot points, Game Forge also

1. **Take** (paladin, water-bucket, palace)
2. **Kill** (paladin, baba-yaga, water-bucket, graveyard1)
3. **Drop** (baba-yaga, ruby-slippers, graveyard1)
4. **Take** (paladin, shoes, graveyard1)
5. **Gain-Trust** (paladin, king-alfred, shoes, palace)
6. **Tell-About** (king-alfred, treasure, treasure-cave, paladin)
7. **Take** (paladin, treasure, treasure-cave)
8. **Trap-Closes** (paladin, treasure-cave)
9. **Solve-Puzzle** (paladin, treasure-cave)
10. **Trap-Opens** (paladin, treasure-cave)

Hero (paladin), NPC (baba-yaga), NPC (king-alfred), Place (palace), Place (graveyard1), Place (treasure-cave), Thing (water-bucket), Thing (treasure), Thing (ruby-slippers), Type (baba-yaga, witch), Type (king-alfred, king), Type (palace, castle), Type (graveyard1, graveyard), Type (treasure-cave, cave), Type (water-bucket, bucket), Type (ruby-slippers, shoes), Type (treasure, gold), Evil (baba-yaga) …

Fig. 1. A simple story represented as a list of plot points (top) and an initial state (bottom).

requires an initial state: a list of propositions that describe story-specific details about the story world. The initial state includes information about NPCs, objects, and places that are referenced by the plot points. For example, information about NPCs includes their names, character classes, and other attributes. The initial state provides the type of each of the referenced places as a set of propositions. Proposition types are general and pre-specified. Figure 1 shows a simple example story with its corresponding initial state.

Our story representation is consistent with AI planning-based story generation systems such as [6] and [8] that either generates plots from scratch or adapt novel plots from existing plots. However, the representation is simple enough that humans – with the assistance with authoring tools – can also author their own stories by assembling and parameterizing known plot point types.

GAME FORGE was implemented to create game worlds corresponding to stories generated by the game plot adaptation system described by Li and Riedl [6]. The game plot adaptation algorithm takes an existing hand-authored game plot – represented as a partial-order plan [15] – and a set of preferences about the types of things the player likes to do in CRPGs and searches for a sequence of changes that transforms the original game plot into a new game plot that meets the player's preference specifications. The game plot adaptation algorithm is responsible for adding and removing plot points until success criteria are met. Once the search is complete, a potentially novel story structure may exist. This story structures is converted into the GAME FORGE story representation (the translation from partial-order plan to our story representation is trivial and straight-forward), and sent to GAME FORGE to render and execute the game.

Note that the GAME FORGE story representation currently only handles linear stories. Computer games typically have a single main storyline that constitutes the set of plot points that are necessary and sufficient for completion of the game. GAME FORGE currently only concerns itself with this main storyline. CRPGs, however, often utilize side-quests, plot points that are optional and do not causally link back in with the plot points in the main storyline. GAME FORGE supports side-quests by generating portions of the game world that branch off from essential parts of the world based on the player's stated preference for adventuring. These branching spaces could be areas of the world where side-quests unfold. GAME FORGE, however, does not generate the story content of the side-quests; the creation of the story content for a side-quest is the responsibility of an external agent and may occur before game world generation – providing input into the required size and shape of the world – or after game world generation – to utilize optional portions of the world.

IV. GAME WORLD GENERATION

We solve the problem of automatically designing, building, and rendering a completely functional game world for a story. Recall that CRPGs interleave plot points and open-ended game play; the game world to be generated must ensure a coherent sequence of plot points are encountered in the world. In the process of generating the world, our approach also attempts to incorporate preferences for game play style into the configuration of the world. The problem can be specified as follows: given a list of plot points of known types, referencing locations of known types, generate a game world that allows a linear progression through the plot points and supports user play style preferences. GAME FORGE is specifically targeted to CRPGs, but demonstrates how other heavily story dependent game genres can be supported.

To map from story to space, GAME FORGE utilizes a metaphor of *islands* and *bridges*. Islands are areas where critical plot points occur. Each event in the plot is associated with a location. For example, a story may involve plot points located in a castle, graveyard, and a cave. The game world generator parses the generated plot and extracts a sequence of locations, each of which becomes an island. For example, the story in Figure 1 plays out in three locations: a castle (plot points 1, 5 and 6), a graveyard (plot points 2 through 4), and a cave (plot points 7 through 10). Bridges are areas of the world between islands where non-plot-specific game play occurs, such as random encounters with enemies and discovery of treasure. Bridges can branch, meaning there can be areas that the player does not necessarily need to visit in the course of the story. The length of bridges and the branching factor of bridges are dictated by a player model.

In addition to the hard story constraint that there are specific islands that appear in the world in a specific sequence, we treat the generation of the game world environment as an optimization process, balancing two competing sets of requirements:

1) **Game world model:** Captures the designer-specified believable transitions between environment types as the player moves through the game world. For example, one would not expect to step directly from a castle into a mountain lair without first traversing through mountain terrain.
2) **Player model:** Captures player-indicated play style preferences. Play style preference is represented by a

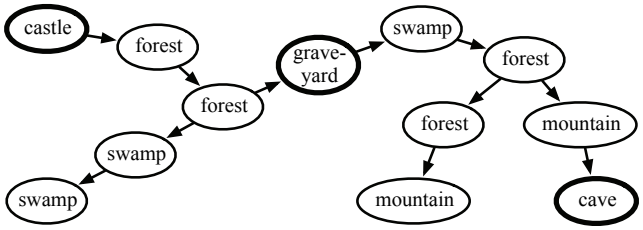


Fig. 2. An example space tree. Islands are denoted by bold outlines.

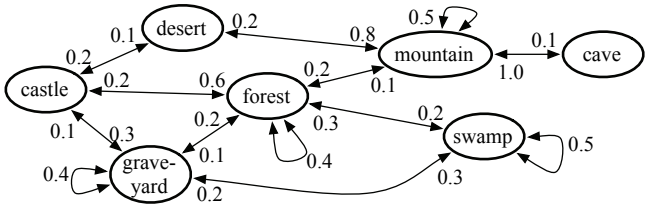


Fig. 3. An example of an environment transition graph.

number of parameters capturing average bridge length (correlating to overall size of the world), linearity of the world (how often there are sidepaths), likelihood of random enemy encounters, and likelihood of finding treasure. The world generation process only uses the parameters associated with bridge length and side paths; the others control gameplay execution.

The game world model is provided by the designer to capture his or her intuitions about believable game worlds. The player model is currently provided directly by the player who answers some simple questions about how much adventuring he or she likes to perform, as well as how much combat and treasure to incorporate. The plot points, manifesting as islands, act as a third, implicit set of requirements. GAME FORGE generates the game world through the use of a genetic algorithm (GA) guided by a fitness function incorporating the above requirements.

A. Genotype Representation

We represent a game world genotype as a space tree, a tree data structure in which each node represents a portion of the world. Each node has an environment type (e.g., castle, field, forest, mountain) that determines what that region of the world should look like in terms of the type of visual qualities, graphical objects and decorations to be found in that region. Space tree nodes also record whether a region is an island or a bridge node. Figure 2 shows an example space tree with islands that correspond to the story shown in Figure 1. Island nodes will always occur along a single path through the tree and islands will always be encountered in the order they are first referenced in the input story.

B. Fitness Function

We employ evaluation criteria based on a game world model and player model. The game world model constrains generation to more natural worlds, while the player model guides generation towards the expressed preferences of a given player. The game world model evaluates the adjacency of environment types along bridges and islands based on an environment transition graph, in which nodes specify types of environments and links indicate the designer-specified probability for two environment types to be adjacent. For example, there is a high probability of a cave being adjacent to a mountain, but a low probability for a forest to be adjacent to a mountain. Figure 3 shows an example of an environment transition graph. Sequences of bridges that more closely match the model are rewarded, while those that incorporate more divergent connections are penalized. Note that while there can be zero probability of two environment types being adjacent according to the transition graph, it is possible for regions of those types to be adjacent in the space tree – it scores very poorly. The game world model encourages game worlds that are believable according to design principles, regardless of personal player preferences.

The player model, on the other hand, drives generation toward game worlds that match expressed player preferences. Bridge length and "branchiness" of the space tree are both evaluated according to this model. The player model specifies a range of bridge lengths desirable to the player, allowing for more densely placed islands (and thus plot points), or longer sojourns between regions encouraging more adventuring through exploration and time spent being lost. Branchiness controls the likelihood of sidepath generation. Players that prefer greater branchiness will be more likely to have side paths from the main course of the plot, allowing exploration to seek additional battles or rewards as any time the player is on a bridge area there is a probability of random encounter with an enemy. The player model skews game world generation towards a desired level of content density and linearity, capturing the unique preferences of individual players.

The fitness of a space tree is the linear sum of feature penalties. The features are: average bridge length, average length of sidepaths, average number of sidepaths per bridge, total number of sidepaths, total number of nodes, environment transition probability, and environment transition variance. Each feature is the *penalty* as distance from a target value. Target values for bridge length and sidepath features are derived from a few simple questions to the player at initialization time. The penalty for environment transition probability is computed by comparing the actual environment transition probability sampled from the space tree to values in the environment transition graph. Because the environment transition probability feature leads the GA to predictable transitions, environment transition variance forces the system to consider transitions that vary from the ideal.

C. Genetic Algorithm Implementation

Our GA starts with an initial population of randomly generated candidate space trees; each is created from a list of islands and pseudo-randomly constructed bridges between each adjacent pair of islands based on the player model and world model. Our GA halts when any individual's fitness is less than a given percentage of the total theoretical possible

```
Let Stock ← Generate initial population
While no individual in Stock is less than or equal to threshold do
   Let Children ← Copy individuals from Stock
   Determine temperature buckets for individuals in Children:
      (10%ile, 25%ile, 50%ile, rest)
   Replace 10% least fit in Children with new individuals
   Foreach individual i in Children do
      If i is chosen according to crossover probability do
         Let j ← Randomly select individual with same temp.
         i,j ← Crossover(i, j)
      End if
      i ← Mutate(i)
   End Foreach
   Survivors ← ∅
   Foreach individual i in Children do
      Let j ← parent of i in Stock
      Survivors ← Survivors ∪ {pick i or j based on fitness}
   End Foreach
   Stock ← Survivors
End While
```

Fig 4. The game world generation algorithm.

penalty. The success threshold can be raised or lowered to increase or decrease the tolerance for variability and randomness; we have experimented with 5% and 7.5% thresholds. Our GA also halts if more than 100 generations pass without significant improvement of the fitness of the best space tree.

To perform a mutation, we randomly choose a node in the given candidate tree to mutate. We then randomly select one of three possible mutations and attempt to mutate the chosen node. If the mutation is successful, we return; otherwise, we chose another node randomly and repeat the process until a mutation succeeds. The possible mutations are:

1) **Addition:** Nodes can be added to a tree to create side-paths (branching) or extend the length of an existing path (extension).
2) **Deletion:** Nodes can be deleted by removing them from their parent's child list and making the deleted node's children refer to its parent as their parent. Deletion fails if the node is an island or the parent of the deleted node has more than two children (as described below this constraint increases the probability that the space tree can be mapped to a 2D world).
3) **Environment Type Change:** The environment of a non-island region can be mutated to any other viable environment, according to the environment transition graph. This may fail if no alternative environment types are possible.

We also use a means of crossover breeding. All space trees share a common set of island nodes in the same order, while differing in the bridges between these islands. In crossover, we randomly select two candidate space trees and randomly exchange some number of bridge nodes between them. The GA pseudocode is shown in Figure 4.

We employ three techniques to maintain diversity in the candidate population while driving the process towards improvement: tournament selection, temperature buckets, and bottom culling. In tournament selection, a space tree and its mutation are compared using the fitness function, with the more fit solution kept for the next generation. That is, each member of the subsequent population must be more fit that its parent. Temperature buckets divide candidate space trees into groups based on their fitness ranks. Temperature is used to describe the mutation rate – higher temperature indicating a greater probability of mutation – and to segregate the population to protect low-fitness solutions. Higher quality solutions have lower temperatures, allowing fine-tuning of high quality game worlds while exploring more broadly using low quality solutions at higher temperatures. Bottom culling involves removing the bottom 10% of the candidate population each iteration and replaces them with randomly generated trees. The cull-and-replace promotes diversity in the stock. Newly generated candidates are placed in the highest temperature bucket where they will be competitive with other candidates in the bucket, yielding offspring that bubble up to lower temperature buckets and thus exploring new parts of the search space.

D. Graphical Realization: From Genotype to Phenotype

The genotype of a game world is the space tree that describes the regions of the world, their environment types, and their adjacency relationships. The phenotype manifestation is a graphically realized virtual world that can be navigated by the player's avatar. Our graphical realization process creates the visual representation of the world that the player will directly interact with. Our graphical realization process builds a 2D world reminiscent of classical CRPG games such as Zelda™. While there are many ways to realize the world, we focus on 2D as a means of demonstrating the general approach of preference plus design constraints in search-based optimization.

For our implementation of GAME FORGE, two issues must be addressed for graphical realization. First, the graph must be mapped to a 2D grid. Second, the grid must be tiled. That is, each grid space receives a graphically rendered tile. The virtual world is a grid where each grid location contains one tile, a 40 x 40 square of pixels. To make graphical realization easier, we hierarchically segment the tiles according to screens and regions. A screen consists of 34 x 16 tiles and is the largest set of tiles that can displayed at once. A region is 3 x 3 screens; there is one space tree node per region.

A recursive backtracking algorithm maps the space tree to regions on the grid. We use a depth-first traversal of the tree, placing each child adjacent to its parent on a grid. In order to prevent an algorithmic bias toward growing the world in a certain direction (e.g. from left to right), our algorithm randomizes the order of cardinal directions it attempts to place each child. To minimize the likelihood that there is no mapping solution because nodes must be mapped to the same region on the grid, we enforce a constraint that nodes have no more than two children, for a total of three adjacent nodes. When there is no mapping solution, we discard the space tree and return to the GA to continue searching for a new optimal space tree configuration. Figure 5 depicts the

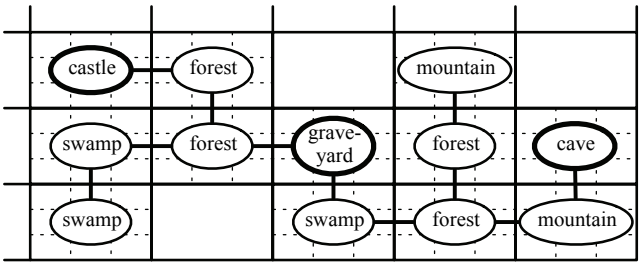


Fig. 5. A space tree mapped to a grid. Regions are solid squares and screens are dashed squares. Space tree nodes are ovals with islands marked in bold.

space tree from Figure 2 mapped to the virtual world's grid.

Once each node in the space tree has been assigned a region on the grid, the module begins graphical instantiation of the world. Each node from the space tree has an environment type, which determines what "decorations" will be placed. Decorations are graphical assets that overlay tiles and visually depict the environment type. For example, a forest environment type will be decorated with trees. The default behavior of GAME FORGE is to distribute decorations based on a Gaussian distribution centered on the center point of a region. To avoid discrete transitions between regions of different environment types, the distribution function extends beyond the boundaries of the region. The net effect is that the regions appear to blend together; decorations can appear in adjacent regions of other environment types.

Environment types sometimes require specific configurations of decorations that are not supported by a simple Gaussian distribution. For instance, a castle would ideally have semi-orderly rows of buildings surrounded by a defensive perimeter of towers. For certain environment types, we use custom distributions. Designers may provide custom distributions that are loaded in the form of bitmaps, where distinct decoration types can have different distributions determined by color intensity. This provides greater levels of authorial guidance to the designer without requiring an exact solution for any environment type to be hard-coded into the game engine. Figure 6 shows examples of the population of different types of environments. The top image shows a forest environment type, populated by a Gaussian distribution; note the blending of decorations from the adjacent swamp above this forest. The middle and bottom images show a castle environment type and the custom distribution bitmap used to generate it; the distribution of buildings is in red, towers in blue, and pavement stones in green.

The graphical instantiation is finished by drawing boundaries around the traversable regions to prevent the player character from walking off of mapped territory or in between adjacent grid positions, which are not connected in the space tree. Boundary walls are created by computing a fractal line connected to other boundary lines; non-passable tiles depicting water are placed along the fractal line.

E. Results

Figure 7 shows a world generated for a plot with three

Fig. 6. Screenshots of forest (top) and castle (middle) regions. The castle is a result of custom distributions over buildings and towers provided by a distribution bitmap (bottom).

islands: castle, graveyard, and cave. Figure 8 shows the game being played. Figure 9 shows some of the range of virtual spaces generated for the same story with two very different play styles: large, branchy worlds supporting adventuring between plot points (top), and smaller, less branchy worlds supporting a more direct progression from plot point to plot point (bottom).

Figure 10 shows the learning curves for three problem configurations expressing different player preferences. Each learning curve is the fitness of the best individual in the population, averaged over 50 runs. The blue solid line shows the average evolution of large, branchy worlds that support a lot of adventuring. The red dashed line shows the average evolution small, non-branchy worlds. The orange dotted line shows the evolution of worlds with moderate size and moderate branching. Recall that our GA minimizes penalty. The horizontal lines show the 7.5% and 5% penalty thresholds, respectively. Each configuration has different thresholds due to differences in maximum possible penalty.

After 100 generations, the moderate world population reaches lower fitness levels those for large branchy worlds or small non-branchy worlds (see Figure 10). For extremely large worlds and extremely small worlds, several features in

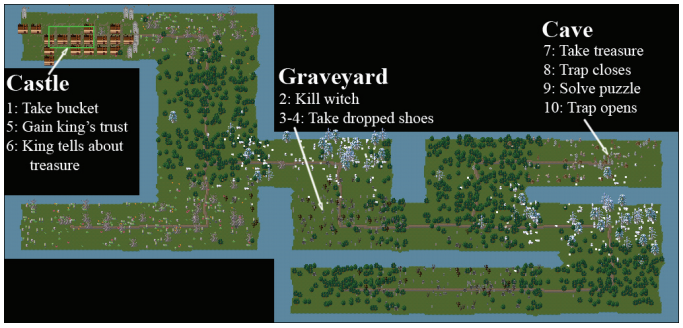


Fig. 7. A game world generated for the story from Figure 1, involving plot points that take place in a castle, graveyard, and cave. Plot points are listed.

Fig. 8. A shot of the game being played. The player is about to encounter the king in the castle island.

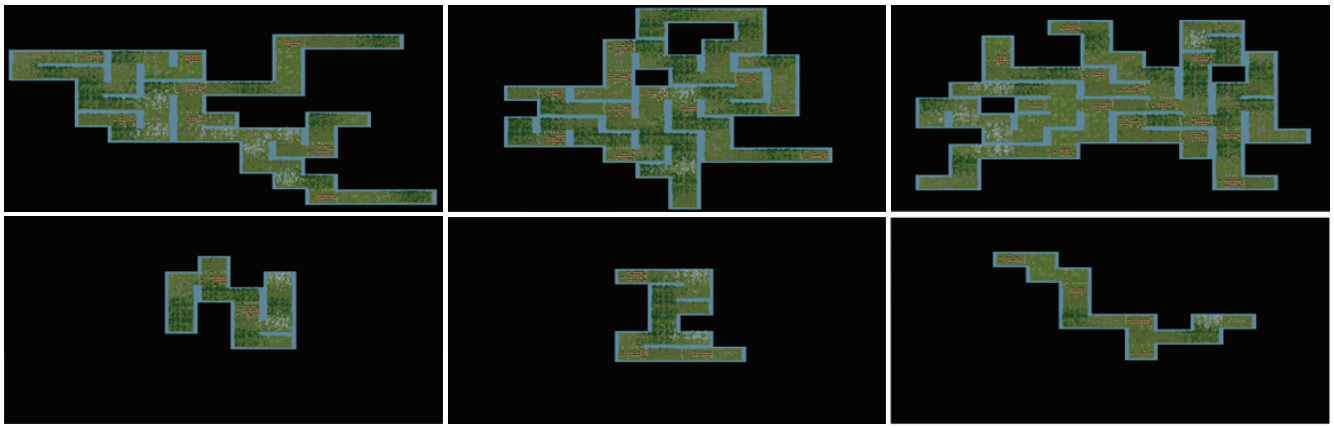


Fig. 9. Example worlds generated for the same plot. The top row was generated with parameters set for a larger world with greater branching. The bottom row was generated with parameters set for a smaller world and little branching.

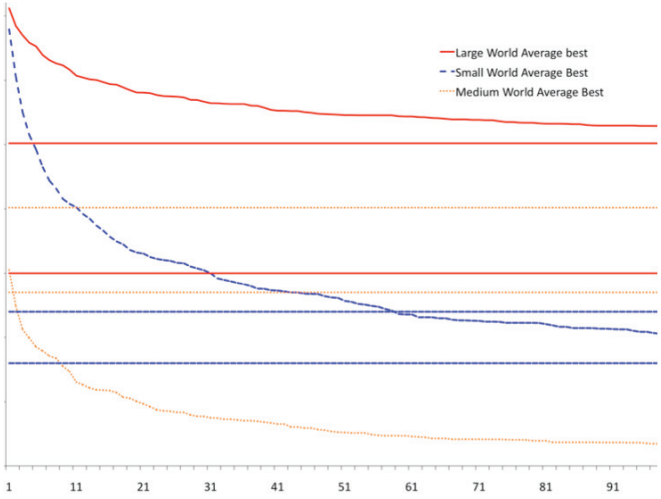


Fig. 10. Fitness of best individual per generation for three different configurations of the player model: a larger world with a lot of branching, a smaller world with little branching, and a moderate sized world with moderate branching. Horizontal lines show the 7.5% and 5% penalty thresholds for each problem configuration.

the fitness function conflict. In large, branchy worlds, we find that as the algorithm adds more nodes to expand the world, the space tree's environment transition variance penalty is likely to increase, making it difficult to minimze penalties for both. For small worlds, the algorithm stagnates when trying to create additional side paths along short bridges. Optimizing the environment transition variance for small worlds also seems to be a difficult task given the imposition of immutable environments (e.g., the islands). Future work is needed to determine the impact of high penalty values on human players perception of the world and enjoyment with different play style preferences.

V. Story Execution

After construction of the game world, the game must be playable without further modification by the user. There are two issues that must be addressed: (a) the world must be populated with NPCs, and (b) the NPCs must act out the story, which was not known prior to execution.

Population of the world with NPCs and items begins by parsing the input story structure. Character type information is used to select the appropriate avatar for each character. In our current system, avatars are simple animated sprites. We use a simple, easily extensible meta-data map from symbolic descriptors to art assets. The same process is used to handle items (e.g., the bucket, shoes, and treasure). The location at which each NPC and item appears is determined by parsing the story structure, looking for the location of the event in which the NPC or item is first referenced (always an island). Thus, the coordinate position at which to instantiate the NPC or item is determined by locating the island in the space tree and determining the center coordinates in Cartesian space with which the island node correlates. See Figure 8 for a

screen shot of the player about to interact with an NPC.

NPC and item behavior in the world is the result of executing *reactive scripts*. A reactive script is a tree of behaviors implemented in the ABL (A Behavior Language) reactive planning language [7]. While many game scripting systems utilize complicated IF-THEN-ELSE structures to manage their control flow, ABL uses hierarchically arranged behaviors defined with conditions and priorities. Each behavior encapsulates the logic to complete a specific goal under a specific world condition; alternative behaviors provide robust execution when game world conditions vary.

Story execution activates each plot point in a totally ordered sequence. Each plot point is associated with a reactive script that puppets NPCs and items, a technique similar to the *action classes* [17] approach. The reactive scripts send instructions to the game engine to modify the game environment and manipulate NPCs and items. The reactive scripts further monitor player actions and the game world state changes through sensors. When a reactive script terminates, GAME FORGE loads the next plot point and triggers the corresponding reactive script.

Plot points are parameterized to account for different narrative needs. When plot points execute, they pass their parameters on to the reactive scripts. These parameters not only determine which NPC or item performs the associated behavior, but causes different branches through the ABL behavior tree. Thus, the content and role of reactive plot scripts can vary with the nature of the type of the plot point. For instance, the "marry" script decorates a location with wedding items, adds guests in tuxes, controls the march of the bride down the aisle, and executes the dialogue of the NPC performing the ceremony. Conversely, a reactive plot script can allow for significant player interaction: the conditions in ABL behaviors coupled with sensors can implement dialogue trees, and a plot point representing a puzzle can directly change environmental conditions as the player manipulates levers and objects.

VI. CONCLUSIONS

GAME FORGE is a system that uses artificial intelligence techniques to integrate story and world in CRPG games. This process balances story requirements, designer control – in the form of insights about good arrangements of environments – and player preferences. The result is an ability to produce a game world that is both functional and favors a particular individual's play style.

Using a genetic algorithm and the metaphor of islands and bridges, GAME FORGE solves the problem of finding an optimal game world configuration that supports a given story structure that it has never seen before. The genetic algorithm attempts to balance player preferences with designer-specified information about appropriate world environment transitions. With a player model, GAME FORGE is able to incorporate individual player differences with respect to adventuring activities that occur between plot points. The designer-specified information constrains against non-aesthetic environment transitions, maximizing the believability and coherence of the game world. After the game world is generated, GAME FORGE is able to use information from the story, combined with a library of reactive plot scripts, to execute the story without further human effort.

While GAME FORGE specifically applies to Computer Role Playing Games, with a specific focus on 2D Zelda™-like games, we believe that the techniques used by GAME FORGE can extrapolate to other story-based game genres. When coupled with automated story generation systems such as that in [6], we believe that GAME FORGE demonstrates the potential for semi- or fully-automated generation and execution of highly story-dependent computer games.

REFERENCES

[1] E. Aarseth, "Beyond the frontier: quest games as postnarrative discourse," in *Narrative Across Media: The Language of Storytelling*, M.-L. Ryan, Ed. Lincoln: University of Nebraska Press, 2004.

[2] C. Ashmore and M. Nitsche, "The quest in a generated world," in *Proc. Annual Conf. of the Digital Games Research Association*, 2007.

[3] J. Dormans, "Adventures in level design: generating missions and spaces for action adventure games," in *Proc. Workshop on Procedural Content Generation in Games*, 2010.

[4] K. Hullett and M. Mateas, "Scenario generation for emergency rescue training games," in *Proc. 4th International Conf. on the Foundations of Digital Games*, 2009.

[5] H. Jenkins, "Game design as narrative architecture," in *First Person: New Media as Story, Performance, Game*, N. Wardrip-Fruin & P. Harrigan, Eds. Cambridge: MIT Press, 2004.

[6] B. Li and M.O. Riedl, "An offline planning approach to game plotline adaptation," in *Proc. 6th Annual Conf. on Artificial Intelligence for Interactive Digital Entertainment*, 2010.

[7] M. Mateas and A. Stern, "A Behavior Language: joint action and behavioral idioms," in *Life-like Characters: Tools, Affective Functions and Applications*, H. Prendinger and M. Ishizuka, Eds. Springer, 2004.

[8] M.O. Riedl and R.M. Young, "Narrative planning: balancing plot and character," *J. Artificial Intelligence Research*, vol. 39, pp. 217-268, 2010.

[9] N. Shaker, G. Yannakakis, and J. Togelius, "Towards automatic personalized content generation for platform games," in *Proc. 6th Annual Conf. on Artificial Intelligence and Interactive Digital Entertainment*, 2010.

[10] G. Smith, M. Treanor, J. Whitehead, and M. Mateas, "Rhythm-based level generation for 2D platformers," in *Proc. 4th International Conf. on Foundations of Digital Games*, 2009.

[11] N. Sorenson and P. Pasquier, "Towards a generic framework for automated video game level creation," in *Proc. of the International Conf. on Evolutionary Computing in Games*, 2010.

[12] J. Togelius, R. De Nardi, and S. Lucas, "Towards automatic personalised content creation for racing games," in *Proc. IEEE Symposium on Computational Intelligence and Games*, 2007.

[13] J. Togelius, G. Yannakakis, K. Stanley, and C. Browne, "Search-based procedural content generation," in *Proc. 2nd European Event on Bio-inspired Algorithms in Games*, 2010.

[14] T. Tutenel, R.M. Smelik, R. Bidarra, and K.J. de Kraker, "Using semantics to improve the design of game worlds," in *Proc. 5th Annual Conf. on Artificial Intelligence for Interactive Digital Entertainment*, 2009.

[15] D. Weld, "An introduction to least commitment planning," *AI Magazine*, vol. 15, pp. 27-61, 1994.

[16] G. Yannakakis and J. Togelius, "Experience-driven procedural content generation," *IEEE Trans. on Affective Computing*, to be published.

[17] R.M. Young, M.O. Riedl, M. Branly, A. Jhala, R.J. Martin, abd C.J. Saretto, "An architecture for integrating plan-based behavior generation with interactive game environments," *J. Game Development*, vol. 1, pp. 51-70, 2004.

International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 08 | Aug 2020 www.irjet.net p-ISSN: 2395-0072

Using 3D Models from Blender for use in OpenGL Virtual Reality Applications

Aditya[1], Anitha M[2]

[1]*Student, Dept. of Computer Science, Dayananda Sagar College of Engineering, Bangalore, India*
[2]*Professor, Dept. of Computer Science, Dayananda Sagar College of Engineering, Bangalore, India*

Abstract - *OpenGL is used in various graphical software implementations that vary from games to medical imaging and virtual reality applications. The use of 3D modelling software such as Blender has increased over time for games, simulations and animations. Such software allows graphical models to be developed with ease and to be visualized immediately rather than at run time. But they lack in terms of platform compatibility and have significant overheads. In this paper we suggest a hybrid approach using Blender for modelling and OpenGL to perform the rendering, animation and transformations of the models is outlined and contrasted against a pure OpenGL workflow. Better workflow efficiency, platform compatibility and low overhead aspects are reaped from this hybrid approach when compared to pure OpenGL.*

Key Words: Graphical Modelling, 3D Animation, Blender, OpenGL, Assimp, Virtual Reality

1. INTRODUCTION

Virtual reality is an artificial three-dimensional environment created by a computer and presented to a person in an interactive way [1]. It refers to the computer simulation displaying an environment through which one can walk and interact with objects and simulated computer-generated people. With the increasing availability of cost accessible hardware components, virtual reality applications are finding its way into various applications such as gaming, learning and social skills training, military training, architectural design, simulations of surgical procedures, rehabilitation of psychological disorders such as anxiety, schizophrenia, depression, and eating disorders. [2,3,4,5,6,7].

The artificial environment is created by 3D modelling which creates a realistic world so that the users could immerse into this environment by interacting with these objects using the hardware devices such as computer mouse, joystick, force sensor, cyberglove having the sense of the real world. The efficiency of VR applications solely depends on the creation of 3d objects. In the period between 1996 and 2005 key changes emerged on the platforms such as Superscape World Builder and OpenGL that supported easier development and distribution of desktop VR applications [8]. OpenGL provides the better workflow efficiency, platform compatibility and low overhead that is needed for the VR applications. Hence, in this paper we suggest the usage of improvised technique of creating 3D objects with the combination of OpenGL and Blender software.

2. MODELLING FOR GRAPHICS APPLICATIONS

OpenGL is a cross-platform, cross-language software interface to graphics hardware for rendering 2D and 3D vector graphics [9]. The interface consists of several hundred procedures and functions that allow a programmer to specify the objects and operations involved in producing high-quality vector graphical images. Blender is a free and open source 3D modelling, simulation and animation suite [10,11]. It supports the entirety of the 3D pipeline including rigging, rendering, compositing and motion tracking, video editing and game creation. A Python API is available for use by advanced users.

Modelling for graphics applications can be performed using one of three techniques:

1. OpenGL only: OpenGL allows models to be created by specifying triangles or quads as arrays of vertices, edges & normals, with additional arrays for color and texture data. Another technique to create models in OpenGL is enabled with the use of toolkits such as freeglut which provide routines to create basic structures such as cubes, cones, cylinders and teapots. This approach is powerful but basic and requires knowledge of the coordinates, normals and colors beforehand, leaving little room for experimentation and creativity. Complex models such as a human face are challenging to work with in pure OpenGL.
2. Blender only: Blender supports game and animation development through its GUI interface. This seems to be the best option, but misses out on the widespread adoption and testing that OpenGL has received. OpenGL has a bare metal approach to graphical programming which may be useful in some cases. Complex models are easily tweaked and flexibility exists in the design of models. Models can be viewed as they are being created. Blender also provides basic structures such as cubes, spheres etc. [26]
3. Creating the models in Blender and importing the data into OpenGL applications: In this approach modelling software such as Blender is used to create the models, and a translation program or library such as Assimp is used to convert the models to corresponding OpenGL code. The data format generated by the modelling software is usually

defined by a set of polygons with position and normal data. This allows for easy modification of models if needed, and incorporates the cross - platform, cross - language abilities of OpenGL.

Modelling is used by several industries with many more such as medical simulations, and virtual surgery as described in [12,13]. Dental processes can be better understood by reconstructing a person's jaw in software[14,15]. Engineering models are now being modelled in 3D graphical environments that enable printing of prototypes and automated CAD generation. Industrial workflow simulations are better illustrated using animations rather than by a verbal description [16]. Human bio-mechanical simulations help understand the internal working of skeletal muscles better as described in [17].

3. MODELLING WITH PURE OPENGL AND FREEGLUT

OpenGL provides functions to generate primitives such as points, lines and polygons with additional functions for view manipulation, texture projection, object transformations, rendering and so on. This along with the cross platform compatibility and open source licensing makes it the de facto choice for development of graphics applications.

OpenGL being an API, does not provide advanced functions for complicated 3D modeling. So creating complicated 3D models has to be performed through the combination of the basic geometry such as points, lines and polygons (Fig 1). OpenGL provides a function (glDrawArrays)[9,18,19] to render a model using arrays of vertex coordinates, edges & edge order, normals, and vertex color.

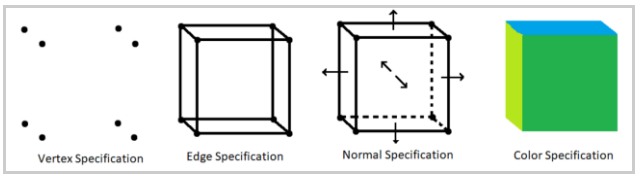


Fig -1: Modelling of a 3D object in OpenGL using arrays [20]

A) Basic Geometry Modelling: Simple models can be plotted by hand and the coordinates provided as input to the OpenGL pipeline. This works well for models that are simple and provides great control over the individual vertices and normals. While rotation, scaling or translation can be handled using functions in OpenGL, transformations that affect sections of a model (facial expressions) are not simple. As the polygon count in a model increases, simple modelling is not feasible.

B) Modelling using OpenGL and freeglut: Utility toolkits like freeglut (A replacement for the original GLUT toolkit) provide abstractions to create simple 3D structures such as boxes, cylinders, cones, spheres, discs, teapots and more[20]. While these toolkits simplify basic 3D objects that could be the building blocks of a model, they restrict the ability to modify individual faces, normals, vertices or colors of faces. Models of realistic looking objects, curved objects such as pillows are not fit for generation through this method, and hence developers rely on glDrawArrays instead.

Using the freeglut toolkit, modelling of a basic snowman structure took about 2 minutes with boilerplate code and 6 lines of modelling code as outlined below. Boilerplate code was required to visually verify the appearance of the model and the positioning of structures [21]. The boilerplate code contained the initializations for OpenGL and freeglut as well as a mouse input handler to allow for a 3D walk around of the model for verification of the model's accuracy to the design.

```
glScalef(1.0,4.0,1.0);
glColor3i(255,255,255);
glutSolidSphere(1.0);
glTranslatef(0.0,5.0,0.0);
glScalef(1.0,0.25,1.0);
glutSolidSphere(0.5);
```

4. MODELLING WITH BLENDER

Blender is an open source modelling and animation suite, which makes it a good candidate for most graphics application development endeavors. Contrary to modelling with OpenGL, it provides real time visualization of the created model and transformations applied to it with an interactive GUI. This makes designing easier and training new designers simpler.

Similar to OpenGL toolkits, Blender provides some basic structures to construct models, similar to OpenGL that can be manipulated to obtain any shape desired [25]. For example, a thin cylinder can be manipulated into a plate. The model/scene generated is saved in BLEND format by default, but can be exported in many other formats including OBJ, DAE, ABC, FBX, X3D. Additionally, Blender allows for easier generation of curved models and high polygon count models through the use of GUI based click and drag tools rather than requiring understanding of various mathematical formulae. Specifying and changing colors, textures are convenient and simple to perform.

Using Blender alone, modelling a simple white snowman structure took about a minute and was achieved using the UV sphere mesh, scale and move options. Then a material with color set to white was applied to both meshes at the same time. The resulting model could be visually verified without requiring any boiler plate code.

5. HYBRID APPROACH WITH BLENDER AND OPENGL

Translation of models created in Blender can be performed in a few ways, each with its own characteristics. A suitable approach has to be chosen based on the requirement of the application. Translator programs that work on a single file format can be simpler to implement if the additional features

of import libraries are not used in the application, while import libraries like Assimp can convert multiple file formats comprehensively to a single data structure format to be translated.

A) Writing a translator program: Blender allows output files to be in several formats. Since OBJ files have a simple file structure, it is a good candidate to use for writing a translator program. Each line in the file represents an entity. Vertices are represented by a 'v' at the start and faces are represented by a 'f' [18,22]. A triangle in OBJ file representation consists of three vertices and a face as enlisted below.

```
v 0.0 0.0 0.0
v 0.0 1.0 0.0
v 1.0 0.0 0.0
f 1 2 3
```

A simple translator program would be (pseudocode):

```
for line in file:
 lineElements = line.split(" ")
 if lineElements[0]=='v':
  queue.push(lineElements[1:])
 elif lineElements[0]=='f':
  glBegin(GL_POLYGON)
  while !queue.isEmpty():
   glVertex3f(queue.pop())
  glEnd()
```

B) Using a import/translation library such as Assimp (Open Asset Import Library):

Writing a translator program directly requires detailed knowledge about the data structure and internal format used by a particular file. This also restricts further development of the graphics application to use that file format. Additional limits are imposed by the efficiency, capability and portability of the translation program on different platforms. Libraries such as Assimp allows a programmer to load model files of various formats dynamically with the added benefit of being well tested and robust. Assimp accepts most of the file formats exported by Blender including BLEND format. It then processes the file into a data structure; a simplified representation of which is Fig 2. The structure of each component of the model is a node in Assimp's tree-like data structure and hence can be recursively traversed. Each node consists of an index reference to mMeshes. Each mesh structure contains vertex, normal, texture, face, and material information either as a reference, or as arrays. A scene is a structure generated by Assimp when a file is imported using the aiImportFile function. aiImportFile preprocesses the model/scene by triangulating faces, correcting for winding order, removing the redundant and hidden polygons and so on [23].

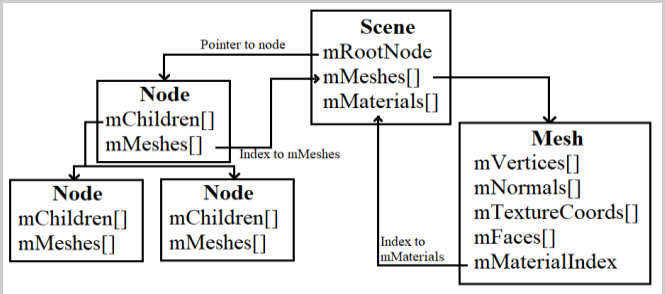


Fig -2: Data structure generated by Assimp after file processing [24]

The following pseudocode is a simple implementation of using Assimp to process Blender generated model files and render the corresponding models in OpenGL.

```
def recursive_generate(Assimp Scene sc, Assimp Node node):
 for each mesh in node:
   Load material from mMaterials and corresponding mesh
   from mMeshes[]
   Generate OpenGL object using GL_POINTS, GL_LINES,
   GL_POLYGON or GL_TRIANGLES.
   Generate substructures of current node recursively by
   calling Recursive_generate on mChildren[]
   Create an OpenGL list with the tree of objects and return
def display():
 For all the lists, apply transformations and call
 glCallList(list) to render onto the screen
```

The tree when recursively traversed and processed results in a list of OpenGL instructions which can be stored in a list for reduced processing needed later. The list can be called using glCallList whenever needed to create the desired model. This makes it easier to decouple model design and software logic development for applications such as game development and AR/VR application development.

6. COMPARISON BETWEEN THE APPROACHES

OpenGL with toolkits like freeglut allow for a wider range of models to be created by using predefined 3D objects. This is however not always useful, and in some cases may increase the number of faces to be processed to render a model. For example, a lamp would be at the least made up of a disc, a cylinder, and a frustum of a cone. Parts of each of those objects would be clipped, and processing power used to render additional regions of the lamp would be wasted. Objects like human models would be complicated to generate without any additional software, and with additional software to generate vertex and face data, any changes made to the model's design would require rework to update the stored vertex and face data to match the color & texture data. Blender and other similar modelling suites overcome this inconvenience and allow for the design of models and program logic to be decoupled. Vertex, color, texture and normal data are stored with the models, and do not require manual updating in OpenGL when a part of the model is

changed. Import libraries convert several file formats to a single data structure that can be used with a custom translation program to render the models in OpenGL. Since Blender has an interactive GUI, creation and design of complex models such as humans is easier with continuous feedback on the changes made to refine the model. This reduces time spent on waiting for the test driver program to compile for verifying that a refinement has the desired effect. Performance does have an impact, but this can be mitigated by importing the model before the program execution starts, importing on a separate thread, or embedding the imported data directly in a production build. Moreover, importing and processing of models is a onetime task, and is not repeated on each frame as with clipping or culling of faces on inefficient model designs.

Fig -3: Model translated and rendered in OpenGL

Fig -4: Model rendered in Blender

Fig. 3 is a screen capture of the OpenGL render of a model created in Blender as saved as a BLEND file. The file was converted into OpenGL code using Assimp and a translator program similar to the one described above. Fig 4 is the same model rendered using Blender's internal render engine.

The render generated through importing the model into OpenGL took an additional second at the beginning of the application to load, process and translate the BLEND file into OpenGL code. This is a single occurrence task and can be hidden behind a loading screen or performed ahead of time and cached. The render generated through Blender's internal renderer has a similar appearance to the one from OpenGL translation. The observed difference is the variance in specular reflection in Fig. 3. This can be attributed to the translation program not processing the stored specular values in the BLEND file. Another difference is that the Blender render processes ambient occlusion while the OpenGL render was not programmed to do so. No other significant differences could be found.

7. CONCLUSION

For the development of graphics applications with OpenGL, the graphics modelling tools available are limited and lack user friendly techniques. A hybrid approach to application development with OpenGL wherein a professional 3D modelling software such as Blender is better suited to modern workflows. The models can be imported into the application using an import library like Assimp. The imported data is translated into OpenGL commands using a translation program. Once translated, the OpenGL commands can be run multiple times without requiring any loading or processing of the model files. This technique allows for faster development of models, and makes it easier to change the models as required. The result concludes that models can be easily and accurately imported into OpenGL when precautions are taken to correct for any parameters that differ between the modelling software and OpenGL.

ACKNOWLEDGEMENT

This work was undertaken in the Department of Computer Science, Dayananda Sagar College of Engineering, Bengaluru, India under the guidance of Prof. Anitha M of Department of Computer Science, Dayananda Sagar College of Engineering.

REFERENCES

[1] Kamińska, D.; Sapiński, T.; Wiak, S.; Tikk, T.; Haamer, R.; Avots, E.; Helmi, A.; Ozcinar, C.; Anbarjafari, G. Virtual Reality and Its Applications in Education: Survey. Information 2019, 10, 318.

[2] Schmidt, M., Beck, D., Glaser, N., and Schmidt, C. (2017). "A prototype immersive, multi-user 3D virtual learning environment for individuals with autism to learn social and life skills: a virtuoso DBR update," in International Conference on Immersive Learning, Cham: Springer, 185–188. doi: 10.1007/978-3-319-60633-0_15

[3] Gallagher, A. G., Ritter, E. M., Champion, H., Higgins, G., Fried, M. P., Moses, G., et al. (2005). Virtual reality simulation for the operating room: proficiency-based training as a paradigm shift in surgical skills training. Ann. Surg. 241:364. doi: 10.1097/01.sla.0000151982.85062.80

[4] Song, H., Chen, F., Peng, Q., Zhang, J., and Gu, P. (2017). Improvement of user experience using virtual reality in open-architecture product design. Proc. Inst. Mech. Eng. B J. Eng. Manufacturing 232.

[5] Alexander, T., Westhoven, M., and Conradi, J. (2017). "Virtual environments for competency-oriented education and training," in Advances in Human Factors, Business Management, Training and Education, (Berlin:

Springer International Publishing), 23–29. doi: 10.1007/978-3-319-42070-7_3

[6] Neri, S. G., Cardoso, J. R., Cruz, L., Lima, R. M., de Oliveira, R. J., Iversen, M. D., et al. (2017). Do virtual reality games improve mobility skills and balance measurements in community-dwelling older adults? Systematic review and meta-analysis. Clin. Rehabil. 31, 1292–1304. doi: 10.1177/0269215517694677

[7] Englund, C., Olofsson, A. D., and Price, L. (2017). Teaching with technology in higher education: understanding conceptual change and development in practice. High. Educ. Res. Dev. 36, 73–87. doi: 10.1080/07294360.2016.1171300

[8] Keshner EA, Weiss PT, Geifman D, Raban D. Tracking the evolution of virtual reality applications to rehabilitation as a field of study. Journal of Neuroengineering and Rehabilitation. 2019 Jun;16(1):76. DOI: 10.1186/s12984-019-0552-6.

[9] Khronos Group, OpenGL 4.0 Specification (2010), https://www.khronos.org/registry/OpenGL/specs/gl/glspec40.core.pdf

[10] Blender Foundation, Blender.org, https://www.blender.org/about/

[11] Ben Crowder, "Blender" Linux Journal, Volume 1999 Issue 60es, 1999

[12] Zhou, Z.-H & Wen, X.-J. (2018). 3D Reconstruction of medical image based on opengl. Journal of Computers (Taiwan). 29. 249-260. 10.3966/199115992018042902024.

[13] Timothy Ellis, “Animating to improve learning: a model for studying multimedia effectiveness,”31st ASEE/IEEE Frontiers in Education Conference, 10 - 13, 2001 Reno, NV, 2001.

[14] Dovramadjiev, Tihomir. (2016). Advanced creating of 3D dental models in Blender software. Scientific-technical union of mechanical engineering Bulgaria. IV. 32-33.

[15] Cankova, Kremena & Dovramadjiev, Tihomir & Jecheva, Ginka. (2017). Computer parametric designing in Blender software for creating 3D paper models. ANNUAL JOURNAL OF TECHNICAL UNIVERSITY OF VARNA. Vol.1 Issue 1. 1. 77 - 84. 10.29114/ajtuv.vol1.iss1.44.

[16] Ma, Yan & Dong, Tianping & Lan, Xiaohong & Liu, Lunpeng & He, Guotian. (2012). Research of Industrial Robot Simulation based on OpenGL. International Journal of Advancements in Computing Technology. 4. 248-255. 10.4156/ijact.vol4.issue19.30.

[17] Zeng, Yanhong, Jianlong Fu and Hongyang Chao. “3D Human Body Reshaping with Anthropometric Modeling.” ICIMCS (2017).

[18] Paul Bourke, Object Files (.obj), http://paulbourke.net/dataformats/obj/

[19] Mason Woo, Jackie Neider, Tom Davis, and Dave Shreiner. 1999. OpenGL Programming Guide: The Official Guide to Learning Opengl, Version 1.2 (3rd ed.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[20] Angel, Edward; Shreiner, Dave. (2009). Interactive Computer Graphics., 6th Edition Book

[21] Angel, E., & Shreiner, D. (2011). Introduction to modern openGL programming. SIGGRAPH '11.

[22] Wikipedia contributors. (2020, August 10). Wavefront .obj file. In Wikipedia, The Free Encyclopedia. Retrieved, August 20, 2020, from https://en.wikipedia.org/w/index.php?title=Wavefront_.obj_file&oldid=972078922

[23] Assimp process.h documentation, http://assimp.sourceforge.net/lib_html/postprocess_8h.html

[24] Joey de Vries, Assimp, https://learnopengl.com/Model-Loading/Assimp

[25] Takala, Tuukka & Mäkäräinen, Meeri & Hamalainen, Perttu. (2013). Immersive 3D modeling with Blender and off-the-shelf hardware. IEEE Symposium on 3D User Interface 2013, 3DUI 2013 - Proceedings. 191-192. 10.1109/3DUI.2013.6550243.

[26] T. Dai, Z. Wang and S. Xu, "Research of Creating and Fetching 3D Models of Virtual Reality Based on OpenGL," 2006 International Conference on Mechatronics and Automation, Luoyang, Henan, 2006, pp. 1991-1995.doi: 10.1109/ICMA.2006.257560

A game design plot: exploring the educational potentials of history-based video games

Baradaran Rahimi F., Kim B., Levy R.M., and Boyd J.E.

Abstract—The number of video games that are developed based on real historical events and evidence is increasing. These history-based video games provide players learning opportunities, but a certain type of such games – first and third person shooters - has not been carefully examined for their potentials. Knowing what players say about their game experience - even if the information and knowledge are inaccurate - helps researchers understand what type of learning could happen with such games. In this paper, we propose a systematic approach to assessing games as learning environments, using the method of comparing authenticity of popular history-based video games. Through a qualitative data analysis, we studied players' comments on the web-based communication services, such as game forums, digital distribution platforms, and discussion websites. Casual players' conversations on these websites showed that there exist several learning potentials in the games for players including building their understanding about history and historical forces of the time, through personally relating to the specific events, social artifacts, and places.

Index Terms— Communications technology, E-learning, History-based video games, Web-based communication services

1 Introduction

There exist controversies about historical video games, especially first and third-person shooter games, if they have potential to support learning. For example, according to McCall (2016, p.526), "the presence of choice and thus counter factual outcomes raise some objections to historical games and their portrayal of the past." Some, however, assert that the video games of historical contexts might provide a useful and motivating method of learning about history and social history (Connolly et al., 2012; and McCall 2018). Social history concentrates upon the social, economic, and cultural institutions of a people and "emphasizes the forces that drive human behavior" (McCall, 2012, p. 20). While historical events such as revolutions and wars form a considerable part of the history, architecture and urban life of people are significant parts of the social history. History-based video games can provide players with a representation of social history. Yet, the educational values can get lost in the entertainment aspects of such games. Consequently, many historical details and values of these games remain as a background for players, if noticed at all.

We recognize the richness of the commercial games regardless of their success. While we point to the relative shortage of adopting their learning potentials, we also acknowledge the danger of reducing the multi-facetted human history to the history of militarism and violence. There exist methods of user-testing to understand how players interact with new games (Desurvire & El-Nasr, 2013). However, there are not enough systematic ways to maximize different learning values of commercial history-based video games, especially first and third-person shooters, for casual players. These games provide historical environments rendered in 3D and navigated by a player agent. Yet, they include some violent interactions regarding their first and third-person shooter nature. Presumably these games can provide the players with historical sights, sounds, and spaces as well as animated characters capable of talking and interacting within the historical environment of the game. These games provide a different experience from the history-based strategy games, such as Civilization, which Chapman, Foka, and Westin (2017) categorizes as conceptual-approach games. Such history games as Civilization have been studied and proven to be useful for learning purposes (e.g. Squire & Barnett, 2004). However, most of these games are played from the "bird's eye view" as an invisible ruler, who does not experience the social history. In contrast and as a result of their view, first and third person-shooter games have the potential to provide players with sense of presence in the historical events of the game.

Conversations of casual players on web-based communication services such as game forums, digital distribution platforms, and discussion websites reveal the learning potentials in commercial video games (McCall 2018). Yet, each game's depiction of the history is value-laden, and there needs a systematic way to understand the values conveyed by these commercial games. How can the learning potentials for the players – to inquire into history in relation to the historical authenticity of different video games – be studied and compared? The goal of this paper is to investigate the learning opportunities that players may have with commercial history-based video games (i.e., historically oriented, situated, or themed video games). Based

- *Baradaran Rahimi, F. is with the Computational Media Design (CMD) program, University of Calgary, Calgary, AB, T2N 1N4, Canada. E-mail: farzan.baradaran@ucalgary.ca*
- *Kim. B. is with the Werklund School of Education, University of Calgary, Calgary, AB, T2N 1N4, Canada. E-mail: beaumie.kim@ucalgary.ca*
- *Levy, R.M. is with the Faculty of Environmental Design, University of Calgary, Calgary, AB, T2N 1N4, Canada E-mail: rmlevy@ucalgary.ca*
- *Boyd, J.E. is with the Department of Computer Science, University of Calgary, Calgary, AB, T2N 1N4, Canada E-mail: jboyd@ucalgary.ca*

xxxx-xxxx/0x/$xx.00 © 200x IEEE Published by the IEEE Computer Society

on this goal, we focused on assessing learning opportunities that players may have with commercial history-based video games instead of assessing games that are designed for learning about history or purely for entertainment. In this paper, we explored what kinds of conversations can happen around the historical contexts of video games and investigated how these conversations are already taking place online. Based on our findings we provide a game design plot for visualizing, listing, and comparing the learning outcomes of history-based video games. The plot and the systematic approach that we propose can help developers, designers, researchers, and educators make informed decisions about application of video games in different situations.

2 LITERATURE REVIEW

With the growing interest in video games over the past few decades, there has been a growing concern about the impact of gameplay on users. Some found negative impacts such as the increase in aggression and arousal (Anderson, 2004), difficulties in regulating the amount of time spent on playing games (Ogletree & Drake, 2007), and addiction (Griffiths & Davies, 2002). Yet, others reported on optimistic visions of video games, such as raising awareness of important social issues (Schreiner, 2008), developing gaming literacy in real life situations (Zimmerman, 2009), and learning through designing games in a participatory culture (Baradaran Rahimi, & Kim, 2018). Despite the skepticism, many scholars believe that video games provide useful and novel ways of learning. Modern theories of learning suggest that learning is most effective when it is active, experiential, situated, problem-based, and provides immediate feedback. Games offer activities that have these features and often involve players' emotions that help them make meaningful links to their prior knowledge and experience (Connolly et al., 2012; and Schreiner, 2008).

Metzger and Paxton (2016) argued, "[h]istory has much to offer video game developers, including ready-made settings that can activate players' prior mental schemas to provide a sensation of verisimilitude within a context that is nevertheless alternate and malleable" (p.533). Games can be seen as forms of public history. McCall (2018) appreciates how historical games can model some aspects of past systems and have the potential to be very powerful media for encouraging thoughts about history. In playing historical games, especially first and third person shooters, players are immersed into the historical settings which can be very powerful for understanding history. However, historical games usually create an experience that does not accurately match all the details in documented history. Therefore, authenticity and accuracy in the historical games require a reconsideration of terms (McCall, 2018).

Supporting that history-based video games can help understanding history (McCall, 2018; Metzger and Paxton, 2016), we believe that defining factors of historical context in games would facilitate the productive use of them. We take the Tschumi (2012), Gerosa (2005), and Pallasma's (2001) views on discussions relevant to factors of historical context as the environment and architecture are important parts of the action in the video games. Later, these provided the grounds for identifying the "learning potentials" in Table 2. To explore the educational potential of such video games, we looked at the most relevant literature in game studies to identify three groups of the history-based video games in terms of authenticity and accuracy. Using the literature in media and architecture five factors are identified to describe the historical context in such games. A plot was then created based on these factors for those three groups of video games (Fig. 1).

2.1 Learning through history-based video games

The ways in which players learn and think about history and social history can be influenced by popular culture and media technologies (Wineburg, 2001; and Schreiner, 2008). Video games, that are historically oriented, situated, or themed (Metzger and Paxton, 2016), offer players an opportunity to alter historical events, such as winning a battle based on their choices. Experiencing these alternatives can also help understanding how a single decision has the potential to change the history or result in better solutions in the context of history:

> Video games can provide a unique mode of engagement for thinking about the world and its past—allowing the young learner not just to observe historical accounts but to actively engage in simulations of history, easily repeating events or sequences, or even controlling and modifying alternate visions of what could have happened in the past (Metzger and Paxton, 2016, p.533).

As Walsh (2013) noted, some game developers tend to move their games from the realm of pure fantasy to the realm of reality in the minds of players. Gee (2008, p.254), recognized such games as "action- and goal-directed simulations of embodied experience." Video games offer an opportunity for learning history in such an immersive, interactive, and engaging way that is almost impossible using traditional media like movies and books.

Metzger and Paxton (2016) suggested a framework that can help identify the video games that facilitate learning history. Their framework offers "deployments"[1] of historical understanding while considering authenticity versus fantasy in video games. Metzger and Paxton (2016) suggested that disciplinary practices, elements, and perceptions of validity are important factors to make a video game a legitimate historical representation. Behind the group of legitimate games lays considerable archival or scholarly research by historians and professionals. This archival and scholarly research can range from a deep investigation of a historical event, atmosphere, fashion, and manner of socialization to a detailed recreation of historic buildings, landscapes, sites, and cities (Metzger and Paxton, 2016). The second group of video games valorizes historic momentums and important events, but not necessarily based on extensive research (Metzger and Paxton,

[1] Deployment is the term that Metzger and Paxton (2016, 533) used for "a designed representation of the past utilized by game creators to reflect a perspective, interest, or purpose."

2016). A third group focuses on reconstructing objects, artifacts, or communications with authentic details, such as historical buildings, furniture, battlefields, and weaponry in a certain period (Metzger and Paxton, 2016). There exist many deployments of history in video games, but the history often collides with fantasy. The more a game goes into fantasy, the more it may depart from the authenticity.

2.2 Factors of historical context in games

In conceptualizing varying adoptions of historical contexts in video games, we considered how architecture and cinema have coexisted in games from the early days of the game industry. DOOM, one of the most significant and influential titles in video game history, pioneered the use of architecture and cinema in a first-person shooter game (Arsenault, 2009). Later, the cinematic exploration of video games was pursued by players on Machinima[1], as creating game movies became "an outlet for creative expression" (Lowood, 2006, p.326). Such movies can be considered as a "part of the player culture to engage with more challenging stories and themes" (Lowood, 2006, p.380). The architectural and cinematic representations engage players as protagonists of video games:

> A central concept for the architecture of video games is that they are part of the action. They live together with the characters. They are often theatrical architecture, with traps, moving walls, pitfalls, etc. They respond to every move of the player: in that way, they are living, organic creatures. But they also live because they are visited, admired and interacted with by people (Gerosa, 2008, p.53).

Bernard Tschumi (2012, p.176) argued that "there is no architecture without event, no architecture without action, without activities." Citing Foucault, Tschumi (2012) said that an event is "the moment of erosion, collapse, questioning, or problematizing the very assumption of the setting within which a drama may take place – occasioning the chance or possibility of another, different setting" (p.176). History-based games unfold in a spatial context, such as an ancient city in the Roman Empire (e.g., Ryse: Son of Rome), Paris during the French Revolution (e.g., Assassin's Creed Unity), or Los Angeles in the 1940s (e.g., La Noire). Players engage in activities to complete a mission (e.g., investigating a crime scene, destroying the enemies' structures, or helping a certain group). Events in a game can occur through unexpected encounters in the games (e.g., a player meeting a street hawker selling something from that period) within the social-historical contexts by roaming the open-world of the game. This can resonate with Foucault's explanation; players may begin questioning or problematizing the very assumption of the game (i.e., gaming as accomplishing pre-determined missions) and find their own missions (e.g., free roaming in the game world to explore its architecture).

Jean Nouvel, similarly, said that "[a]rchitecture exists, like cinema, in the dimension of time and movement. One conceives and reads a building in terms of sequences[2]." As Nouvel explained, movement (activity) of people help them conceive building (space) through time in sequences. Players experience time in different scales within history-based games. In addition to the real-time that players spend with the video games, players engage with the time that unfolds in the virtual world of the game. In this time scale, a couple of days or weeks or months in the virtual world of the game may pass just in half an hour. The players also engage with the historical time that the game narrative belongs to. It is indicative of its culture and customs. People, as historical societies or individuals are inseparable parts of video games considering their roles in historical events.

We consider people, time, space, event, and activity as five factors that represent life in games. These factors can guide game creators to provide players with educational experiences within the historical contexts of the games. These factors are critical in identifying or designing structural attributes of the history-based video games. Thus, these factors can play an important role in transforming a history-based video game into an educational medium. Based on Squire's study (2011), considerable historical content of such games helps players make decisions leading to alternative ideas of the historical consequences.

2.3 History-based video games plot

To visualize the information from the literature, we created a plot in which a continuum of historical authenticity and five factors of historical context represent two axes. The plot can be used as a basis for defining different factors in relation to the historical authenticity.

The horizontal axis does not represent fixed ends of the authenticity continuum. Instead, it represents both the subjective and relative positions of more authentic and less authentic factors of a video game. For each game, the higher the frequency of appearing a factor in players' comments (subjective) the further the factor goes towards right side of the authenticity continuum (relative) in Fig 2. The vertical axis with the factors can play a critical role in identifying or designing structural attributes of the history-based video games. Depending on the nature of a game, specific variables can be used to describe the response to specific occasions in the game. A plot of authenticity against these factors can be used to compare multiple video games to understand the educational values they offer. This plot will be used in this paper for comparing sample video games and their educational values.

3 Method

To understand the learning potentials of commercial video games and provide a tool for others to examine similar games, authors selected five history-based video games for discussion in this paper. Commercial games are still more affordable, already available, and far less obtrusive than sophisticated serious games for a variety of learning purposes. Five selected games are serving as examples of a very specific type of commercial games, namely, first

[1] A global multimedia network focusing on animations made from games.

[2] http://www.pritzkerprize.com/2008/bio

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TG.2019.2954880, IEEE Transactions on Games

and third-person shooter historical games that are not studied before.

A great number of games can be purchased and downloaded digitally through Steam. Steam is one of the main vendors distribute digital-download games (McCall, 2016). Our search for the games with "historical" tag on Steam[3] returned 327 game titles. We selected games for this paper based on the historical authenticity, type (first and third-person shooter), and popularity of the video games. Selection is not across the breadth of historical game types. We only selected examples of what Chapman, Foka, and Westin (2017) calls the realistic approach to sims. Although the experiences of the players of these games are mostly beyond our reach, it is possible to tap into them by investigating their online communities:

> The multitude of Internet forums, however, where players can and do share their thoughts and opinions about games with their peers, offer an untapped resource for investigating how players can interact with the history embedded in games…Accordingly, a qualitative study of several forum threads can illustrate some of the ways players interact with historical games (McCall 2018, p.410).

To investigate how game players, engage in online conversations about historical aspects of the games, we analyzed the comments and discussions of players retrieved from online communities and venues including Steam, NeoGAF, YouTube, and GameFAQs. Investigating opinions of actual players through forum posts is a very new methodology in game studies (McCall, 2018). According to McCall (2018, p.405), "in the forum dialogues spurred by the game, players are empowered to interact with the past and, as they do so, analyze and critique the game designers' visions of the past". Table 1 summarizes the themes, publishers, and genres of the selected games for this study.

TABLE 1
SAMPLES OF COMMERCIAL HISTORY-BASED VIDEO GAMES

TITLE (RELEASE YEAR)	DESCRIPTION
ASSASSIN'S CREED UNITY (2014)	Action-adventure game set in Paris during the French Revolution. The player must fight for peace and free will, against the enemies to expose the true powers behind the Revolution. (Ubisoft Montreal)
LA NOIRE (2011)	Neo-noir detective action-adventure video game set in Los Angeles during the 1940s. The player assumes the role of a detective investigating several real LAPD cases based on a specific type of crime. (Rockstar Games)
THE SABOTEUR (2009)	Neo-noir open-world third-person action-adventure game set during World War II (WWII) in German-occupied France. The player is a rival who joins the French Resistance to take revenge of his best friend killed by a Nazi commander. (Electronic Arts)
CALL OF DUTY: WORLD AT WAR (2008)	First-person shooter game set in the period of WWII. It is open-ended, giving the player multiple ways to complete several missions and objectives. (Activision)
BROTHERS IN ARMS: ROAD TO HILL 30 (2005)	First-person shooter game set around the true story of the 502nd Parachute Infantry Regiment of the famed 101st Airborne Division who were dropped behind German lines on D-Day. It is based on the historical Mission Albany, in which the player must complete tasks based on real actions of the 101st in Normandy. (Ubisoft)

Descriptions are based on data from http://store.steampowered.com/.

The games we chose may have more potential to provide holistic experience for educational purposes rather than the latest games that have online services. In recent years, the video game industry has been taking more and more exploitive, monetized design approach. For example, Call of Duty: WWII has paid loot-box drops, which makes the game more individualized and random. The randomized elements that players pay and receive affect the historical details that players are exposed to.

We studied 952 players' comments and public discussions made by the players of these five sample games to identify themes and keywords based on the historical authenticity and factors of each game. We applied open-coding and qualitative data analysis to these textual materials using NVivo 11 software (Watson, Mong, & Harris, 2011). We went through three rounds of coding. The process of creating codes was emergent. The forum data was not always focused, and conversations could easily go beyond the topic. Yet, such conversations had valuable data that could help researchers better identify the links between and understand learning potentials of history-based video games. Therefore, themes and codes evolved from the data and emergent coding was applied. To ensure reliability and validity of our findings, we triangulate our findings from multiple sources of data between the researchers. Two researchers initially identified codes and categories in line with the research goal. Two other researchers involved in this research independently checked the themes, codes, and interpretations. Then researchers discussed the interpretations for the consensus. Based on the goal of this research, authors discussed and refined the themes and keywords until reaching an agreement to consider selected ones for inclusion in the study. In the findings, we included the comments and discussions relevant to the players' inquiries into history. Later, we used these findings to compare the learning potentials and values of each selected game for the players using the plot in Fig. 2.

Players had commented on several aspects of the games, such as narrative, the backstory of characters, historical values and veracity, game environment, atmosphere, etc. Using Nvivo 11[4] to analyze every sentence of the online discourses, we could organize the codes and themes based on their similarity (Fig 1).

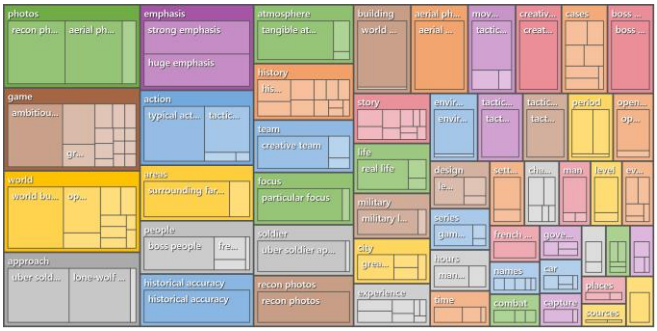


Fig. 1. Hierarchy chart, generated by NVivo 11, showing the keywords.

Overall, 1072 were coded. The frequency of the codes and themes were then extracted from NVivo 11. To exemplify, the frequency of direct and aggregated references to "authenticity" was 239. This number was 170 for "space", 150

[3]Steam is an online distribution platform developed by Valve Corporation that provides digital rights management, multiplayer gaming, video streaming and social networking services.

[4] NVivo is a computer software package designed for researchers working with very rich text-based and/or multimedia information, where deep levels of analysis on small or large volumes of data are required.

for "people", 100 for "activity", 95 for "time, and 78 for "event". The frequency of the references that we later categorized under "learning the history and social history through the game" was 429. This number was 266 for "visiting the historical time and space", and 248 for "learning history by discussion and decision-making". This number was 183 for "finding new ways of doing", and 117 for "sympathy for soldiers and citizens". These themes represent most of the keywords repeated in the online discourses that we studied.

4 FINDINGS

The findings are categorized and discussed below with a variety of textual materials cited from the players in relation to each game. We used pseudonyms (derived from mythology and written in italics) to cite players.

4.1 "A huge labour of love" – Learning the history and social history through the game

Players noticed the effort to include accurate representations in the games. Martin and Turcot (2015) clarified that professors and historians from France helped to develop the world of the Assassin's Creed Unity (ACU) and the everyday life of Parisians during the French Revolution[5]. Players seemed to know about such effort when *Hera* delineated "[w]hen the French re-create France you can assume you'll have a fairly accurate depiction." Many players such as *Thetis* and *Aphrodite* acknowledged the "real-life events you'll see in Assassin's Creed Unity". They might be pointing at the people's lives on the streets during the French Revolution. The information about people's real-life events shapes a considerable part of the social history. Players also found the architecture in the game realistic and accurate. For *Scamander*, "[t]he depth in details to architecture was outstanding" whereas for *Athena*, "[t]he story was just there to keep me going but I really played just for the architecture and visuals."

Rockstar Games[6] confirmed that the events in LA Noire were recreated based on real evidence, documents, and film recordings. The players also acknowledged this in their comments. *Merlin* mentioned that "[l]ast I heard, 90% of the cases were based on genuine crimes." *Iseult* stated that "[a]parently the main dude behind LA Noire read through about 1500 newspapers to get his facts accurate, so I would say a lot of them are very close to being accurate." The players appreciated authenticity and accuracy of LA Noire's representation of Los Angeles beyond the story. Poseidon who lives in LA mentioned that "[l]iving in Hollywood makes it pretty cool to just drive the LA Noire neighborhoods, there was A LOT of work put into the city." His comment reveals the possibility of understanding the urban development of this city over time. *Dido* expressed that "[o]ver time, L.A. Noire has proven to be a masterpiece of recreation, as one thing that the game certainly did well was honoring its source material."

The Saboteur represents Paris during the World War II (WWII) occupation and recreates the atmosphere of the time. Although the story of the game is not fully accurate, the activities of the French Resistance, everyday life of Parisians at the time, and the look of the city during the occupation are true to history. *Polydorus* mentioned that the game is "based on many of the real events" and shared knowledge about the main character: "Sean Devlin is based off an S.O.E.[7] operative named William Grover-Williams, a race car driver who help organize the resistance movements and plot sabotage cells."

As a first-person shooter game, Call of Duty: World at War (COD), pays attention to a variety of real events and important battles of the WWII such as the battles of Stalingrad and Okinawa. The game allows playing alongside historical groups involved in the WWII such as Russian and American troops. *Gaheris* acknowledged that "[t]he campaign is really cool, and it proves the struggle that the Russians had during the Germans' rise to power." Several players, such as *Epaphus* and *Galeshin*, explained that this game taught them more about WWII than their history class at school.

The players of Brothers in Arms: Road to Hill 30 (BIA) compared the game character with an actual historical figure and acknowledged its legitimacy. *Diomedes* claimed that the main characters of the game represent, "a paratrooper part of the 101st Airborne, 502 Regiment, 2nd Battalion." Sharing his inquiries about the game, *Calchas* explained that "[m]any man-hours were spent pouring over old aerial reconnaissance photographs, in order to recreate the landscape as accurately as possible." According to *Nestor*, "they replicated major battlefields down to the closest details of its time", whereas *Teucer* pointed out that "[t]he producers of this game put a huge labour of love into producing this way back in 2004/05 with numerous trips to Normandy, researching, visiting battle sites, testing weaponry etc. and all these add to the realism." In response to *Teucer*, *Agenor* wrote that "[e]ach battle scene in the game was actually a carbon copy of the real battle site. So, the producers did definitely put a lot of effort into it."

Players put their energy into exploration as well as evaluation of the game events against the historical events, which may involve various learning outcomes, such as content understanding and problem-solving. Their evaluations about the historical events behind these games often resulted in acknowledging the effort by the game developers to create accurate representations.

4.2 "This game made me care about its citizens" – Sympathy for soldiers and citizens

BIA seemed to engage players emotionally and intellectually in the life story of the soldiers. For example, *Belus* mentioned that "[t]he game is remarkable not just for the authenticity, but the true to life story of the soldiers." Other players, such as *Theano*, responded in agreement: "it is probably due to the recreated locations and the stellar voice acting." It seems that some players engage deeply with space and the people depicted in the game. Some players, such as *Sarpedon*, delineated that "when you realize that you care more about your men than yourself is a

[5] https://www.ubisoft.com/fr-ca/
[6] https://www.rockstargames.com/
[7] The Special Operations Executive (SOE) was a British World War II organization.

signal that you got the spirit of the game." Similarly, a player of The Saboteur, *Caradoc*, expressed that "the best part of this game and why I keep playing it is that this game made me care about its citizens." According to Kim & Kim (2010, p.14), "having emotions would mean having certain evaluative appraisals of, for example, a person, which indicate having knowledge about the person's characteristics." The feeling and understanding that *Sarpedon* pointed to as well as the sympathy that *Sarpedon* and *Caradoc* described can be considered as the effective learning outcomes of such video games.

4.3 "It's like time travel" – Visiting the historical time and space

Some players took their freedom to explore the open world of the games rather than doing the mission. In discussing ACU, *Iris* mentioned, "as a Parisian, this game gives me the possibility to see some missing places in Paris, such as the Tuileries castle or the court of miracles." Another player, *Pandarus*, explained that "[t]his is why I play Assassin's Creed. To visit places in the past that don't exist anymore. Just walking around Paris during the French Revolution is incredible. Reading the newspaper, listening to people talking in a café, listening to someone making a speech about the politics of the day. It's like time travel." The activity and space (i.e., free roaming in Paris), as well as the time and event (i.e., during the French Revolution), provide the ground for an immersive play.

The case for BIA was different. As the game narrates stories about historical battles, revisiting the history was limited to the atmosphere of the battlefield. *Lycaon* said, "after being in the military for a few years, I found myself coming back to BIA".

Similar impressions were also reported about LA Noire. *Lancelot* played this game with his father who was born in the 1940s in Los Angeles (LA). *Lancelot* explained that he got to learn about his grandfather while playing and having conversations with his father. His grandfather was a police officer in the 1940s and shot a person in LA. He mentioned, "[i]t was the only time my grandfather ever used his gun - and that's just one of the things I learned playing L.A. Noire with my dad a few weeks back." *Lancelot* continued, "[l]ike any son with a father in his late 60s, I assumed his sudden silence meant he was having a minor cardiac event. He wasn't, however: he was simply back in the presence of a building he hadn't seen in half a century." Two players even went further and played the game alongside their grandmothers who lived in LA in the 1940s. *Abas* explained that "one of many reasons I very much enjoyed L.A. Noire, despite its occasional faults, is that I was able to do this too with my grandmother who was a teenager when the game is set." Another player, *Ocalea*, delineated, "I played and drove around, for my grandmother who grew up in LA in the 1940s, and she said it was spot on and loved seeing so much of her world recreated like that." Such activities by players (i.e., playing a game with a person belonging to an earlier generation) echo the affective aspect of learning, which becomes personally more meaningful (Schreiner, 2008). As per Baradaran Rahimi, Levy, and Boyd (2018), such meaningful activities have the power to be applied to museums both during museum visit event and before/after the visit.

4.4 "Getting back to the feel of problem-solving" – Finding new ways of doing

Some games provide players with an open-world to explore (e.g., LA Noire and ACU) and others offer players multiple ways to accomplish a mission (e.g., The Saboteur and BIA). Regarding The Saboteur, *Gareth* explained that "[t]he sandbox quality of this game gives you lots of 'play' time, thinking of different ways you can take out an installation." Other players, such as *Pelleas* and *Zeus*, shared a very similar view. For *Pellinore*, finding different ways of accomplishing The Saboteur missions was an attempt to solve a problem: "I'm finally getting back to the feel of the problem-solving."

Similarly, LA Noire's game mission is for the players to solve criminal cases. As *Apollo* explained, "[e]verything revolves around your own willingness to get to the truth: your ability to find every clue, check every nook and cranny, and get answers from your witnesses and suspects." Players could also develop their own method to solve the cases. According to *Ares*, "[a]fter investigating the scene I would always take a few minutes to put together what I thought happened so if any of the witnesses said something contrary to my version of events I would know to grill them. It usually worked out well." Seemingly, players are in a position to make (good or bad) decisions and deal with the consequences of their decisions in LA Noire.

COD has a potential for players to come up with different ways of playing the game. The players may twist historical content available in the game by creating modifications (mods). This could provide players with an opportunity to reflect on historical events or groups involved in the WWII. One of the most popular mods is to make the invaders in COD as zombies. For example, *Brastius* mentioned that "[t]his is all where it started. Zombie mode!" Such a representation may show how those who created this mod twisted the dark impression of the invaders into non-player characters.

4.5 "It's like being back at school!" – Learning history by discussion and decision-making

Sometimes, the online discussions by the players went into details about the history and consequent sharing of information. In discussing ACU, *Diomedes* delineated that "[t]he layout of Paris was a pretty big mess before Haussmann's renovation."[8] This is true as the plan and distinctive appearance of the center of Paris today is largely the result of Haussmann's renovation (Giedion, 2009). In exchanging historical information, players talked about the similarities of a character in Assassin's Creed Unity (ACU) to Napoleon Bonaparte and referred to the history. *Ornytus* clarified in the ongoing discussion that "the rise of Napoleon was from 1802. I believe when he becomes consul then an emperor in 1804 till 1815 where he loses the Waterloo battle." *Cydon*, another player participating in this ongoing discussion acknowledged *Ornytus's* sharing of information

[8] Haussmann's renovation of Paris was a vast public program directed by Baron Haussmann, between 1853 and 1870.

by writing "[g]ood history recap." Another player, *Pallene*, expressed his amazement about the ongoing discussion by saying "[m]an, it's like being back at school!" Such ongoing discussions raised in a game forum is promising and shows that players not only play the game but also pay attention to the historical content, collect information from sources, and examine the information through the discussions. Similarly, the historical context of BIA was also a source of learning for some players. *Laodice* commented, "I learned more about D-Day from the extras menu than I have ever learned from high school history class." *Iphis* explained, "honestly, this game is a sublime gem for a history nut like me."

Discussions about LA Noire was mostly on the LA Police Department cases incorporated in the game, often verifying their authenticity. *Deneter* mentioned, "Mickey Cohen and the Black Dahlia killer are real people." *Dionysus* responded, "[f]rom what I've read, most of the cases were based on real cases but not all were taken from 1947." *Hades* joined this conversation and delineated, "[t]hink of this game like a huge connected series of Dragnet episodes, the cases are real, the names have been changed to protect the innocent." Later, players talked about other cases in a follow up to the ongoing discussion by creating new threads, such as "Red Lipstick Murder" (by *Hestia*) and "Silk-Stocking Murder" (by *Eros*). *Eros* went into factual details of the Silk-Stocking Murder: "[i]n 1947, Rosenda Mondragon was found dead after being strangled by a stocking near the City Hall in Los Angeles." Moreover, online discussions revealed that several players found ways to play the game together, although LA Noire is a single player game. *Juno* explained, "I played until dawn with the family and that was great. I'd pause at key moments and let everyone discuss what we should do." *Artemis* responded, "I agree and have had similar experiences with friends and family who aren't gamers. The discussions that come out of it are great."

5 DISCUSSIONS

The historical references of each game have different potentials for learning. We compare and summarize the findings on how the players might inquire into history through these video games (Table 2). Knowing what players say they have learned from games - even if the information and knowledge are not completely accurate - helps researchers know the type of content that can be learned from such games. For example, when players learned about a historical figure like Napoleon Bonaparte in their forum discussions - even if they learned untrue things about the figure – it means that history-based video games are potentially useful for teaching about historical figures (McCall, 2018; 2016; 2012). This can be extended to other aspects of history (e.g., sights, sounds, spaces, and interactions) that may exist in a first and third-person shooter game that unfold in the context of history.

Based on the comments and the discussions raised by the players, it seems that players pay attention to and learn from the representations of the historical contexts in the video games. Comments indicate that the authenticity of representations in the historical contexts of the games could be used for several purposes including building understanding about history and historical forces of a time.

TABLE 2
History-based video games learning potentials

TITLE	LEARNING POTENTIALS	HISTORICAL REFERENCE
ASSASSIN'S CREED UNITY	– Learning about the history and social history during the French Revolution – Experiencing Paris scenes from this historical period – Learning the historical details through online discussions	Objects (e.g., outfits), atmosphere (e.g., life during the French Revolution), buildings (e.g., Notre-Dame Cathedral), and communications (e.g., trade)
LA NOIRE	– Learning the history and social history during the 1940s in LA. – Experiencing LA from the 1940s. – Finding ways to solve problems and accomplish the missions – Making in-room group decisions – Learning criminological information through the game and online discussions	Events (e.g., famous crimes), atmosphere (e.g., life in the 1940s), lifestyle (e.g., sociocultural interactions), buildings (e.g., City Hall), and police activities (e.g., investigation methods).
THE SABOTEUR	– Learning the history and social history during the German occupation of France – Sympathizing with situation and people through immersion – Understanding the ways of being French Résistance and doing stealth missions	Landscapes (e.g., Eiffel Tower), buildings (Sacré-Cœur Basilica), events (e.g., the occupation of France during WWII), and people (e.g., French Resistance).
CALL OF DUTY: WORLD AT WAR	– Learning about certain World War II battles – Sympathizing with soldiers through immersion – Finding ways of being and doing the missions of the game	Battlefields (e.g., Stalingrad), groups and troops (e.g., USSR troops), battle style (e.g., Urban warfare), and weaponry (e.g., SVT-40 rifle).
BROTHERS IN ARMS: ROAD TO HILL 30	– Learning about World War II, Mission Albany – Experiencing the battlefield and events – Sympathizing with the soldiers at war – Learning about the historic battle through the game and online discussions	Battlefields (e.g., Normandy), events (e.g., Mission Albany), groups and troops (e.g., the 502nd Parachute Infantry Regiment (PIR) of the 101st Airborne Division), weaponry (e.g., M1911 pistol), and atmosphere (e.g., representing F-Day).

We plot the data (Fig. 2) to visualize the historical reference and authenticity of each game. In Fig. 2, each game is represented on the plot with a certain shape (i.e. indicators). Concentrating on two games (ACU and LA Noire), we show the specific use of the plot for investigating the potentials of these games. For each factor, namely, people, time, space, event, and activity, an assigned shape is located on the plot that can move on the continuum (i.e., the horizontal axis of Fig. 2) towards more, or less, authentic. This continuum is based on the discussions by Metzger and Paxton (2016). Those video games that focus on fantasy shift towards the less authentic end of the continuum on the plot. For example, some versions of the COD like the Zombie modification, take place in a dystopian representation of spaces and events in the past. So, the indicators related to time, space, and event are close to the left side of the continuum. Points enlisted in the callouts may be different from game to game. However, based on the findings of this research and the comments from the players, they play a significant role in making a video game pertinent for particular use as an educational medium. For instance, a video game like The Saboteur may authentically represent the historical occurrences – i.e., event in Tschumi's work (2012) – or buildings, and landscapes – i.e., space in Tschumi's work (2012) – but include fictional characters — i.e., people in Tschumi's work (2012). If the purpose of learning revolves around the historical event or space and buildings, then the game may be a good choice as Pallasmaa (2001) stated that "buildings and cities create and preserve images of culture and a particular way of life". However, if the purpose is to learn about historical figures the game may not be a good choice.

As Fig. 2 shows, ACU integrates fictional protagonist/antagonist into the game although the outfitting style

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TG.2019.2954880, IEEE Transactions on Games

of ordinary people and their social interactions remain loyal to the historical evidence as Ubisoft and players mentioned. The number of players' references to the authenticity of game characters was two. Therefore, on the plot, game characters (first factor: people) appear less authentic on the continuum (Metzger and Paxton 2016). However, the historical time and space (i.e., Paris during the French Revolution) are considerably based on extensive research according to Martin and Turcot (2015). Moreover, for each player, the game could represent the atmosphere of the time, buildings and environment (space) of the game authentically. This recalls Pallasma's (2001) idea of creating and preserving the images of history. Players of the ACU referred to the authenticity of "time" 15 times and to the authenticity of "space" 29 times. The event of the revolution is again fictionalized to some extents and involves fictional characters (protagonist and antagonist groups). The number of references by players to the authenticity of the "event" was three. Therefore, the "event" is located in the mid-low part of the authenticity continuum on the plot. The rebellious activity of the main character in the game has some real and fictional sides as well. Although the character belongs to a fictional society, the nature of rebellious activities of this character, such as guerrilla-warfare and fire/ambush strategy, in the game has similarities in many aspects with the activities that took place during the French Revolution9. Moreover, players referred to the authenticity of such "activities" 12 times. This means that ACU could be appropriate for the learning social history, existing architecture of the time, and the general atmosphere of French Revolution.

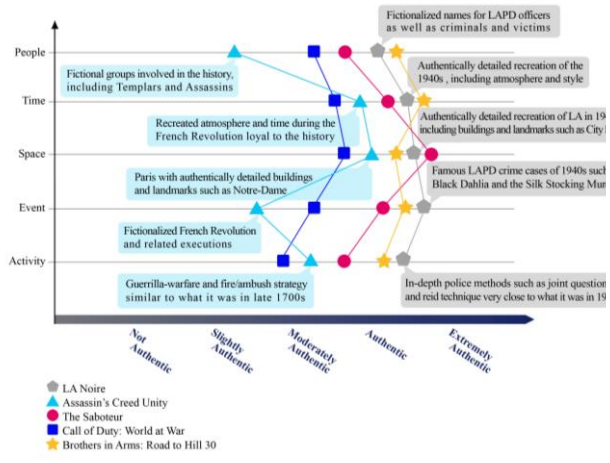


Fig. 2. Completed plot for history-based video games. For each factor, an assigned shape is located on the plot that can shift towards right or left side of the plot. Points enlisted in the callouts may be different from game to game.

According to the findings, LA Noire represents a game with almost every factor close to the right end of authenticity continuum. The number of players' reference to the authenticity of "time" was 23, "space" was 39, "people" was 38, "event" was 23, and "activity" was 31. As the findings pointed out, the atmosphere of the 1940s (time), the city of Los Angeles (space), and the criminal events that happened in that era are authentically recreated in the game based on the evidence and documents. The activity of investigating the crimes are also representative of the methods used by LAPD in the 1940s such as joint questioning and the Reid technique. The outfits of people and their social interactions, as well as the vehicles that people used in the 1940s, are also a good representation of the period (time). However, the main character of the game may not be an exact member of LAPD in the 1940s. This does not dramatically change the authenticity of the game and therefore "people" is placed to the left of – but still close to – the right end of the authenticity continuum on the horizontal axis of the plot. Resonating with Pallasma's (2001) idea about creating and preserving the images of history, LA Noire can be used for learning about the modern history of Los Angeles having depicted with a high degree of accuracy, the atmosphere, architecture, and criminology as well as clothing and vehicles of the 1940s.

Based on the findings from the analysis of the online comments, the Saboteur represents the atmosphere (space and time) of the occupation of France during the World War II (WWII) representative of the reality. The players acknowledged the authenticity of the atmosphere frequently (i.e., "time" 20 and "space" 36 times) in their comments. Moreover, space in this game is quite close to the Paris during WWII. Everyday life of people in occupied Paris, as well as the activities of French Résistance at the time of occupation, are also partially representative of the reality in that era. Players of this game referred to the authenticity of "people" 32, "event" 18, and "activity" 20 times in their comments. Therefore, all factors of this game are located from the middle to the more authentic side of the continuum on the plot (Fig. 2). The Saboteur seems to be useful for learning about general atmosphere and urban landscape as well as the history of France and related events during the German occupation, regarding Pallasma's (2001) idea about the images of history.

Given the findings about the COD, the time and events of the selected battles during the WWII are relatively authentic regarding Metzger and Paxton's (2016) deployments. The number of references to each of these two factors was 13. The ruined buildings and cities in the game (space) are also considerable as they help a lot to reinforce the authentic atmosphere and events of the time. However, disciplinary practices are mixed with fictionalized facts. For example, the activity of fighting with the enemy and the soldiers in the game are partly fictionalized. Players acknowledged the authenticity of "space" 25 times in their discussions. Since the characters change from time to time in the game and there are few links between the narrative of the game and the characters (people), these two factors, people and activity, represent the lower level of authenticity. Yet, the uniforms of the soldiers and their weaponry are authentically represented in the game. The number of players' references to "people" was 24 and to "activity" was nine. Therefore, Call of Duty: World at War (COD) can serve some educational purposes that are focused on selected battles of the WWII, such as the battle of Stalingrad, as well as the general atmosphere of WWII in the battlefields, including weaponry,

https://www.britannica.com/topic/strategy-military/The-French- Revolution-and-the-emergence-of-modern-strategies

related events, and characteristics of regiments belonging to different countries, such as Germany, Russia, Japan, and the United States.

Based on the findings, Brothers in Arms: Road to Hill 30 (BIA), paid considerable, attention to the authenticity. The number of players references to the authenticity of "time" was 24. This number was 41 for "space", 37 for "people", 21 for "event", and 28 for "activity". This resonates with Metzger and Paxton's (2016) deployments. The players acknowledged lots of research and assessment of historical evidence and documents to support the game development. People represented in the game are the 502nd Parachute Infantry Regiment of the famed 101st Airborne Division who were dropped behind German lines (place) on D-Day (time). The game is based on the historical Mission Albany (event) and the player must accomplish the mission (activity). The people, place, event, activity, and time are all historically accurate. Although the game revolves around a limited chunk of the WWII events, spaces, people, and time, it delivers certain values, such as sympathy and caring about soldiers, to the players. Many details, such as the uniforms of the soldiers and their weaponry are authentically represented in the game. Resonating with Pallasma's (2001) idea, the factors of BIA are located towards the more authentic side of the continuum on the plot. This game can be used for educational purposes that concentrate on the study of a particular mission during the WWII. It can serve as a medium to learn about the general atmosphere of the D-Day in the battlefield, weaponry, and characteristics of the 502nd Parachute Infantry Regiment of the 101st Airborne Division.

The outcomes for players include building understanding about history and historical forces of the time, through personally relating to the specific events, social artifacts, and places. Revisiting history in games and post-play discussions could help understanding of historical details. Through their discussions, the exchange of information and knowledge took place. Teamwork to go through a historical event, mission, or mystery was another considerable part of the outcome. Sometimes teamwork was focused on multiplayer mode and sometimes it was focused on playing a single player game with a friend, parent, or a grandparent who was familiar with the specific part of the history represented in the game. Such a teamwork could help them to learn about the history by doing it along with a person familiar with certain historical events. Sympathizing through their immersive game experiences were also part of the outcomes for some players. These outcomes involved caring about others and feeling better about one-self. Moreover, finding new ways of being, such as being a game modifier, could help some players to develop new identities. More explicitly, this process let players see themselves as co-designers implementing a modification of a game to share with others and get feedback from them. In addition, finding new ways of doing the game mission was obvious in the creative ways that players could implement to play a game beyond its intended purposes. It also helps to find their own ways to engage in deeper learning, drawing on their desire. For example, several players refused to play the mission of a game and instead explore the open world of the game to visit the architectural structures and historic monuments.

6 CONCLUSIONS

A considerable amount of online conversations in game forums can be helpful in evaluating players' responses to specific occurrences in the game. These conversations – even if they are not completely accurate - contain valuable points that can help researchers, educators, game developers, and designers understand how casual players play and learn from a commercial game.

This paper explored a certain type of history-based video games (i.e., first and third-person shooters) to understand the learning potentials for the casual players in relation to the historical authenticity and factors of the historical context in such games (i.e., people, time, space, event, activity). Players revealed they inquire into history using different resources such as online archives, documentaries, history books, and verifications from those who lived a certain era. A sample of five commercial games was considered for this study (Table 1). A plot was then created to visualize the learning potentials of each game and to compare them together. Using NVivo 11 data linked to each game was analyzed (Fig. 1) and the relative position of factors towards the ends of the authenticity continuum (i.e., the horizontal axis of the plot) was determined (Fig. 2). As a tool, the suggested plot can provide an overview of learning potentials to facilitate comparison and selection between the history-based video games for the use as an educational platform (Table 2). Future studies may add to the factors or go beyond the authenticity of the games.

Undoubtedly, there are other aspects of learning through video games that are not depicted in this paper. These aspects include a range of skills such as problem-solving and strategic decision making. For future studies, researchers may want to investigate the role of historical authenticity and factors of historical context in developing such skills. Sociocultural differences and preferences in the learning outcomes, which was informally witnessed during the investigations of this study, may have impacts on how players look at learning values of the games. For instance, when players made modifications to the COD, the number of zombie modifications surpassed the other modifications, which could have been influenced by popular media and gender.

In terms of using video games for learning purposes, the outcomes of this research suggest that they have the potential for particular proposes such as learning about certain people, spaces, events, details in the history, social history, and alternative histories. Some games we discussed in this paper, such as LA Noire, may have potentials to be used as a platform in museums for learning and enjoyment. One may continue this work with testing this in an exhibition. Moreover, this paper may be considered as an attempt to motivate game developers to more deeply look at how their games may influence players and learners beyond the entertainment purposes. Developers and game designers may want to provide educational versions or special editions of their games to be used in educational or cultural institutes such as universities and museums. Commercial games have a great potential for in-depth learning, as they incorporate the vast knowledge of experts, such as historians, archivists, and librarians, as well as university professors in the fields of history, social history, and digital media. Maximizing this

vast knowledge of experts may help players learn about history. We hope that our work contributes to the scholarly discussion on design and application of games among multiple disciplines and stimulates further exploration on the synergistic work between industry and academia.

ACKNOWLEDGMENT

The authors wish to thank the Conjoint Faculties Research Ethics Board (CFREB) of the University of Calgary for providing advices on the anonymous use of the information available on web-based communication services. Special thanks go to anonymous reviewers of this paper for their entirely insightful comments and feedback.

REFERENCES

[1] J. McCall, "'Teaching History with Digital Historical Games: An Introduction to the Field and Best Practices'," *Simulation & Gaming*, vol. 47, no. 4, pp. 517–542, 2016.

[2] T.M. Connolly, E.A. Boyle, E. MacArthur, T. Hainey, and J.M. Boyle, "A systematic literature review of empirical evidence on computer games and serious games," *Computers & Education*, vol. 59, no. 2, pp. 661-686, Sep. 2012.

[3] J. McCall, "Video games as participatory public history," in *A Companion to Public History*, D. Dean, ed, Hoboken, NJ: Wiley & Sons, pp.405-416, 2018.

[4] J. McCall, "Navigating the Problem Space: The Medium of Simulation Games in the Teaching of History," *The History Teacher*, vol 46, no. 1, pp. 9-28, 2012.

[5] H. Desurvire and M. S. El-Nasr, "Methods for Game User Research: Studying Player Behavior to Enhance Game Design," *IEEE Computer Graphics and Applications*, vol. 33, no. 4, pp. 82-87, July-Aug. 2013.

[6] A. Chapman, A. Foka, and J. Westin, "Introduction: what is historical game studies?" *Rethinking History*, vol. 21, no. 3, pp. 358-371, 2017.

[7] K. Squire K and S. Barab, "Replaying History: Engaging Urban Underserved Students in Learning World History Through Computer Simulation Games," in *Proceedings of the 6th International Conference on Learning Sciences*, pp. 505–512, 2004.

[8] C.A. Anderson, "An update on the effects of playing violent video games", J. *adolescence*, vol. 27, no. 1, pp. 113-122, Fen. 2004.

[9] S.M. Ogletree and R. Drake, "College students' video game participation and perceptions: Gender differences and implications,". J. *Sex Roles*, vol. 56, no. 7/8, pp. 537-542, Mar. 2007.

[10] M. Griffiths and M.N. Davies, "Research note Excessive online computer gaming: implications for education," J. *Computer Assisted Learning*, vol 18, no. 3, pp. 379-380, Dec. 2002.

[11] K. Schreiner, "Digital Games Target Social Change," *IEEE Computer Graphics and Applications*, vol. 28, no. 1, pp. 12-17, Jan.-Feb. 2008.

[12] E. Zimmerman, "Gaming literacy: Game design as a model for literacy in the twenty-first century," *The Video Game Theory Reader 2*, M.J.P. Wolf and B. Perron, eds, London: Routledge, pp. 23–31, 2009.

[13] F. Baradaran Rahimi and B. Kim, "The role of interest-driven participatory game design: considering design literacy within a technology classroom," Int. J. *Technology and Design Education*, submitted for publication, Apr. 2018.

[14] S.A. Metzger and R.J. Paxton, "Gaming history: A framework for what video games teach about the past," *Theory & Research in Social Education*, vol. 44, no. 4, pp. 532-564, Aug. 2016.

[15] B. Tschumi, *Red is not a color. Architecture concepts*, New York: Rizzoli International Publications, 2012.

[16] M. Gerosa, "Degrees of virtualization," *Space between people: how the virtual changes physical architecture*, S. Doesinger, Ed, Munich: Prestel, pp. 46-55, 2008.

[17] J. Pallasmaa, *The Architecture of Image: Existential Space in Cinema*. Helsinki: Rakennustieto, 2001.

[18] S. Wineburg, *Historical thinking and other unnatural acts: Charting the future of teaching the past*. Philadelphia: Temple University Press, 2000.

[19] B. Walsh, "Signature pedagogies, assumptions and assassins: ICT and motivation in the history classroom,"in *Using new technologies to enhance teaching and learning in history*, T. Haydn, ed, London: Routledge, pp. 131–142, 2013.

[20] J.P. Gee, "Video games and embodiment," *Games and Culture*, vol. 3, no. 3/4, pp. 253-263, Jul. 2008.

[21] D. Arsenault, "Video game genre, evolution and innovation," *Eludamos Journal for Computer Game Culture*, vol. 3, no. 2, pp. 149-176, Apr. 2009.

[22] H. Lowood, "Storyline, dance/music, or PvP? Game movies and community players in World of Warcraft," *Games and Culture*, vol. 1, no. 4, pp. 362-382, Oct. 2006.

[23] K. Squire, *Video Games and Learning: Teaching and Participatory Culture in the Digital Age. Technology, Education--Connections (the TEC Series)*. Amsterdam: Teachers College Press, 2011.

[24] W.R. Watson, C.J. Mong, and C.A. Harris, "A case study of the in-class use of a video game for teaching high school history," *Computers & Education*, vol. 56, no. 2, pp. 466-474, Feb. 2011.

[25] J.C. Martin, and L. Turcot, *Au cœur de la Révolution. Les leçons d'histoire d'un jeu vidéo*, Paris, Vendémiaire, 2015.

[26] *B.* Kim, and M.S. Kim, "Distributed emotions in the design of learning technologies," *Educational Technology*, vol. 50, no. 5, pp. 14-18, Oct. 2010.

[27] F. Baradaran Rahimi, R.M. Levy, and J.E. Boyd, "Hybrid Space: An Emerging Opportunity That Alternative Reality Technologies Offer to the Museums," *Space and Culture*, Aug. 2018. Available: https://doi.org/10.1177/1206331218793065

[28] S. Giedion, *Space, time and architecture: The growth of a new tradition*. Cambridge, Harvard University Press, 2009.

Farzan Baradaran Rahimi is a Ph.D. candidate in the Computational Media Design (CMD) program at the Faculty of Science, University of Calgary. His background is in architecture and his academic interest lies in the interdisciplinary studies.

Beaumie Kim is an associate professor of the Learning Sciences at the Werklund School of Education, University of Calgary. Her work is focused on engaging learners in designing learning resources and tools using games.

Richard M. Levy is a Professor of Planning and Urban Design at the University of Calgary. He is a founding member of the Virtual Reality Lab and was a Co-Director of the Computational Media Design Program at the University of Calgary.

Jeffrey E. Boyd is an associate professor and Associate Head in the Department of Computer Science at the University of Calgary. He was previously at the Visual Computing Laboratory at the University of California, San Diego, and the Department of Computer Science at the University of British Columbia.