Using the Unreal Engine to build playful cities – how the modding experience of Levee Patroller can be applied to sustainable urban development

Position Paper submitted to the seminar
THE PLAYFUL CITY – SERIOUS GAMING FOR SUSTAINABLE URBAN DEVELOPMENT
November 29, 2007, TU-Delft

Casper Harteveld
Delft University of Technology & GeoDelft

ABSTRACT
Modding, or the ability to adjust/reuse existing technologies and content, has created an enormous potential for serious game applications. With this technology it suddenly becomes much more accessible to use gaming technology. This paper discusses the experiences of using one of the most popular modding technologies, the Unreal Engine, for building a game called Levee Patroller. The paper also looks into how these experiences can be applied to sustainable urban development. It turns out that this game engine can be considered a powerful tool for building playful cities, but one which is costly and which has some limitations.

AUTHOR KEYWORDS
Game engine, modding, serious games, Unreal Engine.

MODDING, THE FUTURE’S LANGUAGE
In 1974 Duke’s influential book “Gaming, the future’s language” was published. In this book Duke argues that games are a good way of representing and understanding complex situations in reality. More than a quarter of a decade has past since the publication of Duke’s book, but his central idea still holds. In fact, his idea is more alive than ever with the still steadily increasing interest in what has been called “serious games”.

Serious games are broadly defined by Sawyer (2002) as “using digital game entertainment techniques for non-entertainment purposes”. This definition implies that serious gaming does not need to involve around a “game” per se. Building a visualization tool to show a model of a city using game entertainment techniques can be considered serious gaming just as much as developing a training application for fire workers. Nonetheless, the main idea behind the serious game movement and the reason for its enormous attention worldwide is that these digital game entertainment techniques can be used for a wide variety of societal applications, from medical to educational. This ideology of applying gaming techniques to real world situations is very reminiscent of Duke’s original hypothesis that games help in understanding reality.

Despite the striking resemblance, things have changed drastically since 1974. Instead of using analog games we are of course able to develop digital ones. More importantly, since that period people have become much more conscious about the power of this “future’s language”. This has led to an increased interest of using gaming techniques in all sorts of ways by people who do not have a specific affiliation with games. Although it is difficult to exactly proof what factor was most crucial in creating this awareness, an important contribution has definitely been the various technological innovations that have made it easier for lay people to use gaming technology.

Probably the most influential development of these various technological innovations is the ability to modify existing games or use existing content, functionalities or other parts of a game in another context. This type of modularity and flexibility in gaming technologies is referred to as “modding”. Modding makes it possible to use gaming technology in a fairly easy way. Many functions, such as network connections, are already taken care of and it takes a couple of mouse clicks to add or change content. This means gaming technology has suddenly become more available to lay people as it involves less programming, less development time and leads to better results (i.e., better graphics, less network failures etc.).

Due to the large potential modding technologies have created for serious game applications it is arguable that we can speak of modding as the new future’s language and as a complimentary one to the language of gaming. Gaming makes it possible to model complex situations in reality and modding makes it possible to realize gaming applications. Although modding sounds as a very promising language in the area of serious games in theory, it is still important to acknowledge what the use of this language implicates in practice and in particular for building playful cities. For that reason, this paper will discuss the
experiences of using a popular modding technology, the Unreal Engine, for building a game called Levee Patroller (Harteveld, Guimarães, Mayer & Bidarra, 2007).

THE UNREAL ENGINE

Back in October 2005 the geo engineering research institute GeoDelft came up with the idea of using gaming technology to train levee patrollers. Levee patrollers are people who inspect the quality of levees. This inspection is quite important as levees are infrastructures that protect the land from flooding. Training these people is difficult as levee failures do not occur that often. With a game it becomes possible to train these unexpected situations. Other partners, such as Delft University of Technology, recognized the potential of this idea and after finding the right people the development started in February 2006.

Figure 1: The editor of the Unreal Engine

One of the initial decisions of the design team, which consisted at the start of four people (a game designer, a programmer, a project leader and a part-time modeler), was to use the Unreal Engine. The Unreal Engine is the game engine behind commercially successful games as *Unreal Tournament* and *Gears of Wars*. Game engines form the core of a game; they describe all the functions that a game has. Frequently a game engine is accompanied with an editor to make it easier to add, change or delete content (see Error! Reference source not found.). By using this engine many functions were already taken care of. On top of this we received an excellent tool (i.e., the editor) to develop the game.

Many other game engines exist next to the Unreal Engine. The main reason for choosing this engine is based on the good experience of creating a mock-up prototype in the summer of 2005. One modeler created in only one month a fictional three dimensional landscape with levees and with some interactivity, such as flooding the area, driving a truck and retrieving information. Besides this experience, we noticed that a great number of serious game developers have chosen to use the Unreal Engine as well. Good examples are Carnegie Mellon’s *Hazmat: Hotzone*, Dartmouth’s *Unreal Triage* and of course *America’s Army*.

Figure 2: A levee failure in Levee Patroller

During the development we quickly came to the conclusion that despite using a game engine three and half people were insufficient to reach our initial deadlines. We expanded the team in April 2006 with one full time game developer, another part-time modeler and three computer science bachelor students. With this team we finished the first version in October 2006. In total it thus took 9 months to develop the game, which has been called *Levee Patroller: The Levee Inspection Simulator* (see Error! Reference source not found.). The total costs of this development can be roughly estimated at 150,000 euros. Other than telling facts about costs, team size and development time, the development of *Levee Patroller* also led to many insights about the power and limitations of modding. These insights can proof very valuable for judging to what extent the Unreal Engine or modding in general is valuable for building playful cities.

THE POWER OF MODDING

For serious game developers the foremost power of modding is that they can reuse existing content and technologies. This saves time and enables developers to focus on their specific content rather than on all kinds of technical issues. A good example of a function which is already present is importing models from drawing tools, such as 3D Studio Max and Maya. More game related functions are also present, such as creating a character and controlling this by using a keyboard and a mouse. Due to this, it is possible to build and walk around in a simple three dimensional city in just one day.

More difficult functions, like changing a building over time or moving objects around in a city (e.g. a car), can be implemented without touching a single code. The engine comes with standard building blocks and these can be used to implement these types of functions. Of course, if these standard building blocks are not satisfying some code needs to be written to slightly adapt the function. More
specific functions may require a new set of codes. For example, for Levee Patroller complete new lines of code needed to be written to describe the interactions of the player with levee failures.

Figure 3: Screenshot of a map in Levee Patroller

Another powerful characteristic of the Unreal Engine (but which can be limiting too) is that it provides a framework of what is required in a game. It comes for example with standard menus, which can be easily adapted. These standardized aspects show what has to be kept in mind for the development of a game. This makes the development clear and transparent.

When looking at the contribution of gaming to sustainable urban development the advantages of the Unreal Engine become immediately apparent. Most urban development plans involve creating digital mock-ups and since these can be easily imported into the Unreal Engine, it is just a matter of a couple of mouse clicks to make it possible to virtually walk around in the original static mock-up. Such virtual tours can be used for participatory evaluation with citizens, companies and other stakeholders.

On a more advanced level it is possible to create a real game like SimCity. For example a game in which a player has to make certain choices about housing, vegetation and transportation. These choices are then built into the virtual environment and affect a number of conditions, like the general happiness of the population. By looking at the results of different game sessions it is possible to see if patterns can be recognized among players. An analysis of these patterns could consequently lead to a better understanding of urban development. Portugali has used such an approach, although he focused purely on simple analog games. With modding technologies as the Unreal Engine it becomes possible to recreate these experiments into a virtual environment or to simulate even more complex ones.

LIMITATIONS OF MODDING

Modding sounds easy and promising, but it literally and figuratively comes with a high price to pay. Although many excellent open source game engines exist, such as the Torque Engine and Ogre 3D, the high quality engines, like the Unreal Engine or the Source Engine, are not really freely available. It is possible to create a mock-up for “free” with the Unreal Engine, like we did in the summer of 2005. The game Unreal Tournament 2004 is, like most games nowadays, accompanied with a game editor. With this editor players are able to create their own maps or add content to the game, such as weapons or vehicles. It can however also be used for serious purposes. The disadvantage of using a game engine in this way is that the original game is always required to use the serious game application and that this application can only be accessed through this original game. In case of Unreal Tournament 2004, this means users first have to go through menus full of violent images of robots and humans before getting to the serious content (see Error! Reference source not found.). This is obviously unwanted.

Figure 4: Menu screen of Unreal Tournament 2004

For making a stand-alone version the game engine needs to be purchased. Different price categories exist; the more expensive licenses come with more accessibility to the code and more freedom in distribution of the software. For commercial distribution and full access to the code the costs are calculated at 100,000 dollars per installment and a certain profit percentage of the sales. These costs are enormous for a research institute, and so for Levee Patroller we initially purchased the cheapest license (10,000 dollars per installment). This enabled us to create a stand-alone version and to distribute the game internally. Sadly, this license also comes with limitations in accessing the code. Because of this, we were unable to change fundamental aspects of the engine, such as the basic structure of the menus, and to connect other software to the game. One of the ideas was to connect existing geo engineering software to the game to simulate levee failures. Currently, we have
made an extra purchase (another 10,000 dollars) that enables us to connect other software to the game.

For sustainable urban development the possible connectivity with other software can be considered a major issue. Many good simulation models exist and it would be a waste if these have to be implemented again in a gaming context. Besides saving time, it can be expected that these simulation models are more accurate than their gaming conversions. Unfortunately, next to the costs involved in making it possible to connect software, many concurrency and distribution issues will arise. If, for example, the time in a simulation goes 100 times faster than in the game environment, a proper conversion should be made to make the two systems fit.

Another striking limitation is that the game engine is particularly written for one single type of game. For the Unreal Engine this concerns the First Person Shooter (FPS) series *Unreal Tournament*. The functions in the game engine are optimized to build these types of games. Probably due to this we frequently bumped into problems of implementing some desired features that are not needed to build these FPS games. For instance, it turned out to be impossible to put grass on levee slopes. It was only possible to put these on flattened surfaces. Another problem concerned the implementation of failures. We wanted to build a completely randomized map. For this reason, the maps were built out of similar building blocks. In the end, it appeared however to be impossible to really randomize the failures. We could only turn failures on or off in a particular building block.

These limitations indicate that it may become difficult to implement certain ideas for sustainable urban development with the Unreal Engine. It is of course difficult to know what types of problems are going to occur beforehand, but knowing from the beginning that certain ideas might not be feasible to implement will lead to another way of working. One in which the key ideas are first roughly tried out in prototypes, before the real implementation is started. In addition, the good thing about the Unreal Engine in comparison to other engines is that it has a large community. This community can help in solving these sorts of problems and this makes it more likely that a playable city can be built as initially planned for.

**CONCLUSION**

Nobody wants to reinvent the wheel and that is why modding can be seen as the future’s language in the area of modeling, simulation and gaming. It enables to adapt or reuse existing content in a fairly easy way. This possibility makes clear why modding is such an important contributor to the serious games movement. With this technology it suddenly becomes much more accessible to use gaming technology, for instance for sustainable urban development.

In this paper we examined one of the most popular modding technologies, the Unreal Engine, and how the experiences of using this engine can be applied to building playful cities. In sum, we can conclude that the Unreal Engine is a powerful technology; it makes it possible to rapidly build a high quality game. However, it has some technical limitations and it is quite costly. This shows that modding will not make us capable of “building Rome in just one day”. Nonetheless, it does make us capable of building a virtual version of Rome with much less effort than before.

**REFERENCES**

3. Portugali

**AUTHOR INFORMATION**

The author Casper Harteveld (c.harteveld@tudelft.nl) is a PhD researcher in the faculty of Technology, Policy and Management at Delft University of Technology. Currently his research focuses on the interrelationship between games and learning processes from an organizational and socio-cognitive perspective.

**ACKNOWLEDGMENTS**

We would like to thank the project participants at GeoDelft and Delft University of Technology, Rafael Bidarra, Micheline Hounjet, Jos Maccabiani, Igor Mayer, Raymond van der Meij and everybody else – none named, none forgotten – who have contributed to its ongoing success.

**MORE INFORMATION**

2. About the game *Levee Patroller*: [http://www.delftgeosystems.nl/leveepatroller](http://www.delftgeosystems.nl/leveepatroller)