Location-based Game Design Pattern Exploration Through Agent-Based Simulation

Thomas Heinz
Research Group on Computing in the Cultural Sciences
University of Bamberg, Germany
thomas.heinz@uni.bamberg.de

Abstract

The development and balancing of location-based games is challenging, because the gameplay depends on local environmental conditions. A tool based analysis of common game mechanics could provide a remedy for this problem, but requires precisely defined gameplay patterns. This paper presents systematic selection of spatio-temporal game design patterns together with a software tool, able to simulate location-based gameplay in different spatial environments.

Keywords: Geogames, Location-based Games, Game Design Patterns, Agent-Based Simulation

1 Introduction

In video- and board game design, patterns are long established tools. These design patterns are used to gather inspiration, communicate with peers and professionals or solve design as well as interaction problems. Extensive collections for these types of games have been compiled by various authors (e.g. Bjork/Jussi 2004). When it comes to pattern libraries, location-based games have not received an equivalent amount of attention although first attempts have already been made to compile lists of patterns that cover design aspects new to mobile location-based games (Will 2013, Davidsson 2004). The authors gathered these patterns by analysing a selection of existing games. Patterns discovered that way are related to a wide variety of very different aspects – starting from technical related features up to the social consequences of putting a player into an outdoor environment. However, authors stress that these pattern lists are far from being complete and have to be extended continuously by increasing the number of analysed games. Most pattern languages give the definition, usage description and resulting outcome to a reoccurring game mechanic. Typically, these facts are described textually, but are not defined in any kind of formal language (Dormans 2013). The issue of giving a formal description of a defined subspace of design problems has not been sufficiently addressed by research so far. Using formal description, location-based game patterns could be implemented into software tools, that would be able to support game designers in exploring strength and weaknesses of given designs just by adjusting various parameters. Tools like these could overcome the difficulties, developers face in location-based game development (Jacob/Coelho 2011).

In this paper I present a systematic approach for the identification of a subset of spatio-temporal design patterns for location-based games. By means of a self-imposed restriction to the examined problem space, I am able to give pre- and postconditions for each pattern. There are no detailed descriptions for every pattern in this work, instead an exemplary account of only one pattern is included. Rather than concentrating on elaborating on individual items, the focus lies on giving a clear and straightforward definition of a variety of related patterns. A software tool has been created which is able to simulate the introduced patterns in different geographic environments, giving designers the opportunity to explore advantage and disadvantages of each pattern under different conditions.

The remaining paper is structured as followed: In section 2 the approach to systematic exploration of spatio-temporal game design patterns is given. This is followed by a list of the patterns found. As an example, one of the patterns is described in further detail. In section 3 the simulation framework, that serves as a basis for the simulation routines that implement identified game design patterns, is introduced. The practical benefits of this application are discussed and the visualisation output is portrayed. Finally, I conclude with a discussion of lessons learned and give an outlook on future research.

2 Pattern Exploration

The purpose of this library of location-based game design patterns differs from that of other collections. Its intention is not just to collect patterns that are applied in existing games, but to maybe even uncover, describe and name unknown and never before used patterns that could be implemented in future generations of location-based games. For this reason, the collection process was not researching a selection of existing games. Instead I chose a proven game model as starting point for the analysis, the player model of Heinz and Schlieder (2015). This player model abstracts from the details of specific location-based games by providing generic descriptions of the game elements needed for modelling a wide range of game mechanics.

The geogame model allows for a multitude of different relations between the game entities it defines, such as topological, Euclidean and also independencies between entity states. The possible outcomes are too numerous to be all covered in a first exploration/collection process. Therefore, further restrictions were imposed. Patterns should initially only refer to the relation between the two most important entities. These are:

Players: The most essential element of every game. In the case of location-based games, players move through their geographic environment while trying to win the game by executing given game actions. In this process players experience their environment. The knowledge about this environment is enriched by data displayed on devices – in most cases smartphones or tablets – running the game application, displaying a map that visualizes entities relevant for the player.

Places: A finite set of (immobile) areas of interest in the geographic environment. Game actions are often bound to corresponding places and a player can only execute them if he/she is located there.

Every game application has to contain an implementation of its relevant game mechanics and has to be able to decide over the outcome of this game. Hence, virtual representations of relevant game entities have to be stored. Location-based games also require the software to have a GIS component that handles the position and geometries of mentioned entities. In almost any location-based game, players are reduced to simple point objects. In the case of places, the respective GIS objects are chosen depending on game mechanic use cases. Sometimes it is sufficient to store the place as a point (**p**oint **o**f **i**nterest). If the game mechanic needs to calculate exactly whether the player is located inside a specific region, this place (**r**egion **o**f **i**nterest) has to be linked to a polygon geometry.

Game design patterns that will be explored in this work specifically refer to the virtual representations (PoI, RoI) of involved game entities. These representations are also used when the game decides on the outcome of a game action. Therefore, they play a central role in the balancing of game mechanics. Area of focus of the patterns will be the player. Differentiating between the geometry types for places and the specification that places cannot act on themselves, three couples of entities can be identified:

- Player – PoI
- Player – RoI
- Player – Player

The game patterns will describe relations between the entities of each couple that belonging to the perception of the player, that can also be applied to the geometries of the entities. Two fundamental relations provide the basis for the pattern exploration procedure:

Equals/Contains: Relates to the position of the geometries and indicates whether a player is located directly at a place or shares his/her location with another player. Small differences in the position can be neglected depending on the respective application. This subsection of the region connection calculus (RCC) (Randell et al., 1992) was chosen because it plays the most important role in games, already implemented on top of the geogame model. Other RCC relations will be evaluated in future works.

Line of Sight (LoS): Indicates whether a line of sight between the player and other entity exists. The line of sight may be blocked by the environment the player is moving through. Any kind of entity state changes are ignored.

2.1 Spatio-temporal Pattern Listing

For a first pattern listing, each pattern will be defined by a change of the connection between an entities couple. Both relations can be observed separately or in interpendence with each other by building the Cartesian product of each relation and its negation. In the latter case not all combinations are possible because a LoS is automatically in existence if the equals/contains link is evaluated to be true. In table 1 all of the 45 different relations between the entity couples are listed. Each entry represents a game design pattern and was given a name. Some of the pairing's patterns offer a lot of similarities. This is also shows in the pattern names, which in some cases read exactly the same because identical game mechanics are applied to different kind of game entities.

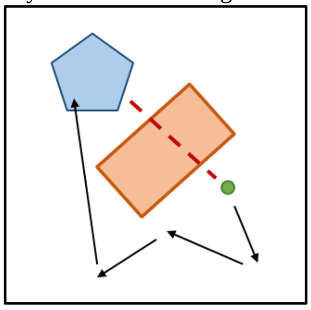
Table 1: Listing of selected spatio-temporal game design patterns

Precondition	**Postcondition**	**Player - POI**	**Player - RoI**	**Player - Player**
¬EQ	EQ	Go to	Enter	Meet
EQ	¬ EQ	Leave	Leave	Separate
¬ EQ	¬ EQ	Stay away	Avoid	Stay separate
EQ	EQ	Stay at	Stay inside	Stay together
¬LoS	LoS	Get in sight	Get in sight	Get in sight
LoS	¬ LoS	Get out of sight	Get out of sight	Get out of sight
¬EQ, ¬LoS	EQ	Search and go to	Search and enter	Search and meet
¬EQ, LoS	EQ	Go to	Enter	Meet
EQ	¬EQ, ¬LoS	Leave and get out of sight	Leave and get out of sight	Separate and hide
EQ	¬EQ, LoS	Leave and stay in sight	Leave and stay in sight	Separate and stay in sight
¬EQ, ¬LoS	¬EQ, LoS	Look for	Look for	Look for
¬EQ, LoS	¬EQ, ¬LoS	Get out of sight	Get out of sight	Hide from
¬EQ, ¬LoS	¬EQ, ¬LoS	Stay out of sight	Stay out of sight	Stay out of sight
¬EQ, LoS	¬EQ, LoS	Stay in sight	Stay in sight	Stay in sight

2.2 Example Pattern Description

The following short description for one of the listed patterns will serve as an example for an entry as it will be found in the pattern library. Whenever possible, pattern names as well as pattern descriptions will be given from the perspective of the player.

Figure 1: Symbolization of the game mechanic



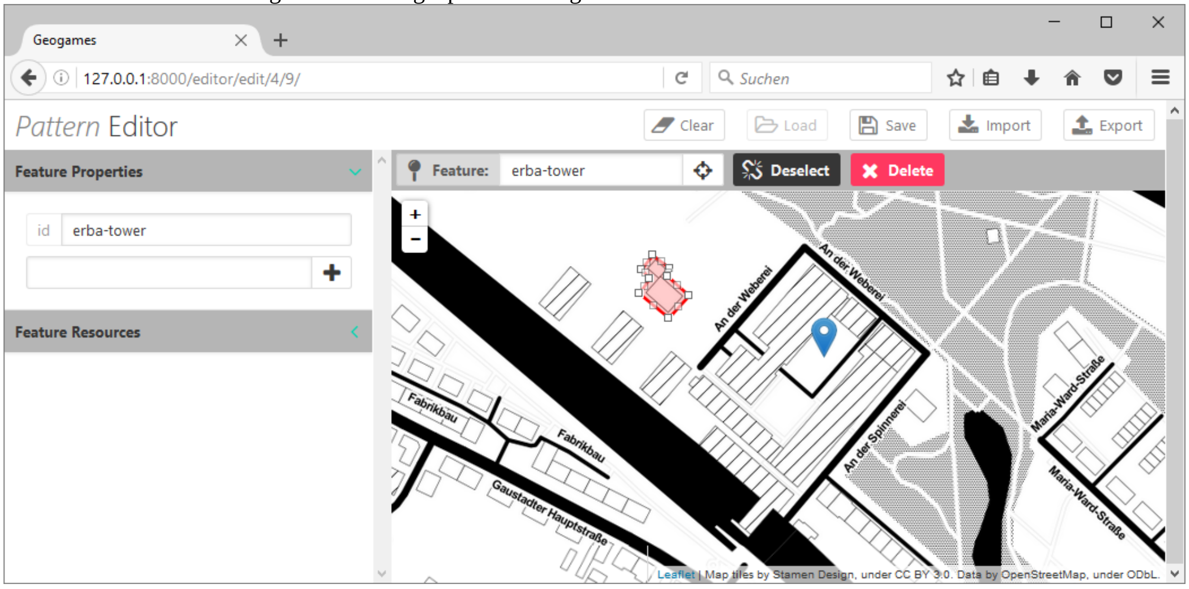
Pattern title: *Search and enter*
Preconditions: ¬EQ, ¬LoS
Postconditions: EQ
Explanation: The players goal is to enter the RoI geometry. At the beginning, the player has no line of sight to the RoI. The player may, however, have knowledge about the location of the RoI via textual description or any kind visualization of the RoI. The player moves through his/her geographic environment in search for it. The pattern is terminated once the player has entered the geometry defined by the RoI.
Sibling patterns: *Search and go to, Search and meet*

3 Simulations Routines

Technical details and a detailed walkthrough of the simulation framework created especially for the exploration of said patterns will be omitted. These will be explored in a follow-up work. Nevertheless, a short description will be given to explain the advantages over established agent-based simulation frameworks.

The simulation framework is based on a heavily modified version of the agent-based simulation framework MESA[1] (Masad/Kazil 2015). One of the most interesting features of this software is the ability to create HTML based visualizations that are updated and can be made interactive via websocket-protocol. It is intended to embed said visualizations into local, HTML based notebooks. These notebooks are being used as "interactive computational environment, in which you can combine code execution, rich text, mathematics, plots and rich media"[2]. By switching the "visualization server" component of the framework it was modified in such a way that simulation visualizations can be served over the internet. This enables users to use the software through a browser on any device connected to the internet and removes the need to install additional software. The simulation server was embedded into a Django-Channels[3] application. This enables not only the real-time communication between clients and the server but also makes it possible to integrate user and rights management to the application, that can be applied to the execution of simulation runs. Registered users are able to create their own configurations for each pattern routine listed in section 3. Figure 2 shows a simple setup for the *search and enter* pattern made, using the editor. Users of the editor can draw and edit vectors and geometries on a map. By doing this, they are creating game entities. Created editor configurations are saved into a database and can be edited later as well as shared with other users via a hyperlink. To make location-based simulation possible, MESA was extended by a GIS component. This

Figure 2: Creating a pattern configuration with the browser based editor



127.0.0.1:8000/editor/edit/4/9/

Pattern Editor

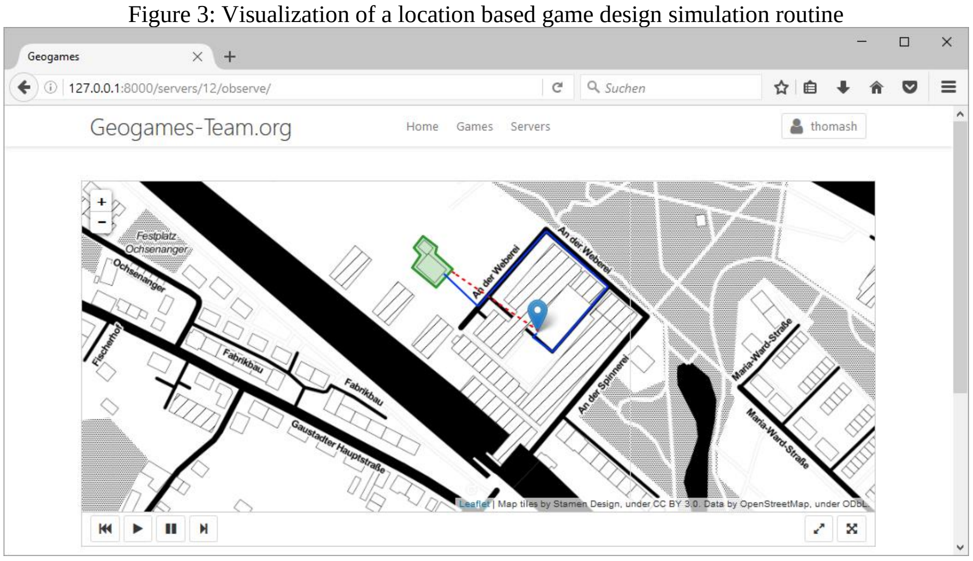
Feature: erba-tower

id: erba-tower

Leaflet | Map tiles by Stamen Design, under CC BY 3.0. Data by OpenStreetMap, under ODbL.

[1] https://pypi.python.org/pypi/Mesa/
[2] https://ipython.org/notebook.html
[3] https://github.com/django/channels

Figure 3: Visualization of a location based game design simulation routine



component uses the well-known python libraries "shapely"[4] and "pyproject"[5]. To enable simulations with realistic player locomotion the routing functionality presented in Heinz and Schlieder 2015 was recreated. A browser based visualization of a pattern simulation is shown in figure 3. To create map based visualizations, the software library "leaflet"[6] was used. Map tiles originate from the OpenStreetMap project. Both, the editor, as well as the simulation visualization web application are also able to run on mobile devices (e.g. smartphone, tablet). As a consequence, simulation configurations and simulation runs can be created/run on-site if desired. Depending on the kind of pattern, each game entity, placed in a configuration, is associated with a specific, built-in agent-behavior. Agents act in such a way that they try to transform a patterns state from the given preconditions to the specified postconditions. In doing so they try to match human player behavior in the same way as it is described in Heinz and Schlieder 2015.

4 Discussion and Outlook

This paper presents a small selection of location-based game design patterns. In the future the library of game design patterns is going to be extended. Moreover, it will be easily applicable to wide variety of game design problems due to its usage of an abstract game model. Usage and implementation of the described patterns is facilitated by their formal description.

The simulation of these patterns via a web based simulation framework already works flawlessly. Being able to set up self-chosen configurations in all kinds of geographic environments, without the need for any kind of special hardware seems like a promising approach for designers of location-based games and applications. With this kind of simulation framework – usable on any portable device – users are able to create and simulate pattern configurations for site specific game mechanics, while being on-site and making themselves familiar with the environment. Users can easily share simulations with others via a hyperlink. This will further improve communication about problems or benefits of applying game design patterns to certain conditions.

A next topic to explore are the combinations and interactions between different patterns. Computational tools will be added on top of the existing simulation routines. This will help designers with the task of balancing game mechanics for specific spatial environments.

References

Adams, E. and Dormans, J. (2012). Game mechanics: advanced game design. New Riders.

Davidsson, O., Peitz, J. and Björk, S. (2004). Game design patterns for mobile games. Project report to Nokia Research Center.

Dormans, J. (2013). Making design patterns work. In Proceedings of the Second Workshop on Design Patterns in Games, DPG '13.

Heinz, T. and Schlieder, C. (2015). An Agent-Based Simulation Framework for Location-Based Games.

Jacob, N. and Coelho, A. F. (2011). Issues in the development of location-based games. International Journal of Computer Games Technology

Masad, D. and Kazil, J. (2015). Mesa: An Agent-Based Modeling Framework. 14th PYTHON in Science Conference, 53-60.

Randell, D., Zhan C., and Anthony C. A spatial logic based on regions and connection. KR 92 (1992): 165-176.

Schlieder, C., Kiefer, P. and Matyas, S. (2005). Geogames: A conceptual framework and tool for the design of location-based games from classic board games. Intelligent Technologies for Interactive Entertainment, 164-173.

Will, C. (2013). A Pattern Language for Designing Location-based Games. Diploma Thesis, RWTH Aachen University.

[4] http://toblerity.org/shapely/project.html
[5] https://jswhit.github.io/pyproj/
[6] http://leafletjs.com/