In-Game Market Systems and Their Agents

Melanie Freilinger

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im September 2014

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

Hagenberg, September 29, 2014

Melanie Freilinger

Contents

Kurzfassung

Diese Masterarbeit beschäftigt sich mit dem Thema *In-Game Marktsysteme*. In den letzten Jahren, kamen immer mehr Spiele auf den Markt, welche In-Game Marktsysteme benutzen. Es gibt viele verschiedene Arten von Systemen welche in diesen Spielen verwendet werden, was abhängig von dem Spielgenre ist. Im ersten Teil der Arbeit werden diese verschiedenen Systeme genauer beleuchtet und Beispiele gezeigt in welchen Spielen sie zu finden sind. Weiters wichtig sind ebenfalls die Hintergründe für eine Implementierung eines In-Game Markts in ein Spiel und Strategien, wie dessen Stabilität gehalten und gewährleistet werden kann. Der erste Abschnitt vermittelt, wieso es heutzutage ein wichtiges Thema in der Spielentwicklung ist. Die Arbeit blickt auch kurz auf Ökonomische Grundlagen, um das Wissen für wirschaftliche Systeme aufzubessern, da diese gebraucht werden um die Funktionsweisen verstehen zu können. Weil die Marktsysteme in Spielen denen in der realen Welt sehr ähnlich sind und gleiche Funktionsweisen und Verhalten aufweisen, kann die übliche Wirtschaftstheorie angewendet werden. In der Implementierung, zugehörig zu dieser Arbeit, geht es im wesentlichen um die Simulation eines virtuellen Markts, in dessen Hintergrund intelligente Agents stehen, die Spieler simulieren und die Wirtschaft im Spiel dynamisch halten sollen. Dazu werden auch essentielle Grundlagen für das Design von intelligenten Agents präsentiert. Im Implementierungsteil wird näher auf die reale Ausarbeitung eingegangen und das vorher gesammelte Wissen, praktisch angewendet. Das dabei entstandene Tool ist für jedes Spiel verwendbar und besteht aus zwei Teilen: dem Userinterface des Marktes, welches die Spieler verwenden können um mit dem Markt zu interagieren, und den Agents im Hintergrund. Am Ende der Arbeit wird das Tool getestet und evaluiert, ob das Verhalten des Marktes und das der Agents dem entspricht, was erwartet wird und wie diese Ergebnisse eventuell angepasst und verbessert werden können.

Abstract

This thesis deals with the topic of *in-game market systems*. In the last decade more and more games came up with the idea to include market systems into their games. The range of markets are big and the forms of it can differ from one game to another. The form of the market depends on the game itself. In the first section some of the diverse systems are figured out and in which games they can be found. Also some backgrounds why markets are introduced into games and how to hold the stability of it, are presented. After getting some impressions why this topic is an important part of today's games, the thesis gives some basic knowledge about economical basics which are somehow needed to understand how these market works. Because the real world market and game markets have very much in common, traditional economic theory can be used. One of the main issues of the implementation part, is to simulate a virtual market for a specific game. And because this market can not work with only few participants, the design of agents with economic behaviour is also covered in this thesis. At the end of the thesis, the gathered knowledge about economics, simulated virtual market and intelligent agents is summed up to create a small in-game market system which can be implemented into every game. The tool is split into two parts: the user-interface part, which is planned for the human players and in the background there are working simple agents using theories described earlier in the thesis. At the close of the thesis, these agents are evaluated and tested, if the planned behaviour acts like expected and what can be done to adjust and improve these working routines.

Chapter 1

Introduction

1.1 Goal of the Thesis

Gaming is getting more and more important in our daily lives. Every year there are thousands of participants at conferences like the *Gamescom* at Cologne[1]. Games are going social, target groups are spreading and the games industry is an multimillion dollar business. Because of these big markets, topics around games are increasingly relevant.
An interesting topic in game development, is the production of an in-game market system and economy. It became very popular to integrate these into games. Furthermore the ideas of creating virtual markets is an much treated topic, not only in games. There is great interest in finance in the simulation of markets, for example for research issues. Financial theorists discovered, that the virtual environment has perfect conditions to research new ideas of economic theory. The ideas used in traditional simulation of markets can be used in games as well. Because of these facts, the topic of the integration of such systems is worth to take a deeper look. In this thesis the topic pool of virtual markets and simulating these, is elaborated. The theoretical part is supported with a real implementation of an virtual market system for a game, which uses some of the presented ideas including financial theory, artificial intelligence and knowledge about virtual market simulations.

1.2 The Beginnings

Asheron's Call, Ultima Online and Everquest were one of the first games which involved virtual economy. In Asheron's Call it was very risky and cumbersome to posses in-game money. It had no market system and players were not able to trade secure because there was no feature for this. Additionally the most valuable item in the game were ubiquitous. Furthermore there was little motivation to gather much money. There was no crafting

[1] http://www.gamescom.de/de/gamescom/presse/presseinformationen

system, the loot were most of the times only trash, and when players found a good item, the did not want to trade it for money, so for what else? Everyone wanted and needed the ubiquitous shards and moulds, because for every good weapon and armour these items were indispensable. Because of that fact, it developed a barter system and players listed on forums what they wanted to trade. The 'currency' was the shards and moulds. So even despite the lack of trading tools like today common auction houses or bazaars, there was much trading in the game [13].

1.3 Virtual Economy

Because of the growing popularity of in-game markets and economies, game developers often hire economic experts for these issues so that they can ensure that the economy remains stable and efficient, what then helps the player to enjoy and like the game. The average player spends around 22 hours per week in-game and spends about 200$ per year in monthly subscriptions, so the players call for a stable game with all its features [24]. Today virtual economies are found in many different games. The larger virtual economies are often found in MMORPGs. But also inherent in life simulation games, browser games and mobile games. In the popular life simulation "Second Life" the virtual economy is tightly bound to the real world. Players can create in-game assets and sell them to the other players of the game and also earn real world money. The following characteristics may be found at virtual goods in games. The characteristics are very flexible and rely on the specific game design [7].

- **Rivalry** The resource is limited to one person or a small amount of players.
- **Persistence** The resource is persistent across the users sessions. They are even sometimes visible when the user is logged out.
- **Interconnectivity** The resource effects and is effected by other players in the game world. The value of the resource depends on the persons ability to use it.
- **Secondary markets** Virtual resources are created and traded sometimes also for real world money.
- **Value added by user** The user is able to enhance the value of a resource by customizing it.

As the conditions are similar to the real world, economic theory can be used to study these virtual worlds. In-game items are often priced according to supply and demand of the in-game economy. The virtual economies are a common core feature in the games. What the real world economy and the virtual ones distinguishes, are following facts:

- every player has the same intrinsic capability to get into the economic activities, like harvesting, crafting and trading,
- crafting activities are often not bound to a fixed place and can be done anywhere in the world or at explicit crafting stations, at any time,
- all products are homogeneous, means that every heal potion is always the same and has no quality issue, no matter who is crafting it,
- the production costs are always the same, not matter how many items the player makes.

The key issue is how a game designs the market around the concept of *Perfect Information*. Perfect Information labels the theoretical idea that all producers and customers know price and quality of goods. In games that would be the information shown to the players. In many games that happens by hovering over an item and players are able to get all relevant information. Players can have all important information to make a decision [13]. With the ability of having all the necessary information about data and the ability to track individual actions, virtual economies can be a new way for economists and social experts to research real world problems of inequality, poverty and unemployment at a model like a game [24].

1.4 The Price of Time

Players not necessarily do want to maximize their profit (depends on the game design), rather than playing to have fun. Some players will rather chose the more funny way of receiving items. But also sometimes the more quick way is chosen, which is often to simply buy the item than farming itself. It creates the twist between time and playing for fun. Fundamentally the value of an item, its costs to obtain it, means the time which the player have to invest. Players often extrapolate, in order to find out which actions will show results earlier. This behaviour leads to rough irregularities. If many players enjoy farming more than buying them, the market can get depressed if there are to less players to buy these superfluous items. So deflation does not come from an overcrowding in supply of the resources rather from too many suppliers [13].

1.5 Mixing Real and Virtual Money

Today in many games it is very common to create a mixture of using real and virtual currency. The items and goods are typically changed for in-game currency and often these currencies are also related to real world currency. This system is often used in free mobile and Facebook games like *Farmville*, *Candy Crush Saga*, etc. where this is called *In-App Purchases*. Also in other games it is common to use a in-game currency related to real money. In

League of Legends the players have a currency which is directly bound to real money. With these points users can buy *skins* for their heroes, which only have a character of make-up. There are also many MMOs, mainly ones without monthly subscriptions, which have in-game currency paid with real money.

1.6 Games using In-Game Markets

1.6.1 MMO-RPGS

Eve Online

The popular MMO *Eve-Online* is known for its stable and complex economy. In Eve-Online you even loss more money when not attending in money gathering activities. The two main activities in the game are gathering ISK, which is the currency, and PVP. When participating on PVP the players can win great loot and invest it. Players can trade raw materials, participate in complex markets and create powerful trade alliances and financial institutions like banks [24]. For the game the developer even has its own economy specialists to help to keep the game balanced, one of them is Eyjolfur Gudmundsson [23]. The first economical breakdown in the game was caused by a group of players which manipulated the game's economy to their favour. Eyjolfur told the *Wall Street Journal* that: "there is nothing virtual about this world".

Guild Wars 2

Guild Wars 2 has a combined system using in-game and real world currency. Diamonds can be bought with real money and spent in the in-game market. Players can either spent the diamonds on items like skins for their armour and weapons, or gimmicks like pets, but none of these items bring an advantage for the game itself, they only have cosmetic character. Furthermore the player is able to convert the diamonds into in-game gold which can be used by the game character to buy all the other items available in the game. The rate of exchange between diamonds and gold can change over time and does not stay the same, depending on supply and demand of diamonds. The more people are buying diamonds the less value the have and otherwise. So clever players will use these informations to decide when it is the best point to buy or change diamonds.

X-Rebirth

In the online space game X-Rebirth there is also a big economy. They have agents which are do trading as real players do. In this system the prices are dependent on the simulated economy that means the developers designed the

games' market like a realistic economy. There would also be the possibility to design it player friendly, like consider the players wishes and needs. But they decided not to choose the top-down way although this could be more satisfying for short periods. For balancing the economy and to have an idea how to design the market system, they use a time accelerated mode to simulate many weeks of game play in a couple of hours. With that it is possible to see effects of influences on the market also for the future sight. Effects can for example occur on prices and also on storage levels. With these parameters it is possible to define and evaluate if the economy is robust or weak. Small bugs in the artificial intelligence (AI) can cause big damage onto the economy even destroy it. So in this testing phases developers can spot possible bugs or errors in the economy or in other aspects of the game which have influences. For balancing the in-game economy it is necessary to observe it. Means that the behaviour of it has to be analysed and compared to a list of wishes how it should behave. With this data then behaviours can be tweaked. A furthermore important aspect, is to design the economy interesting for players. They should see a good combination of perfectly working or partly damaged economy, like in real life. It would be very boring and players will not use the market system any more, if everything is straight ahead and every time perfectly balanced. So also factories can be destroyed so the delivery of many resources is stopped, but these resources are important for the market system. So every game design decision will have an impact onto the economy and can harm or improve it [11].

1.6.2 Example of Failure

The example of Diablo 3's closed market system, shows that game designers should not implement market systems into games where they do not belong to or fit into. At the launch of Diablo 3 an in-game market was implemented. It was an auction house similar to Blizzard's popular World of Warcraft, with one difference: player could make real money in selling Diablo items there. One of the problems was, that the developers did not expect that the players will sell nearly every item at the AH, not only epic ones. The players did so, because the did not have any reason to do not. Another and main problem why the market did not worked as expected was the fact, that people did not want a market like that, because it does not fit into the gameplay. Items are meant to be dropped by enemies in the game and not to be bought with real money or gold. That design of the game and drop chances led to, that players rarely got good items and had to use the AH to get good ones. In June 2014, Blizzard removed the AH completely out of the game and did this to the joy of most of the players [27].

1.7 Broken Models and Virtual Worlds

In financial environments, there is no measurement of doing it right or wrong. The traditional models of economic solutions are broken, what is proved by the economical crash in 2008. These crashes coming back from time to time, are caused by the same thoughts which are used over and over again, with no new solutions. Edward Castronova said:“Complex economic models, rely too much on human rationality and worse, they do a terrible job predicting events.” Virtual worlds give new possibilities for researchers about assumptions of human decision making. The problem of ideas tested in laboratory environment, often lack on validity when transformed to the real world. Travis Ross, a PhD candidate at Indiana University Bloomington has the opinion, that the study and research of decision-making in games can help us, to understand what motivates people and their behaviour. Games can help to build a theoretical bridge between the individual motivation and the collective outcome of complex economies [24]. Using games as test environment brings the advantage of being able to use trial and error and also to provide innovative new solutions. Attention has to be paid, because not every game with an economy is suitable for real research. It is important to keep one fact in mind that games are intended to make fun and often problems like in the real world does not exists in-game. In most games it does not matter in what profession a player is engaged, the game design takes care to treat everyone equally.

1.8 Conclusion

The first ecosystems in games were simple and first not intended by the developers. The community of players created their own trading markets, often including a barter system. The game developers recognized, that a market system can bring new value to the game and started to implement real market systems and balanced game economy. Quickly it became popular to integrate these in-game economy. But care must be taken, because not every game is suitable for the usage of these markets. It has to fit in the overall game design. Virtual worlds are a suitable environment to test new ideas and approaches in economic research.

Chapter 2

Related Work

2.1 Computational Intelligence

Computational intelligence (CI) techniques are important for finding strategies and solutions to behaviour in economical games. The article from the *IEEE Computational Intelligence Magazine* [10], written by Dawid, Poutré and Yao, shows an big overview about the relevant topics concerning CI. The following section presents ideas and solutions out of this article.

Economic games can for example be games with basis in game theory or games which using market models like microeconomics. CI is needed where analytical or mathematical approaches end. Suitable techniques are *evolutionary algorithms* (EA), *neural networks* (NN), *reinforcement learning* (RL) and *fuzzy systems* (FS).

2.1.1 CI in Game Simulations

It has to be distinguished between the development of individual players (agents), and the study of behaviour of players together (market behaviour). For both situations there are according CI techniques which are capable of learning behaviour and can be used for these issues. This can be for example done with EAs oder NNs. Explicit bids can be represented with chromosomes, specific actions done in certain states of the environment can be realized with RLs, EAs, NNs or FSs. Either the agents learn on their own, or they learn as a society (social learning). There often EAs are used. For researches with CI it should be considered to use repeated games, because they allow the usage of learning strategies. For the creation of learning strategies, self-play and co-evolutionary strategies are used. When dealing with learning systems in games, it is essential to develop them also for N-player games, because assumptions made in 2-player games, hardly fit into N-player games conclusions. Co-evolutionary algorithms are also suitable for games with multiple choice possibilities. They are important when modelling realistic environments.

2.1.2 CI in Simulated Markets

CI techniques also can be used to simulate economic markets. When running the simulation of such systems, the main problem is the robustness. The area which is handling with these issues is called *agent-based computational economies* (ACE) and widely used in finance.

2.2 TAC

The trading agent competition (TAC) gives researchers the possibility to compete with others with their agents. The goal of the competitions are, that their agents have to compete with other agents to a specific topic. For example, the agent should travel from A to B. It have to get a well selected portfolio of hotel bookings, flight tickets and event tickets, but there are many other problems to be solved at the competitions. An similar competition is done every year considering the electric labour market[1]. The TAC was introduced 2000 by Wellman and Wurman [18] and since then, the international competition is annually repeated. Participants have to build an automated trading agent, which acts as an pc assembler and have to compete with other agents for customer orders. The game simulates the typical three-level supply chain in an pc product market: component suppliers, assemblers, and end customer [21]. The main issue in these events is to master the challenges in supply chain management (SCM). The trading agent design have to be capable of well defined decision-making sequences of daily production and product pricing. The TAC considers price (demand) and quantity competition (supply) [21].

2.3 Intelligent Agents in Finance

Capital projects with very large investments (giga-investments) have a very long economic life cycle, often up to sixty years. There are many unknown and unpredictable risks and potentials and often hard to plan and predict at start of the project. Even the requirements of these investments can change over the whole lifetime. That can be the market for the end product or technical improvements [4]. Managers need support of decision tools for these projects, to be able to act in an appropriate way, choose the nearest optimal way with most profit. In this highly dynamic environment, these tools need constant access, to new information which could affect the project, to the real time situation of the project itself and easy access to advanced analytical tools [5]. A common solution for such decision tools are the usage of fuzzy logic. The statement: “The project will produce a cash flow between fifty and sixty, in two years from now”, is an typical fuzzy statement from

[1] http://www.powertac.org/

financial managers. It includes his intuition about the project and if this manager is the best expert around, then this statement is the best available estimate of the future outcome. With fuzzy systems it is possible to use such statements and estimates without bringing them into a single value. The estimates are understandable by the managers and if included into the actual estimation and profitability calculation, there is no loss of information. Another advantage is, that the fuzzy sets can be dynamically adjusted to reflect future trends and can give further insight into the real uncertainty of large investments [5].

Chapter 3

Economic Basics

3.1 Overview

Before starting to create an economic system for a game, it is required to learn simple economic basics. In the following chapter some important and basic terms are described. Some of them also find their way into the final implementation.

3.2 Term: Economy

In modern times the term *Economy* means the sum of all actions that serves the supply of people in a society. The economic system consists of production, distribution, trading and consumption of goods and services by different agents in various geographical locations. These agents can be individuals, businesses or governments. Transactions occur when two parties agree on a certain price for a specific good or service, mostly expressed in a specific currency.

The concept of demand and supply usually occurs in the *market-based economy.* Goods and services are exchanged between the participants according to demand and supply. The transactions are done by barter or other mediums of exchange with a credit or debit value. In contrary to that there is also the *command-based market* where decision of production and distribution are handled by political parties.

3.3 Real World Stock Market

A stock market in a simple view is a organised market where buyers and sellers can trade their items. But not real items but rather so called stocks. The market is controlled by supply and demand and the market price regulates the differences between that. A stockbroker is a person who sells and buys the items for his clients. So the clients do not trade their goods themselves.

Vol Bid	Bid	Ask	Vol Ask
2	240.5	245.0	1
5	120.9	121.0	10
23	528.0	529.0	30
15	234.54	235.7	13
25	159.0	157.8	25

Figure 3.1: Orderbook

Today it is not common any more to be locally there to trade, all the trading are transacted over the computer and also the stock paper is only virtual. To give clients an overview how a specific stock developed over time, there exists the stock index. This index predicates the development of price to amount in a certain time-slot. With this index it is easy for the clients to see if the stock is more rising or falling over a certain time. For ordering stocks there exists a so called order book. The client can fix a maximum price for buying a stock, it is called *Bid* and he can also fix the minimum price for selling a stock, called *Ask* and the difference between both is called *spread*. This value has to be between Bid and Ask and is then the final stock price. Then for the final sale the stock broker decides to which course the stock is sold. It depends on how much profit he can make. He will take the price which will result into the most profit what means that he will decide on how many orders he can execute [16].

3.3.1 Orderbook

In former times, investing into stock primarily was done by going to a bank or engage a stock broker to trade. In modern times everything is done electronically. It is common to use limit order books. This is a system, in which all items to buy and to sell are displayed, best priced at the top of the list. Figure 3.1 shows an example of such an order book. In the columns named *Bid* the client fixes a price and amount for buying a stock. On the right hand side in the columns named *Ask* the best offers for this share are displayed. The numbers of sizes for a stock are cumulative what means when an order for bid is executed to a price of 240.5, the item will reside in the order book until the 2 bid offers are executed. A buying order will be executed instantly with a price of 245.

3.3.2 Influences on Stock Market

Trading in a stock market is a psychological topic because the behaviour of traders is influenced by expectations. These expectations can be the hope

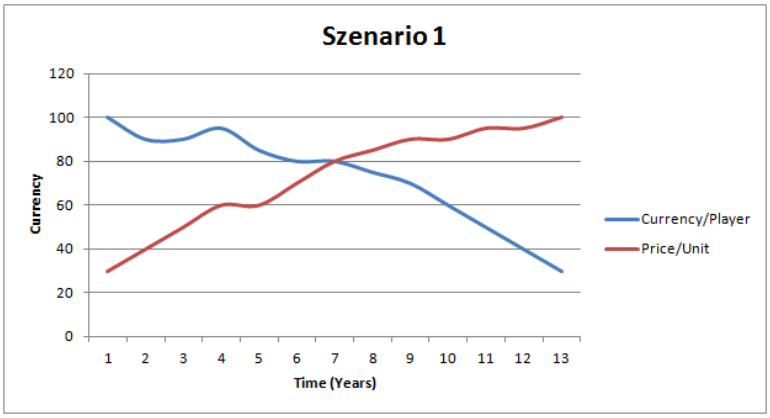


Figure 3.2: Inflation

of a better future development, or be influenced by new business numbers either good or bad, the growth of the economy itself and also prices of materials can influence prices at the stock [16].

Inflation and Deflation

In economic theory *inflation* means the reduction of the value of an currency, see figure 3.2. The general price level of goods and services increase over a period of time and with that it comes to a reduction in purchasing power per unit of money. A measure of the price inflation is the inflation rate which is calculated annually and shows the percentage of change in the general price index [29].

Deflation describes the decrease of the general price level of goods and services over time, see figure 3.3, means that individuals can buy more goods with the same amount of money. Deflation occurs when the inflation rate falls below 0%. Deflation should not be confused with *Disinflation* which is a slow-down in the inflation rate [22].

3.4 Conclusion

Designing a realistic simulation of an trading system is a difficult task, because of the unpredictability of the impacts which influences market behaviour. Some of these effects can also occur in the virtual simulation of a game, sometimes this behaviour is allowed and required but also often need to be prevented.

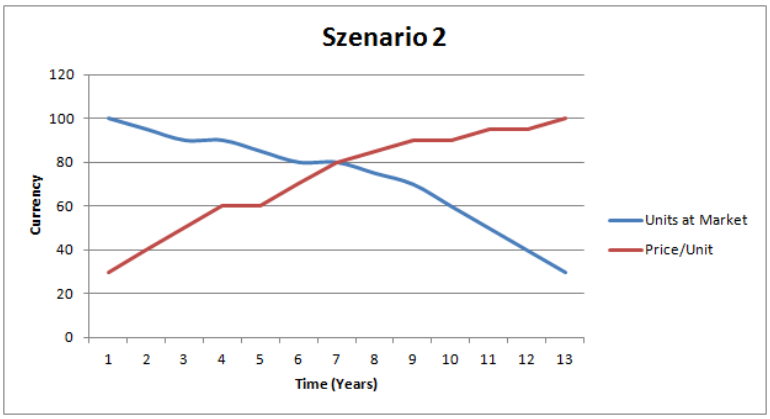


Figure 3.3: Deflation

Chapter 4

Dynamic Ingame Market System

4.1 Overview

After retaining knowledge about real world economies, the focus in this chapter, is on the game theoretic part. Real world economies and in-game economies can differ but can be compared in general. The main difference is, that virtual markets do have perfect conditions for every participant, because every player is seen equal in the game world.
An overview about game markets in general is given. Thereafter possibilities of creating an in-game economy are discussed and some of the results are then used in the according implementation.

4.2 Secondary Market Place

Since the release of Blizzard's MMORPG *World of Warcraft* (WoW) in 2004, many games with this new market system followed. It made the secondary markets mainstream. It is very popular to sell and buy WoW accounts for real money on the internet. Users can buy accounts with specific characters on it. The benefit for buyers are, that they do not have to invest the time into playing to get that far[1]. The main problem in games like WoW, Guild Wars, RuneScape, Lord of the Rings Online etc. is, that the developers did not intended to give players the possibility to do this. So users who invest real money into buying accounts or in-game currency, are risking to be banned from the game. Paying real money for virtual currency and markets have become a huge multi-billion dollar industry. Also *power-leveling services* are very popular. These services are called *Third party online gaming services.* The gold is often earned in sweatshop-like gold farms in developing countries

[1] http://www.thegamesupply.net/

like China and India. People doing this jobs spending days in the games to get the virtual value for very low wages. Their in-game gold is then sold for real money to other players, often Western players. These virtual goods are only some lines of code but gained a high reputation an luxury aura like expensive cars, watches, and other material welfare [23, 12].

4.3 More controlled Markets

There are also more controlled markets in games, where in-game currency is only available from the vendor itself, typically for cash. This is a popular model for freemium games. In such games it is very common, that players can buy themselves benefits for the game. That can be items which reduce the amount of time needed to archive a specific goal. Reaching goals or states in these games can often be very time-consuming and upsetting, because the game design is also intended to do so, to make the players buy advantages for real world money [13].

4.4 Stability in In-Game Markets

For games with an huge in-game market systems, it is an issue to guarantee a stable economy. A good balance between currency, resources and sinks must be found. EVE Online runs on a system based on *sinks* and *faucets*. The sinks causes that the money is flowing out of the game such as paying taxes and buying goods and services from Non-player-characters (NPCs). Faucets then bring back money into the game. This faucets can be bounty prizes, bonuses or even insurance payments. This results in an balanced economy responding to players activities [23].

4.4.1 Cashflow

To keep the economy balanced the game design should have a look at the *MIMO* concept. It means “Money in, Money out” and denotes a principle where all the money which gets into the game, has to have a way to be spent again. The basic rule and key equation, mentioned by D.Hart [25], of handling currency in-game is

$$Buy + Earn = Spend. \tag{4.1}$$

Taking World of Warcraft as an example, daily quests produce money regularly and buying an epic armour reduces it again. This balance between gaining and spending is highly important for in-game economies. Inflation is not caused by the money in pockets of the players but rather the balance of MIMO does [13].

4.4.2 Creation of a Game Economy

In the creation of an economic system in a game, there are no fixed rules or theories how to do this. It is necessary that the market system fits into the game, means it should not undermine the character of the game. As an game developer you have the advantage to do trial and error on your own game economy, if one part of the system does not work as expected, you try to find another solution to fix it. It is also a fact that traditional economics do not even work perfectly in the real world. But despite these facts, there are many tips on how to get started or what should basically be considered. The basic rule is that also the market system fits into the core loops of the game. A big problem in designing a economy in games is, that there is a infinite supply of resources and materials so a game designer has to think about on how players can spend their money and endless supply. To have a look at the MIMO system it is essential to create important, desirable items for the players. This items should most of the time be temporarily, means that the player has to buy it more than once in the game history. Prices could be adjusted dynamically regarding the market situation or the richness of players. It is also valuable to adjust the prices over the lifetime of your game. In World of Warcraft the *epic mounts* in the early times where very precious and expensive, it even was not so easy to gather gold. But in newer add-ons it were made much easier to collect money and with that also mounts are more easier to get. So Blizzards approach was, to implement other kind of mounts that could not be bought with gold, instead players have to gain certain achievements or doing quests for it.

Apart from the money based approach there is also the possibility of using a more barter based system which is also easier to handle, because it is harder to manipulate the flow of items than the flow of money.

4.4.3 Methods to prevent Inflation

In games there can happen inflation and deflation, but it does not happen to the currency. One piece of gold will be always one piece of gold in the game world. With the same amount of gold, players can buy the same amount of items at the NPCs (Non-playable characters) at every time. The drops or rises concern the resources itself, it is the value of the items which changes. But apart from that the rules are nevertheless valid. As already mentioned, it is important for the game economy to create the right money sinks. Such sinks can be

- **Exchange fees:** At the vendor the player gets less money for items, but on the AH he has to pay a certain percent of fee for selling it.
- **Automatic money sinks:** They scale to the richness of the player.
- **Auctions intended from the game developers:** At this events, money goes completely out of the game and does not only change its

owner. Such an event could be for example a lottery where players can win rare items.

- **Death penalty:** If players die, their armour and other equipment gets damaged and need to be repaired. The costs could scale with the quality of equipment. For expensive armour, the repair costs are higher.

4.4.4 Bi-Transactional Markets

The concept of a bi-transactional market means that players do both, place sell and buy orders for any item. In most auction-house based games only placing sell orders are possible. An auction house can be somewhat compared to a *broker*. You present your item in the auction house and place the sell order. You do not have to wait to get the item sold, instead you will get a message when someone has bought your item, so the auction house is doing the work for you. Many games using auction houses where players only can place sell orders. It is only common that players can place a bid on items and hope nobody will bid more on it. In markets with only sell orders, the market is purely supply-driven. Buyers have no influence on the price of the item. The available actions are only buy or not buy. The prices in such markets are only supply-driven resulting in maybe unfair prices or even failing transactions because the market stagnates.
In *Eve-Online* it is possible to place both: sell and buy orders, working as described earlier in this thesis. Using this kind of system makes the traders to compete each other finding the best prices for selling and buying the items, but not too high or too low to make no profit. These markets are supply and demand-driven. Any user of the market can see the lowest and highest prices for items in the market. Most of the players do not want to compete in market and only want to get fair deals [13].

Market Order

The market order (see figure 4.1) system provides a guaranteed execution but the investor does not have any control over the price. The price is fixed by the economy which depends on supply and demand.

Limit Order

The limit order system (see figure 4.2) can give a guarantee that the items are sold to the desired price but not to be executed. It can happen that the investor has to wait for a long time to sell his items.

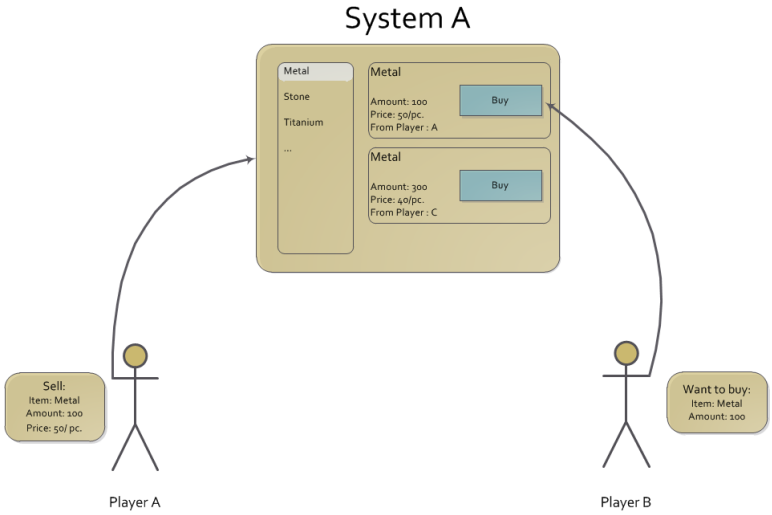


Figure 4.1: Market Order System

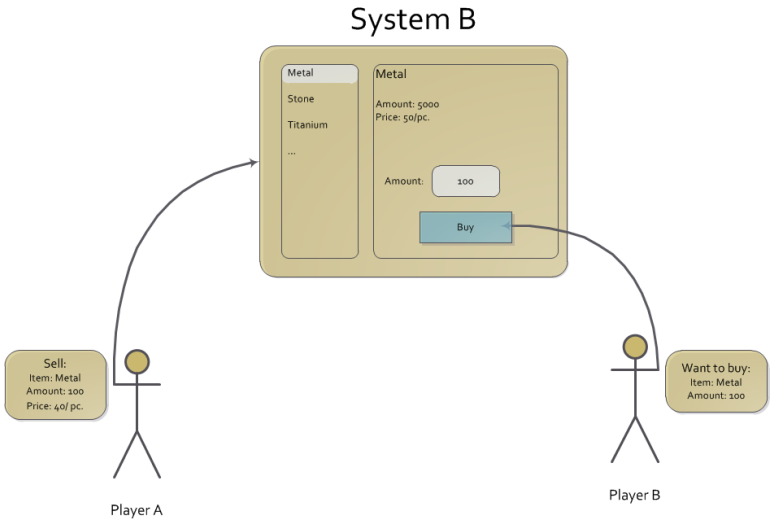


Figure 4.2: Limit Order System

4.5 Conclusion

In this chapter the main part was about gaining knowledge about how to take real economics and put it into a game to simulate an economy. It depends on the game and the game design how to design the economy then. There are many freedoms, but one thing always should kept in mind: get the money out of the game again.

Chapter 5

Simulating Virtual Markets

5.1 Overview

The next step in the creation of the game economy, is to occupy with the mechanics behind an economy simulation. What ideas are used for topics like this? During the research for the thesis, in nearly every paper about simulating economies, the topic *Fuzzy Logic* has been used. But there is a huge amount of methods and ideas how to accomplish this. Some of the ideas are basically described and Fuzzy Logic and Genetic algorithms are regarded closely.

5.2 A Game Theoretic Simulated Market

If there is the talk about economic games, games with basics in game theory with an economic background or models in micro-economics are meant. Examples for the first are negotiation, auction and the prisoner's dilemma [10]. The first thoughts which can be made about creating a simulated market, if there are learning capabilities by the agents, if the agents learn on their on experiences or if they use knowledge from the whole market behaviour. But for creating a real-life-like market model there are different challenges to handle. There are so many impacts that influences markets, like interest rates, the rate of economic growth and surely liquidity and also natural disasters. But these effects are non-linear, non-stationary and time-lagged [1]. Especially one group of computational models seems to be very appropriate: Natural Computing Algorithms but there are also other approaches for designing economical simulations. This chapter also goes deeper into the fuzzy logic system and will be described very detailed because the implementation (later in this thesis) is also using it.

5.3 Natural Computing

Natural Computing Algorithms receive their inspiration from the nature itself. The used phenomena exists in high-dimensional and dynamic environments which is also typical for financial markets [1]. Fitting methods according to Anthony Brabazon [1] for creating economic intelligence based on nature:

- **Neurocomputing** It takes it's inspiration from structure of the human brain and uses it in simplified models of it. Along with that the Artificial Neural Networks (ANN) also come under the Neurocomputing. ANNs are used for prediction, classification and clustering.
- **Evolutionary Computing** These methods underlie the Neo-Darwinian principles and is a population-based model in which the fitness of a individual solution is the measure of quality.
- **Social Computing** For Social Computing algorithms the swarm behaviour of birds, fish and ants are examined. It contains self-organization, flexibility, robustness and the communication between members, directly or indirectly.
- **Immunocomputing** As Neurocomputing the Immunocomputing concept takes it's inspiration from the biology. It' ideas are based on the biological immune-system and is able to recognize, destroy and remember thousands of foreign bodies. This models are ideal for classification and optimization problems. In financial issues it can be used for financial pattern recognition to identify possible fraudulent credit card transactions.
- **Physical Computing** Physical Computing uses physical methods like simulated annealing and quantum mechanics.
- **Developmental & Grammatical Computing** Algorithms based on Grammatical Computing uses concepts which are also found in linguistic grammars.

5.3.1 Evolutionary Computation

The history of evolutionary computation reaches back into the years of Alan Turing where he writes about the possibilities of evolutionary search. Evolution selects individuals depending on their relative success in surviving and reproducing [1]. Figure 5.2 shows the cycle of the process of evolutionary computation, how to obtain variations in every generation. Individuals can be represented in different ways, for example:

- as simple binary strings with fixed length,
- complex data graphs,
- computer-code with flexible length.

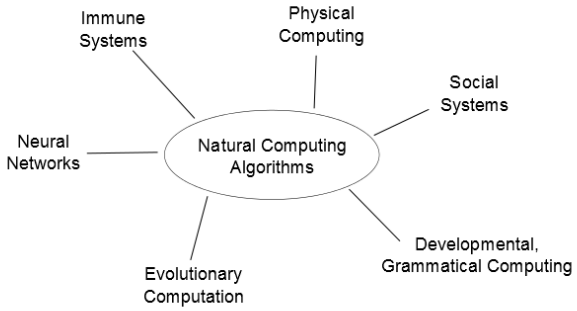


Figure 5.1: Natural Computing Algorithms [1]

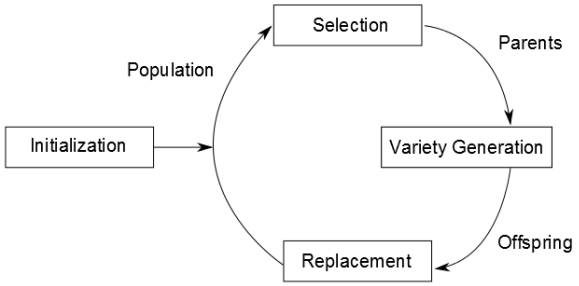


Figure 5.2: Cycle of evolutionary computation [1]

With simple binary strings it could be possible to decide, if the agent should take an variable or not, like a sell and buy action [1]. The choice of the fitness-function takes influence on the behaviour of the model. Genetic search operators are used to breed high-quality solutions. The selection of the best individual is fitness-based where fitness measures the quality of problem solving.

5.3.2 Genetic Programming

Using genetic programming (GP) allows it, that the model structure can evolve together with the parameters. Instead of using fixed-length strings GP uses variable length what brings the benefit that the model does not have to know the solution structure yet. This makes it a perfect candidate for the usage in finance theory, which is rich in data but have a lack of theory. The use of GP gives a better insight in the system and provides a more human readable pattern. GP also allows to use domain knowledge,

what means that persons with knowledge in finance and trading can use this, to influence the evolutionary process with his strategies and see the possible improvements afterwards. Also for optimization problems in complex finance theory, GP can be used [1].

Evolutionary algorithms have the benefit, that agents based on genetic approaches, have the ability of learning. It can include only themselves or even other agents or participants.

A genetic algorithm (GA) can be used for the simulation of a competitive system. The stock price model is non linear, stochastic and fully controlled by the agents. With the GA algorithm it is possible to promote the fitness of an individual and the survival of the fittest. The GA algorithm can be split into three modules:

- Population,
- Fitness evaluation,
- Reproduction.

Genetic Algorithm

Genetic algorithms are discrete optimization methods inspired by evolution and were invented for solving non-linear non-quadratic problems [9]. This evolutionary approach is based on the fitness of the individual (survival of the fittest) called genome. The three main modules of a GA are the following:

- **Population** The size of the population depends on the size of the search space because the bigger the search space the larger the population.
- **Fitness evaluation** Describes the individual which is evaluated every round.
- **Reproduction** Describes how the next generation of genomes will be. The outcome of individuals can be split into three parts:
  - elite individuals or its children go to the next generation without any changes,
  - mutations, some genes from the parent individuals are changed for the new generation,
  - cross-over, some genes from two parent individuals are combined for the new generation.

Each individual is represented through a collection of chromosomes and each gene position either takes the value 0 or 1. The initial population is chosen random. The search for the fittest is driven by the repetition of interactions between agents and other agents (artificial mating) and the environment (through fitness evaluation and selection). While iterating through the population samples the solution may converge to the most fit. There is no guarantee that the solution is correct, but the probability rises with each

iteration. For creating the most fit individual, biological operators are used to combine the best individuals which are going then to the next iteration. Following operators are used:

- **Selection** Probability of surviving is assigned to individuals, based on relative fitness. A high fitness value, leads to a high probability of selection.
- **Crossover** Can be seen as artificial mating. Two individuals with high fitness values may produce an offspring with a higher fitness value. A high fitness value leads to a high probability of mating.
- **Mutation** It represents innovation with random adjustments in the individuals genetic structure.

5.3.3 Fuzzy Logic

Another common approach in designing a simulation of virtual markets, is using Fuzzy Logic. Lotfi A. Zadeh published his theory of fuzzy logic in 1973. Generally in mathematics and logic, only binary values are used to describe statements. Such statements are called *crisp* values. But sometimes it is too hard, or to much effort to describe a system or situation mathematically. Fuzzy logic becomes handy when it is intended to work with imprecise, noisy, vague or missing information [26]. In binary models an element either belongs to a specific set or not. There is nothing in-between. Using Fuzzy Logic it is possible to design overlapping statements. There are basically three steps in the process of designing a fuzzy-logical system:

- **Fuzzification** Crisp input values are converted into fuzzy input sets.
- **Fuzzy Rules** Fuzzy inputs are processed with fuzzy rules to create fuzzy output.
- **Defuzzification** Result is an output dimension which shows the suitability to perform specific actions.

The suitability of actions is defined by the membership functions.

Membership Function

For each element in the system there exists a membership value which describes how much an element is related to a statement, partial membership is possible. Describing the grade of membership either formulas or tables are used. Membership functions can have different forms. Basically there are three forms: Triangle, Trapezoid, Singleton (peak at value 1) [28].

Linguistic Terms

For example the terms *it rains* and *it costs money* are *linguistic terms* and are used to characterize the problem statement. They can attain values

between 0 (false) and 1 (true). Giving a statement a value of 0.5 means that this statement is half true. Linguistic terms are human readable and the goal is to use the natural language to build a fuzzy expression like *Temperature t is cold*, *Product x is expensive*.

Fuzzy Sets

The linguistic terms mentioned before are segmented into more precise statements called *fuzzy sets*. For the money example above, fuzzy sets with following values can be created: *it costs little*, *it costs not that much*, *it is expensive*. These terms often overlap and the passages are fuzzy where they end and start. The price is *not cheap* until a certain cap, the degree of fulfilment of the statement *cheap* reduces step on step while expensiveness rises.

Fuzzy Rules

Fuzzy rules are needed to convert input values into the according actions. Using rules all possible combinations and actions are determined. Every rule has one or more input values. These input values have to be converted from crisp values into fuzzy values which is called *Fuzzification*. For every value in the linguistic term, a degree of fulfilment is assigned. Looking at the example of cost and a range of values of 0 and 1, the term *it costs little* has a value of 0.45, the term *it costs not that much* a value of 0.1 and *it is expensive* value of 0. If the fuzzification is ready, the next step is to write the rules which are built with **if then** instructions. Additionally to the existing linguistic terms, another set of terms is added. Suitable values could be the term *wealth*, divided into *low wealth*, *average wealth* and *high wealth*. With both input sets the output actions are calculated. For the output values there are according actions. For this example this could be: *Buy many*, *Buy some*, *Do not buy anything*. This approach is called *Inference*. Different operators for the conjunction between the linguistic terms are available in fuzzy system. Common are *AND* (see figure 5.3) and *OR* (see figure 5.4). The fuzzy logical *AND* is an intersection of two fuzzy sets, and *OR* is the unification [28].

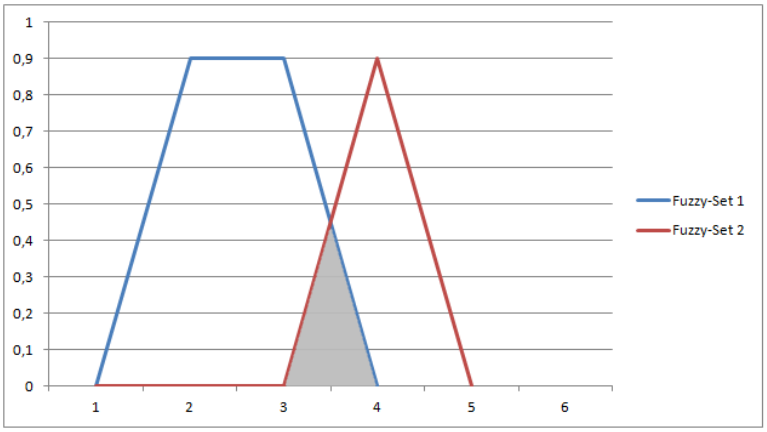


Figure 5.3: Dynamic system with fuzzy logic

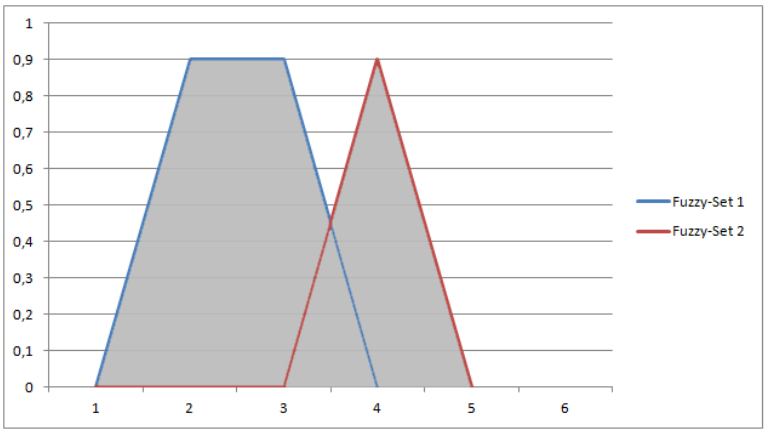


Figure 5.4: Dynamic system with fuzzy logic

The output will be defuzzified into a crisp value on a scale of $\langle -n, +n \rangle$, where $-n$ means that the agent should not buy, 0 is neutral and does not have a fixed action, $+n$ is a good deal and the agent will definitely buy.

Defuzzification

Before any action can be fired, it is important to defuzzify the fuzzy output set into a crisp value with an acceptable range. There are many different methods to do this and basically divided into two groups: maxima methods and distribution methods. The maxima methods are more suitable for reasoning systems, the distribution methods for fuzzy regulators [20].

- **Mean of Maximum** The Mean of Maximum (MoM) calculates the most plausible result. In contrast to averaging the degrees of membership of the output set, the MoM-method selects the typical value of the most valid output term. The disadvantage is, that only the maximum peak of the membership function is considered.
- **Center of Gravity** The Center of Gravitiy (CoG) uses the integral calculus to obtain the centre of gravity of the output set.

5.4 Conclusion

The advantage of using a fuzzy system is, that no models for describing a problem mathematically are needed. This makes it more human readable and easy to understand. It even reduces effort because there is no need to find complex mathematical functions for modelling. Fuzzy systems are commonly used when the specifications of a problem are fuzzy, noisy or incomplete.

Chapter 6

Intelligent Agents

6.1 Overview

When creating an economy for a game, there can be situations where it is necessary to support the economy with artificial agents. These agents should simulate players and they give the game designer the possibility to influence the in-game markets without direct intervention. This can be useful in cases where the economy is not stable as expected or if it is the game's concept to manipulate the market. In the following chapter the concepts of creating an artificial intelligence is illuminated.

6.2 Intelligent Agents

An artificial agent is an computational created intelligent individual, who acts with certain behaviour, situated in an environment. This environment can be real, like for robotic agents, but also only virtual like in computer games. As Franklin and Graesser [8] said: "An agent is a system, situated within and a part of an environment, that senses the environment and acts on it, over time, in pursuit of its own agenda". The agent perceives information about its environment with its sensors and can act with its actuators, both can also be real or virtual. There are many different kinds of agents, also some with learning abilities. For developing agents with economic background, if they should have the ability to learn, we can decide either on agents which learn on their on from previous actions or learn together as a society.

Differences to Object Oriented Design

Designing agents differs from traditional object orientated programming in many issues. An object is an encapsulation of an entity, like a person or anything in the real world. These objects define the structure of a class. The

system of object orientation categorize and arrange the whole program to make a simple picture of the real world problem. The differences to agent systems are following according to Wooldrige [19]:

- Agents are more autonomous than objects.
- Objects have no control when they are used and when methods are executed, agents can decide on their own when to execute certain jobs.
- Agents can have flexible behaviour.
- Multi-agent systems are multi-threaded, at least the agent have one thread of control.

Properties of Agents

Properties defines how agents work and how complex they are. Russell and Norwig [15] determined following properties, which defines the simplest version of an agent (weak agent):

- **Autonomy** The agent is acting by himself without any intervention.
- **Reactivity** The agent reacts to stimulations and decides when to execute its jobs.
- **Pro-activity** The agents reacts in the best possible way to possible actions in the future.
- **Social Ability** The agents has the ability to communicate with other agents or also with human beings.

More complex agents can have more properties [6, 19] for example:

- **Adaptivity** The agent has the ability to learn based on previous experience.
- **Rationality** The agent can perform rational, informed decisions.
- **Versatility** The agent can have multiple goals.

Autonomous Agents

Autonomous agents are working independently from intervention, and are situated in an environment with other agents, and objects but can also be alone, like an data mining agent searching for specific data in a database. These agents can have many different behaviours ranging from purely reactive to cognitive behaviour. A purely reactive agent is only stimulated at it's senses. There is less need to maintain the representation of the environment because the environment is it's own database and can simply look-up by direct interaction. An agent can be called cognitive when it is dependent from the stimulus from outside. It uses representations of the environment to check what is happening and to decide for it's appropriate actions. Reynolds [14] says, that combinations of properties of agents define different classes of agents (see table 6.1).

Table 6.1: Classes of autonomous agents [14]

Type of agent	Description	Example
Virtual agents	Real agents which are situated in a virtual world.	Non-playing characters (NPC) in games.
Virtual semi-autonomous agents	These agents situated in a virtual world and are partly autonomous and partly controlled by humans	First-person shooters in games, agents in sport simulation games
Simulated agents	These agents are situated in a virtual world and studied by a simulation.	Artificial life simulations

6.2.1 AI Agents and Environments

As mentioned before an environment is the world around the agent and everything in it, what is not the agent itself. It lives and operates there. The whole environment is designed after the purpose and must not have realistic limits, because in virtual environments also physics can be altered. Examples for that are games. Artificial agents there, can maybe fly or teleport, but also in simulations there are no boundaries. Environments can be distinguished by their attributes, table 6.2 shows possible environment attributes.

6.2.2 Agent Behaviour

The behaviour of an agent defines the way it acts in certain situations. These situations are defined by environmental conditions, the current circumstances and the knowledge currently available. If an agent do not have enough information to choose the optimal path, it may decide to search for further knowledge about the actual situation. The behaviour can be designed with sub-behaviour [17]. Aspects of behaviour are [17]:

- Sensing and movement,
- Recognition of the current situation with classification,
- Decision-making based on the recognized situation,
- The execution of the appropriate actions.

6.2.3 Trading

For trading, investors have to identify objects which are worth trading. Agents have to determine which objects are either under-priced (good value)

Table 6.2: Environment attributes [19]

Attributes	Description
Observable/Partially Observable	An agent has to observe its environment so this environment itself must be observable. An partially observable environment could be for example the "fog of war" in games.
Deterministic/Stochastic	An environment can be considered fully deterministic when actions always have the same outcome so the future state of an action can be completely determined. A stochastic model have some unpredictable issues and influences the outcome.
Episodic/Sequential	If agent's actions do not affect the future states or influences by previous states then the environment is episodic otherwise sequential.
Static/Dynamic	A static environment does not change like in a turn-based game, in between each moves nothing happens. When an agent does not do any action in an dynamic environment, it's choice is to do nothing.
Discrete/Continuous	In a discrete environment the amount of possible values are limited instead of continuous environment the values are infinite.
Single-agent/Multi-agent	In a multi-agent system agents are able to act cooperatively or competitively with other agents. In single-agent environments the other agents can be seen as part of the environment.

or over-priced. For that the agent or investor has to have a screening to find out in which stock to invest. The more technical approach to this problem is to identify imbalances in the supply and demand economy with the information of prices and volume. A technical indicator could be the moving average convergence divergence (MACD). The MACD is calculated by taking the difference between a long-run and a short-run moving average. A trading rule could be as mentioned in the paper [1]: $IF\ x - day\ MA(movingaverage)\ of\ price \geq y - day\ MA\ of\ price$
$THEN\ buy\ ELSE\ sell$

If the solution is positive than this could be a buy signal and a sell signal could be sent for negative values.

6.3 Conclusion

Artificial intelligent agents are systems which can sense their environment and act accordingly and autonomous. Artificial agents can decide on their own, when and what actions to perform. The environment around the agent has also influence on the behaviour of the agent. It does matter if the environment is fully or only partially observable or if the outcome of actions is always the same. A requirement for an AI agent is the ability of realizing price changes of items.

Chapter 7

Implementation

7.1 Overview

Until here, the thesis presented how to create a working in-game market system with artificial intelligent agents, in theory. In the following chapter, it goes deeper into the real implementation, resulting of the previous gained information and knowledge. Suitable methods for the solution will be described in more detail. The implementation consists of two main parts, the graphical user interface and the second part, the agents, which are also divided into different modes of agents. The different modes describe their behaviour. In the following sections the strategy used is described in detail, also explained with some code snippets and charts.

7.2 Strategy Development

Because this thesis deals with an implementation of an in-game market system, the constraints for this trading system may differ from real financial systems. First it is important to think about what players should be able to do in-game. These constraints are forming the outer shape of the system. The possible actions of players affect also the trading system and hereafter also the work of the agents.

For example it could be possible to design the economy in an ancient way of trading. The players have to go to a special place where they can trade. This concept is often used in *MMORPG's* where the players identifies themselves with the characters. There it is also thinkable that the players can order a stockbroker who is responsible for buy and sell, done for the player. What is important in simulating a realistic market cycles is the speed of running a simulation. For getting fast results it is not possible to run the simulation in real-time. Changes in the economy system should be early recognized and then adjusted till the outcome is acceptable. The simulation should also run several times to get an aggregate behaviour and

robust enough to get realistic economic results from it [10].

The agents in this system should be able to control the market and when desired, even to manipulate it. The agents should give players the feeling, that there are other players in the game world, with which they can trade. Additionally it is desired to keep the market economy dynamic, to give players an incentive to use the market tool.

7.3 In-Game Market System

The dynamic in-game market system described in this paper, was basically developed for the open world multi-player game *Silent6*. The game is situated in the space and is comparable with *Minecraft*. There are no specific goals for the players, they only have to survive. In order of that, it is important that the player collects resources from the planets around him. He need it to upgrade his own ship to get further in the space. He can also defend himself from not getting killed by other players. So the players is forced more or less to collect items. If the player goes further he will find more planets with new and more valuable items. Because of that, the player maybe do not longer need some of the basic stuff. But to throw it away would be a wasting of invested time, so the game gives the player the opportunity to sell his items somewhere and still get revenue from the items. Because of the fact that it is a multi-player game, a overall trading system is very suitable.

When designing a trading algorithm the structure of the market itself has to be determined. It is important how investors or agents can interact with this market. In this thesis there are two basic system modes for the market: *Market order* and *Limit order*.
The first system serves as a platform between players. Players can buy items from another player via the market system. The items are sold in so called: *stacks*. That means, the selling player decides how many stacks he wants to sell, and fixes a price for this stack. This approach is like an auction house, which is often used in games. The other players then can buy exactly this stack for the fixed price. Within the second system, the price is fixed by the market and economy itself. The users can buy the desired amount of items to a fixed price. Normally the value of the item varies to the price of buying. Here the player is able to buy the quantities from an item he wants. That means, he is not bound to a certain stack size but there are no benefits at the pricing. Here it is only possible to place market orders.

7.3.1 The Market's Ecosystem

In the current implementation the market order system is accomplished. It consists of two parts the graphical user interface and the agents working in

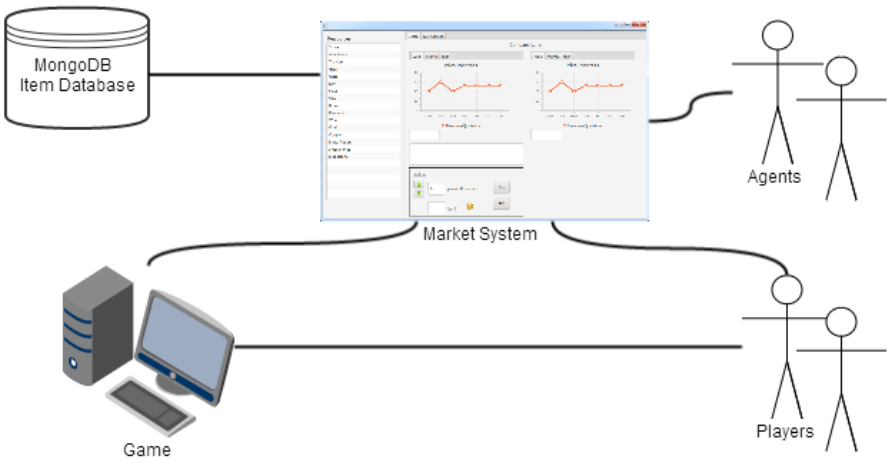


Figure 7.1: System overview

the background of the system. Figure 7.1 shows an overview of the actual implemented market.

The market system is bound to an *MongoDb* database in which all items are saved. At the first start, the game designer can add a list of items with initial amount and price, also with a description for each item. At start the program connects to this database and grabs the data and insert it into the market. The agents are invisible to the users. They work in the background of the program and interacting directly with the market place. Different agents with different logic and behaviour are situated here and try to decide on buy or sell actions at specific time steps. The players interact over the game and the graphical user interface (GUI) with the market and can also do the same as the agents. The functions can be described in more detail, figure 7.2 shows the steps graphically.

When resources going out and into the market, the amount steadily changes. Also the price of each resource have to be adjusted to fulfil the requirements of a demand and supply system. It is important to do this step because otherwise there would be no dynamic behaviour in the market. The prices would always be the same and it would be very boring to use the market. So at specific time-steps, which are also editable by the developer, the prices of each resources are adjusted to fit the new amount of items available on the market. For the calculation only items in the market itself are considered, items which are in the inventories of players and agents, are not included in the calculation. The calculation is kept simple: For every percent of items which are missing compared to the previous time-step, the

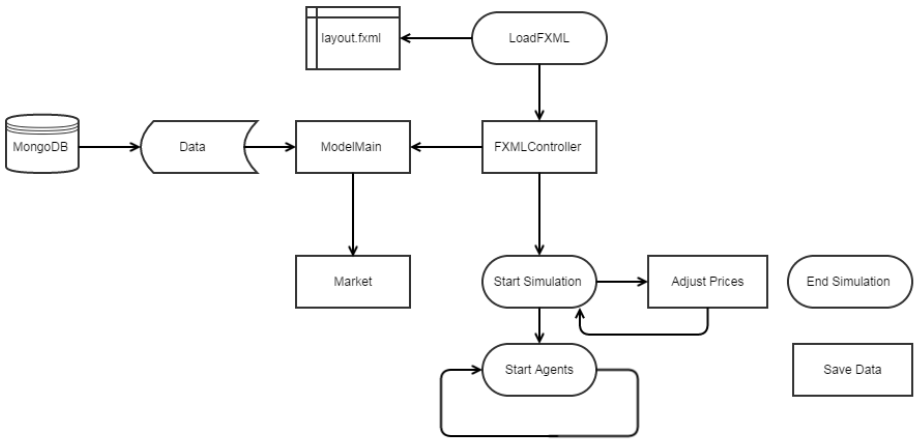


Figure 7.2: Flowchart of system

price rises or fall for exact this value of percent. If there are 5% less items of one resource in the market than before, the price will rise to +5% of the original price. The consequences are: if players only buy one specific resource but do not sell it into the market, the prices will rise slowly and in the future it will be very expensive to purchase these items. The other way round is when players only selling the same resource, there will be too much items in the market and the price will drop. This could then be attractive to buy these resources again, because they are cheaper than before.

7.3.2 The Market Interface

The market interface itself (see figure 7.3), is created for the players for placing sell and buy actions. It has a list with available items in the market, with informations like price and available amount of each item. The players can buy and sell items with desired amount but to a fixed price. The value of the item and its price to buy differs around 10% but can also be set to a other desired value or even be adjusted dynamically during the game's life. So in this market system it is only possible to place buy orders and the transactions are done immediately.

The market is also able to offer more features, but not implemented yet:

- **Comparison of two items** Price and price over time.
- **An order book** All actual orders from the player are saved here (needed for market order system).
- **My tradings** At this tab the player can have a look at his previous transactions.

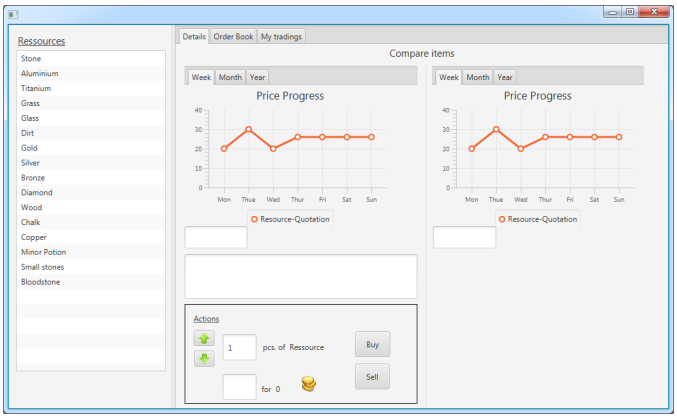


Figure 7.3: Graphical market tool

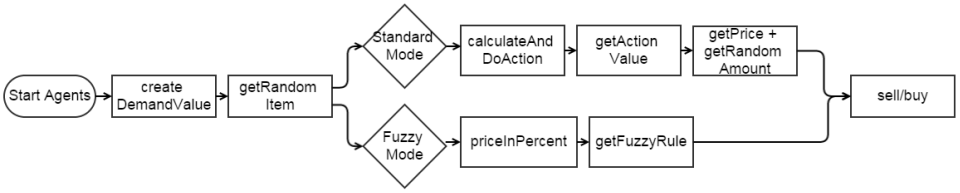


Figure 7.4: Agent's procedure

7.4 The Design of the Agents

The artificial trading agents were developed because of the fact, that in early stages of the game, there will be few gamers, but the market system, as an important feature in the game, will only be interesting, when it makes sense to use it. If there are few players, then the market will not be very dynamic, players will not be able to make good transactions or big profit. So the idea came up to implement an artificial agent system, which is able to simulate the missing players. It should also be able to handle the ecosystem, so that the game designers can influence and manipulate the market indirectly if needed.
A system was developed which uses a complex and simple trading agent. The method for the complex agent is divided into different phases, graphically described in figure 7.4. The agent is able to handle the standard mode or using fuzzy logic.

The agent depends on his demand on a certain product and quantity of this product. The demand depends on the actual market situation and on chance, as shown in figure 7.5.

The agent has a value of demand, for measuring how much he needs the goods. At start the initial value of demand is set randomly. The possible

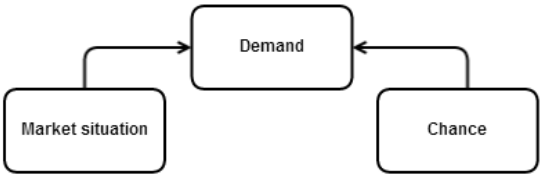


Figure 7.5: Demand composition

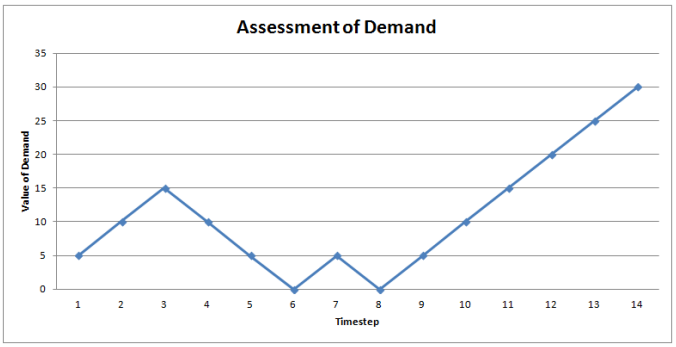


Figure 7.6: Assessment of Demand

range for values are 1% to 100%. To get the value of demand, a assessment of demand is required. For deciding at which item the action should be done, the agent gets an item randomly. Also the amount of the items is set randomly.

7.4.1 Assessment of Demand

The agent starts with a randomly selected initial value of demand. Then at a specific time-step this value is raised or lowered. At every time-step the value is manipulated at fixed steps, means, that at the beginning of the simulation, a value for the units to step is set, that could be for example at 5%. So at every time-step the value of demand is raised/lowered for 5%, starting by 1% ending by 100%. For retrieving the info if it should raise or fall, a random selected value of 0 or 1 is taken. If 0 the value of demand lowers, if 1 the value rises. An example is shown at the chart in figure 7.6.

The resulting percentage is then used to identify the actions which are available to the agent at a certain value of demand. The agent has to decide, out of a pool of actions which to take, depending on the received value of demand. It can either be done in the way shown above or in fuzzy mode which is described in the next section. The last important issue for the agent to decide on buying or selling an item is the assessment of price.

7.4.2 Assessment of Price

The assessment of price is needed to decide, if the actual price of an item is cheap or expensive. First, the quality of price has to be calculated. This is done with

$$\frac{ActualPrice}{PaidPrice} = PriceValue. \tag{7.1}$$

If the agent did not bought any resource of this item before, then the actual price of the item, is used for the next steps. With this new information it is possible for the agent to make a decision on the action. The previous created values of demand can now be used for that. The values have to be classified into groups of actions. The classification depends on the design of the agent, if it has a more greedy or rational thinking. Another developed method for this thesis, is to define actions for specific percentage steps. Percentage steps are fixed thresholds at which the actions change.

Looking at the value of demand at 50%. As a reference the quality of price is taken. If the result is greater than 1 than the agent will sell the item. If the result is lower than 1 than the agent will buy it. The thresholds for the actions are defined for every percentage step. At both ends of the scale only one action can occur. That is because of the fact, that items would be sold, when the price is very good but also having high demand at the top of the scale, and would be bought at a good price but with absolutely no demand at the lower end of the scale.

7.4.3 Conclusion of Agent Solution

Because of the randomly selected values the whole system is very dynamic and can simulate irregularities of the economy. This system also gives the opportunity to tweak values to adjust the procedures to specific needs. Different values can have different impact on the agents behaviour. The results can be easily compared but can also uses different agent behaviours. The constants time-steps, initial value of demand and percent steps can be changed to tweak behaviour. The time-steps could be greater or smaller, or also randomly changed. The percentage steps for the value of demand can be bigger/smaller/random. The values also can be used in a more dynamic way and can change over time.

7.5 Other Methods

7.5.1 Fuzzy Approach

A second method for decision making is implemented. It is the approach of using fuzzy logic. Fuzzy logic can be used for fuzzy regulation, that is

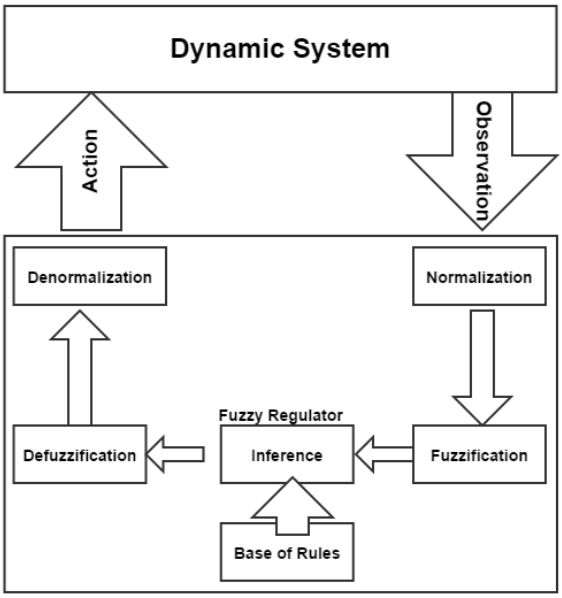


Figure 7.7: Dynamic system with fuzzy logic [28]

a system that can actively regulate a dynamic environment (see figure 7.7) [28].

The approach starts with the observation of the environment, means that the agent tries to figure out the requirements of the system and of the world around him. Next the available crisp values have to be fuzzified. That means that the input variable *the price is 10 currency* has to be converted into *It is cheap*. But no program can work with such expressions, so every linguistic term also has an membership function which describes the membership to a specific group. The more the term belongs to a group the higher the value of the membership. If the value is definitely not in the group, then the membership is 0. The next step is the *inference*. It is simply the progress of converting the input fuzzy set, with the first fixed rules, to a output fuzzy set. All outputs with an membership greater than 0 have to be considered and be converted with an mathematical function into a crisp value which is called *defuzzification*. The rules shown in table 7.1 are used in the implementation.

At the end of the process the program has an fuzzy output value. With the usage of an method of defuzzification this fuzzy value, is converted into a crisp value with which decisions can be made. For the creation of the fuzzy approach a Java library is used: *jFuzzyLogic*, it is described in more detail in the next section.

Table 7.1: Fuzzy rules

Rule	Cost	Need	Result
1.Rule	low	low	buy
2.Rule	average	low	sell
3.Rule	high	low	sell
4.Rule	low	average	buy
5.Rule	average	average	sell
6.Rule	high	average	sell
7.Rule	low	high	buy
8.Rule	average	high	buy
9.Rule	high	high	buy

7.6 Tools and Frameworks

7.6.1 Mongo DB

MongoDB is an open-source cross-platform document oriented database. It is an *NOSQL*-based system and written in *C++*. Because it is document-based the database can handle *JSON*-like objects which makes it easier to model the data because it is a more natural approach. In this project *MongoDB* is used to store the games item data.

At start of the market tool, the database must run and the market connects to the database and receives its items which are populated before out of an .json document.

The item resource list is written in *JSON* format. Developers can manipulate the available data entries and if adding new elements, they should have the format

```
1 { "_id" : { "\$oid" : "52f0c534996e0a0e09d06458" }, "name" : "
      Stone", "detail" : { "amount" : 31, "price" : 31, "
      description" : "This is Stone" } }
```

The connect method

```
1 try {
2   client = new MongoClient("localhost", 27017);
3
4   /**** Get database ****/
5   // if database doesn't exists, MongoDB will create it for you
6   database = client.getDB("marketDb");
7
8   /**** Get collection / table from 'testdb' ****/
9   // if collection doesn't exists, MongoDB will create it
```

```
10
11   collection = database.getCollection("items");
12
13   } catch (UnknownHostException e) {
14     e.printStackTrace();
15   }
16 }
```

is used to connect to the running *MongoDB* instance. After the connection to the database was successful the items are requested from it

```
1 public Set<String> getCollections() {
2 Set<String> tables = database.getCollectionNames();
3
4 for (String coll : tables) {
5   System.out.println(coll);
6 }
7 return tables;
8 }
```

And as easy it is to get this data, it is to change it

```
1 public void updateData(String _keyOld, Object _valOld, String
      _keyNew, Object _valNew) {
2 BasicDBObject query = new BasicDBObject();
3 query.put(_keyOld, _valOld);
4
5 BasicDBObject newDoc = new BasicDBObject();
6 newDoc.put(_keyNew, _valNew);
7
8 BasicDBObject updateObj = new BasicDBObject();
9 updateObj.put("\$set", newDoc);
10
11 collection.update(query, updateObj);
12 }
```

7.6.2 JavaFx

JavaFX is an framework for cross-platform *Rich-Internet-Applications*. The current release supports desktop computers and web browsers and there is also a mobile version for mobile platforms. It is pre-installed since Java 7.6 and since *JavaFX 2.0*, developers can write *JavaFX* code in standard *Java* style, in previous versions a scripting language was used. Many graphical objects are derived from the original *Java* GUI environment. New custom elements are easy to create, reuse and they are style-able with *CSS*. Oracle also introduced the *SceneBuilder* in version 2.1. It is a graphical design tool with which developers are able to create GUI easily with drag- and drop-able GUI elements. The so created layout is then saved in an *.fxml* file which is an special form of *XML*. This file then can be loaded in the standard Java application. Such an *FXML* file can look like

```
<AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="385.0"
  prefWidth="575.0" xmlns="http://javafx.com/javafx/8" xmlns:fx="
    http://javafx.com/fxml/1"
  fx:controller="at.fhooe.im.controller.FXMLControllerAgent">
  <children>
    <Button fx:id="startAgentButton" layoutX="421.0" layoutY="
    94.0"
      mnemonicParsing="false" onAction="#handleButtonEvent"
    prefHeight="40.0"
      prefWidth="102.0" text="Start Agent" />
    <Button fx:id="stopAgentButton" layoutX="421.0" layoutY="
    148.0"
      mnemonicParsing="false" onAction="#handleButtonEvent"
    prefHeight="40.0"
      prefWidth="101.0" text="Stop Agent" />

    <Label layoutX="421.0" layoutY="8.0" prefHeight="30.0"
      prefWidth="45.0" text="Timer: " />
    <Label fx:id="labelTimer" layoutX="472.0" layoutY="9.0"
      prefHeight="29.0" prefWidth="56.0" text="Label" />
    <TextArea fx:id="textFieldData" layoutX="20.0" layoutY="14.0"
      prefHeight="356.0" prefWidth="385.0" />
  </children>
</AnchorPane>
```

Loading the *JavaFX* application is done in the main method

```
public static void main(String[] args) {
  Application.launch(FXMLLoad.class, args);
}

@Override
public void start(Stage primaryStage) throws Exception {
  instance = this;
  mainContainer = new ScreensController();
  agentContainer = new ScreensController();
  agentContainer.loadScreen(AGENT_SCREEN, AGENT_SCREEN_FXML);
  agentContainer.setScreen(AGENT_SCREEN);

  StackPane secondaryLayout = new StackPane();
  secondaryLayout.getChildren().add(agentContainer);
  Scene secondScene = new Scene(secondaryLayout, 600, 400);

  Stage secondStage = new Stage();
  secondStage.setTitle("Agent");
  secondStage.setScene(secondScene);
  secondStage.show();

  mainContainer.loadScreen(MARKET_SCREEN, MARKET_SCREEN_FXML);
  mainContainer.setScreen(MARKET_SCREEN);
```

```
24
25   StackPane root = new StackPane();
26   root.getChildren().add(mainContainer);
27   Scene scene = new Scene(root);
28   primaryStage.setScene(scene);
29   primaryStage.show();
30 }
```

with a reference to the .fxml layout file. With the creation of different stages, it is possible to handle multiple instances of *JavaFX* windows. For accessing the data from this *FXML* design files there is the need to create a controller class for each layout file in which the GUI elements can be accessed with the same name qualifier with the notation *@FXML*

```
1 @FXML
2 private Button buttonSell;
3 @FXML
4 private ListView<String> itemList;
```

7.6.3 jFuzzyLogic

jFuzzyLogic[1] is an open source *Java* library for *Fuzzy Logic*, also including an *Eclipse* plug-in. It implements a *Fuzzy Control Language (FCL)* to describe fuzzy systems to reduce programming effort. The fuzzy system can be easily described in a fuzzy manner. The library contains many methods for defuzzification and different methods for rule creation [2, 3].

The following code shows an example of an *FCL* file

```
1 VAR_INPUT    // Define input variables
2   need : REAL;
3   costs : REAL;
4 END_VAR
5
6 VAR_OUTPUT    // Define output variable
7   action : REAL;
8 END_VAR
9
10 FUZZIFY need  // Fuzzify input variable
11   TERM minimal := (0, 1) (40, 0) ;
12   TERM average := (10, 0) (40,1) (60,1) (90,0);
13   TERM high := (60, 0) (100, 1);
14 END_FUZZIFY
15
16 FUZZIFY costs // Fuzzify input variable
17   TERM low := (-20, 1) (20,0) ;
18   TERM average := (-10, 0) (30,1) (50,1) (60,0);
19   TERM high := (50, 0) (100,1);
```

[1] http://jfuzzylogic.sourceforge.net/html/index.html

```
20 END_FUZZIFY
21
22 DEFUZZIFY action // Defuzzify output variable
23   TERM doNothing:= (0,0) (5,1) (10,0);
24   TERM sell := (10,0) (15,1) (20,0);
25   TERM buy  := (20,0) (25,1) (30,0);
26   METHOD : COG; // Use 'Center Of Gravity' defuzzification method
27   DEFAULT := 0; // Default value is 0
28 END_DEFUZZIFY
29
30 RULEBLOCK No1
31   AND : MIN;    // Use 'min' for 'and'
32   ACT : MIN;    // Use 'min' activation method
33   ACCU : MAX;   // Use 'max' accumulation method
34
35   RULE 1 : IF need IS minimal AND costs IS high THEN action IS
       sell;
36   RULE 2 : IF need IS minimal AND costs IS average THEN action IS
        sell;
37   RULE 3 : IF need IS minimal AND costs IS low THEN action IS buy
       ;
38
39   RULE 4 : IF need IS average AND costs IS average THEN action IS
        sell;
40   RULE 5 : IF need IS average AND costs IS high THEN action IS
       sell;
41   RULE 6 : IF need IS average AND costs IS low THEN action IS buy
       ;
42
43   RULE 7 : IF need IS high AND costs IS high THEN action IS buy;
44   RULE 8 : IF need IS high AND costs IS average THEN action IS
       buy;
45   RULE 9 : IF need IS high AND costs IS low THEN action IS buy;
46
47 END_RULEBLOCK
48 END_FUNCTION_BLOCK
```

With the *FCL* language it is easy to describe the fuzzy system. It accepts every size of variables and membership declarations. Also different methods for defuzzification are available. The framework is able to understand *Java* code and the fuzzy system also could be implemented only in *Java*. With the according Java classes it is possible to get results directly at runtime and use them to get the crisp output values for the actions.

7.7 Conclusion

For the creation of the agents, the gained knowledge from the previous chapters is used. There is the standard method where the agents decide depen-

dent on demand values and prices, and the additional method where the decisions for actions happens using fuzzy logic. Fuzzy logic is an convenient approach to describe an system which has no crisp values, and where it is hard to decide for one or another action. It is a handy way to decide, if an resource is to expensive for the actual value of desire. It also offers a way to make a problem more human readable and then pass it to the program which can also work with these expressions.

Chapter 8

Evaluation

8.1 Overview

The implementation of the market system and the agents which belongs to it, has to be tested and evaluated, to examine if the expectations are fulfilled. In this chapter it is described how this is done and if the result is showing a success. First the parameters and requirements of the agents are described. Followed by the results of changing and experimenting with parameters. At the end of this chapter, a résumé is given, if the used approaches are suitable for the idea of creating an in-game-market system, controlled by artificial agents.

8.2 How to Test Artificial Agent Systems

Testing AI systems is more complicated than simply testing if the programming part works. It is important to analyse the outputs and compare it to other similar systems, to get a solid picture of the systems behaviour, and if it is working properly. Without the correct evaluation, it is nearly impossible to check if progress has been made, or if the design issues have met. For evaluation the right questions have to be asked and also the answers have to be interpreted in the right way. Basic questions to be asked can be:

- Is the system stable? What happens when prices drop/rise heavily?
- How do the agents react?
- How have prices changed?

The case of heavily dropping or rising prices, is described by the phenomenons of *Inflation* and *Deflation*. The clear definition of these terms are described in the previous chapter of the economical basics. Important in these situations is, that the game design is able to catch the economy before it gets critical. A simple solution to both situations could be, that the agents can change their behaviour and do only sell or buy actions, depending on

the current situations. An example: If the market shows indicators of rising prices, they could possibly only do selling actions to get more items back into the market economy. For deflation then the other way round. For being able to answer all questions proposed to the system it is also relevant to define the basic parameters. Different parameters with different values will bring different results. For testing the system efficiently, it is necessary to define a bunch of varied starting situations.

8.3 Setting the Basic Parameters

For testing variations of situations and entry points, the agents are designed to accept a configuration file. For every different agent, a agent configuration file has to be passed. The file consists of these initial values

```
1 public boolean    fuzzyMethod       = false;
2 public String   name          = "";
3 public int      initialCapital      = 0;
4 public int      maxAmount       = 0;
5 public int      initialDemandValue    = 0;// 1-100 possible
6 public int      demandValueSteps    = 0;// 1-100 possible
7 public int      actionSpeed;// in milliseconds
```

These values arrange the outcome of the agents actions and if the values are set different, also the agents acting different. Every parameter affects the agents in different ways.

The parameters in the configuration-file are all required:

- **FuzzyMethod** Defines if the agents should use the fuzzymethod or the normal method.
- **Name** The agent can have a name as identifier to distinguish between others.
- **Initial Capital** Sets the initial capital value for the agent.
- **Max amount** Is used to define how much items it can buy/sell at each transaction.
- **Initial demand value** The demand value is needed to decide how important it is for the agent, to do the actions).
- **Demand value steps** This parameters fixes the steps, with which the agents demand values are rising/falling.
- **Action speed** Sets the speed of the agent.

The configuration issues can be easily extended and passed to a new designed agent with new behaviour. After setting initial values and behaviour, the concrete test cases in which the agents should act have to be defined.

Table 8.1: Test-cases

Parameters	Test 1	Test 2	Test 3
Amount of items on market	200.000	1.000	500.000
Amount of money for agents	10.000	10.000	1.000
Maximal amount for actions	20	50	50
Initial value for demand	50	50	50
Steps for demand value	10	10	10

8.4 Define Testing Situations

For testing the agents with set parameters, it is important to first decide on a test case. Test cases define the world around the agents. That means that also the market tool receives initial values. Dependent on these values, the agents will behave differently. Parameters for the market itself can be the initial amount of resources and also the according prices. Changes there, will have direct impact on the behaviour of the agents and also on the whole economy. For the first tests, the agents are tested with a market with very few/much resources available. According to this test case following questions are formed:

- Does the agent buy/sells anyway?
- How the simple agents react?
- How the complex agent react (fuzzy, normal)?
- What effects does the price adjustment have?
- How the initial values effect the agent behaviour?

At start of the tests, it is hard to predict the exact outcome, so maybe there is the need to adjust the questions during the testing phase. For getting an game environment which fits the needs of the game, can be an elaborate task and has to be done by trial and error and regularly adjustment of parameters and values.

The following table 8.1 shows the exact parameters which are set for the testing sessions. Three different tests were started, where each test consists of a simple agent, a complex agent with his standard behaviour and a complex agent with fuzzy behaviour. In each test session, each agent can perform 1440 transactions, which matches a day in real-time simulation, because the agents are restricted to only make one transaction per minute.

The initial amount of the items is set by the database, the parameters are set in the agent's configuration files and passed to the agents at the start of the simulation. Looking to the next section, the according results of these test cases are shown and described in detail. To be able to compare the

results, every agent receives the same amount of money and also the other parameters are set with the same values.

8.5 Results

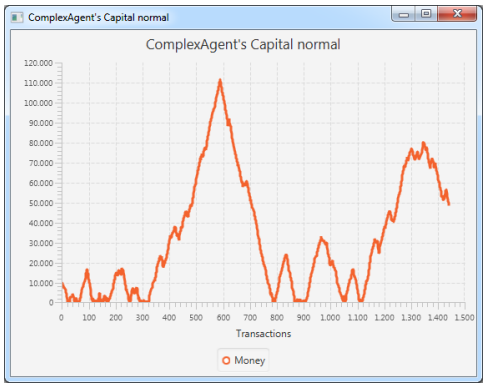
After setting all steps including initial values and test cases, the system can be run for testing. Starting with the first test where market and agents wealth is balanced.

8.5.1 Test-Run 1

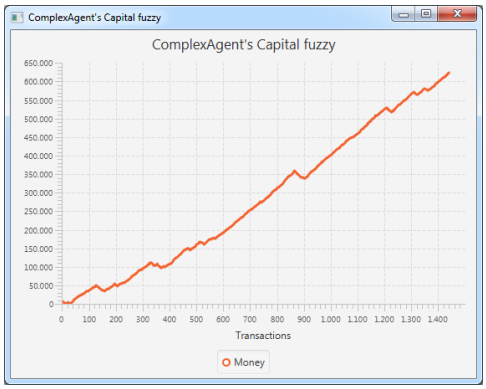
In the first test, both the agent's and the market's wealth are balanced. So neither the market nor the agents are poor in items or currency. The results of the first test case are shown in figure 8.1. In the first figure 8.1a the wealth over time of the complex agent in its normal mode, can be seen. It is clearly visible, that there are heavy drops, but it also rises again. The heavy drop is resulting of a transaction of buying a huge amount of an expensive item. At the current status of implementation, the agent does not estimate the amount of the items in each transaction. The value to choose, is set randomly with a cap. Instead looking at figure 8.1b, where the fuzzy approach is displayed, it can be recognized that there are no big drops. The agents wealth is rising steadily. With that information, it can be assumed that the fuzzy approach is better to weight the goodness of an price of an item. The fuzzy approach is also more handy because the game designer can build the rules and define the actions, as he imagines. The problem with the normal mode is, that setting the borders, when to do which action, is harder to decide and maybe needs more time to tweak, to get the expected result. But this approach is nevertheless not useless, because maybe the game designer intends the agent to work not perfect and also to make loss.

There is also another agent used in the tests (see figure 8.2a), the simple agent, which only does sell and buy transactions randomly. He does not consider, if the price of selling is good or bad, and the amount of items for each transaction is also set randomly with a cap. Depending on the use case, even this simple approach can be used but behaviour can only be influenced by the initial parameters.

The market system is able to adjust the prices of items, according to the current amount on the market. Figure 8.2b shows the course of price over the time, in which the agents can perform transactions. Because there are more buy than sell actions and therefore more resources in the market, the price is slowly falling.

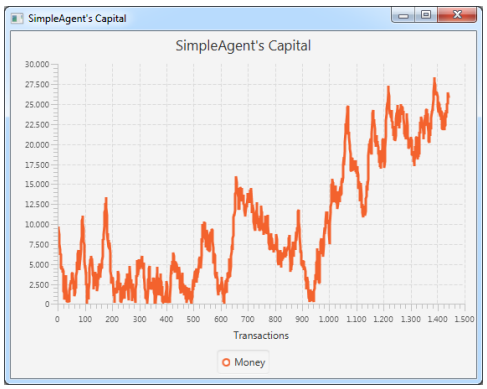


(a) Complex agent's in normal mode wealth over time

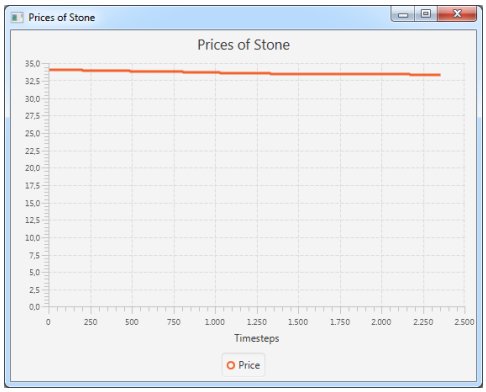


(b) Complex agent's in fuzzy mode wealth over time

Figure 8.1: Test-case 1: Comparison of complex agent's behaviour modes



(a) Simple agent's wealth over time

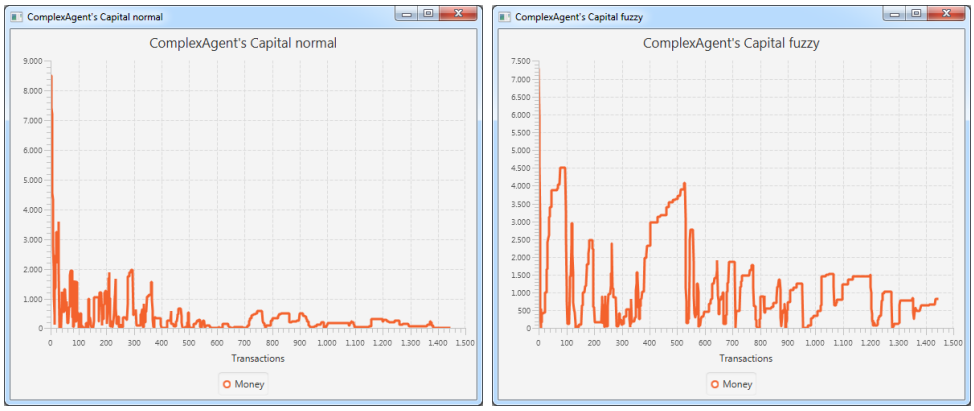


(b) Price of item over time

Figure 8.2: Test-case 1: Simple agent's wealth and price of an item over time

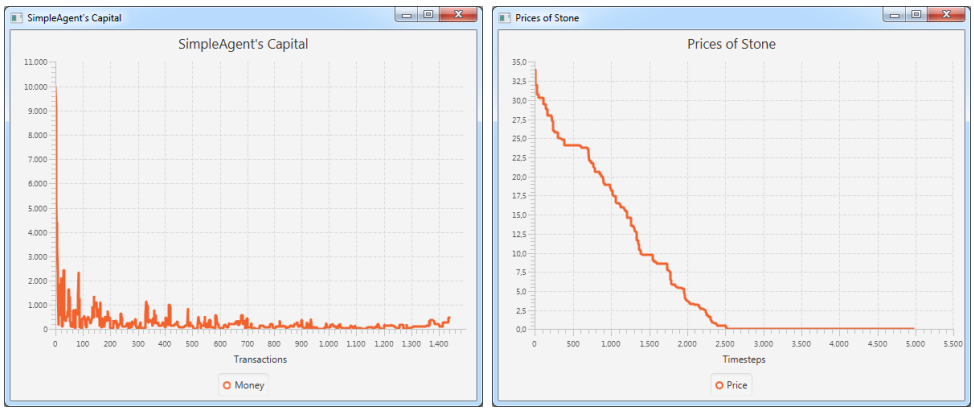
8.5.2 Test-Run 2

In the second test the market is poor in items, but the agent is rich in currency. Looking at the charts in figure 8.3 it can be seen, that the amount of available resources on the market, has direct impact on the wealth of each agent. If the market is poor in items, the agents do not take much time to buy every item in the market, which causes in extremely dropping prices (see figure 8.4b). If there is no system, players or other agents, who can fill up the market with resources steadily, the agents will not be able to make much



(a) Normal agent's wealth over time **(b)** Fuzzy agent's wealth over time

Figure 8.3: Test-case 2: Comparison of complex agent's behaviour modes



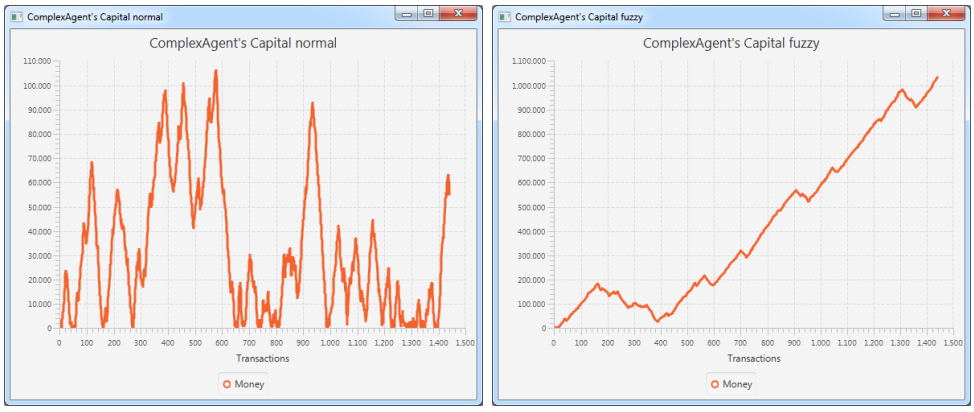
(a) Simple agent's wealth over time **(b)** Price of item over time

Figure 8.4: Test-case 2: Simple agent's wealth and price of an item over time

profit, which is clearly visible in the chart of the normal agent's chart 8.3a the fuzzy agent's chart 8.3b and the chart 8.4a of the simple agent. The profit curve is situated in the lower half of the chart and is not able to rise, because there are no resources to buy.

8.5.3 Test-Run 3

In the last and third test, the agent is poor in currency and the market is rich in items. Despite the fact, that the agent has not much currency at start, the fuzzy approach is able to make good profit over the time (see 8.5b). Generally a random based approach like the simple agent's behaviour, clearly does not have a stable outcome. The price of the item (see figure 8.6b) is again slowly falling, because of the perfect work of the fuzzy agent, who is able to maximize it's profit, even if the initial values are bad. The behaviour of the complex agent using the normal mode, is not optimal in the point of view of profit. The tweaking of the parameters of this agent, is more complicated and obscured.

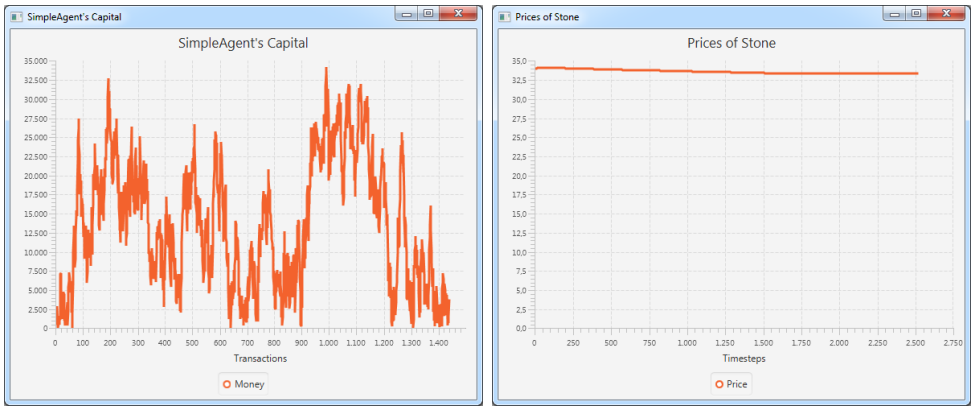


(a) Normal agent's wealth over time **(b)** uzzy agent's wealth over time

Figure 8.5: Test-case 3: Comparison of complex agent's behaviour modes

8.6 Conclusion

The evaluation of artificial intelligent systems can often be hard. Also game design has an impact on the test-cases and often trial and error is the only method to get valuable and desired results. After some testing-sessions it can be assumed, that the ecosystem and it's according agents, met the majority of the goals set at start of the project. The agents are able to decide for different actions dependent on the actual parameters. The behaviour can be changed at start by changing initial parameters. Each parameter influences the final outcome. Each agent has to run in its own thread where it is encapsulated from the primary game loop. That makes it possible, that the game also can run without it's agents. They can even be removed in a later state of the game, when there are not needed any more. Look upon the differences



(a) Simple agent's wealth over time **(b)** Price of item over time

Figure 8.6: Test-case 3: Simple agent's wealth and price of an item over time

of the normal and the fuzzy agent, it can be assumed, that both approaches are usable in a game economy. Both can interact dynamically with their environment and can be adjusted to fit the specific needs. The agents can also be duplicated and their clones can have other initial parameters.

Generally, the fuzzy agent is a straightforward approach. with the usage of the *jFuzzyLogic* library, it is pretty easy to define rules, membership values and defuzzifier methods. Each value of membership can be adjusted simply. The fuzzy agent can be trimmed, to work profit orientated, as seen in the test-cases.

The agent in normal mode, is a bit more complicated in usage. The developer has to have a clear idea of caps for the prices. He has to define clear caps where prices are too high or which value of demand represents a certain group of need. The output of the normal mode agent, is not as clear as of the fuzzy one and depends highly on random values. This leads to the fact, that the behaviour is maybe not clearly fixed. This agent is more suitable, when developers intend to make the agent to work not in a perfect way. Even the simple random agent can be useful in-game, when the requirement is, to be unpredictable.

Summarized it can be stated, that the chosen plans and ideas of the implementation are an success. They basic requirements are satisfied. There are different agents, which are easy to extend and behaviour can be tweaked. They are able to interact properly with their environment and do not cause any inferences on the players. The agents are capable of manipulating the market and simulate other players.

Chapter 9

Conclusion

The topic of virtual markets either in games or in real life, is a very interesting subject in sciences, because it is also an very actual topic. There is a big spreading of different knowledge bases: finance and economy theory, artificial intelligence when it comes to simulating agents, different methods of learning algorithms like evolutionary algorithms and neural networks, game design when it comes to add virtual markets to a game or when to research economical issues with games. The thesis tried to show the diversity of this subject and to present a specific part out of this topic pool. It dealt with the creation of an simulated market system for a multi-player game. To get into this topic, it was essential to present the low-base economical basics which are used. Furthermore it has to be described how virtual markets are in actual games, how the work and what other issues there are which are associated with virtual markets. The creation of a in-game economy differentiates from economies in the real world, because the requirements are different. But concepts which are used for simulating real world markets, are also useful for simulating a game economy. Because a simulated market can not work without participants, it is also meaningful to implement an artificial agent system which are capable of interacting with the virtual economy and influence it. The ideas of how to simulate markets are coming in here and used in one of the final agents. The ideas from the previous chapters are then united in the last chapters of implementation and evaluation, where a real in-game market simulation has been built.

9.1 Development Potential

The main goals which were set at the beginning, are almost accomplished. The now available framework prototype can be taken as a basis for further development and research. Sadly it was not possible to test the system in the real game, only in a closed testing environment. That would be a highly desirable goal for the future. Because only in the real environment, it is

possible to test appropriate and to adjust the system to the game. The presented implementation is in a prototype status and can be further developed. It would be interesting to test more different methods for the agents like genetic algorithms or other appropriate approaches presented in this thesis. For the development of the agents, there are many possibilities to change or optimize the behaviour. Suitable and desirable improvements would be:

- **Smarter agents**
  - Actions depend on the agent's wealth and available items.
  - Dynamic behaviour depending on the actual market and economy situation. More buy-actions when the market has few resources left and more sell-actions when there are many resources.
  - Agents have the ability to learn from their previous actions and change them.
  - Agents can look at previous price changes, even if they have not bought that item before.
- **Service for item creation** Add a service which is responsible to fill the market with new items to prevent that the market can run out of resources when agents buy to much.
- **More agents** Creating more agents, even of the same type. This could be interesting to test, if the behaviour changes if there are more agents.

References

Literature

[1] Michael O'Neil Anthony Brabazon. "An Introduction to evolutionary computation in finance". In: *IEEE Computational Intelligence Magazine* 4 (2008), p. 14 (cit. on pp. 19–22, 30).

[2] Pablo Cingolani and Jesus Alcala-Fdez. "jFuzzyLogic: a Java Library to Design Fuzzy Logic Controllers According to the Standard for Fuzzy Control Programming". In: *International Conference on Fuzzy Systems*. IEEE. 2013, pp. 61–75 (cit. on p. 43).

[3] Pablo Cingolani and Jesus Alcala-Fdez. "jFuzzyLogic: a robust and flexible Fuzzy-Logic inference system language implementation". In: *International Conference on Fuzzy Systems*. IEEE. 2012, pp. 1–8 (cit. on p. 43).

[4] Mikael Collan, Christer Carlsson, and Péter Majlender. "Fuzzy Black and Scholes Real Options Pricing." In: *Journal of Decision Systems* 12.3-4 (2003), pp. 391–416. URL: http://dblp.uni-trier.de/db/journals/jds/jds12.html#CollanCM03 (cit. on p. 8).

[5] Mikael Collan and Shuhua Liu. "Fuzzy logic and intelligent agents: towards the next step of capital budgeting decision support". In: *Industrial Management & Data Systems* 103.6 (2003), pp. 410–422. eprint: http://www.emeraldinsight.com/doi/pdf/10.1108/02635570310479981 (cit. on pp. 8, 9).

[6] O. Etzioni, Oren Etzioni, and D. S. Weld. "Intelligent Agents on the Internet: Fact, Fiction, and Forecast". In: *IEEE Expert* 10 (1995), pp. 44–49 (cit. on p. 28).

[7] Joshua A.T. Fairfield. "Virtual Property". In: Indiana Legal Studies Research Paper No.35. Boston University Law Review. Washington & Lee University School of Law, 2005 (cit. on p. 2).

[8] Stan Franklin and Art Graesser. "Is It an Agent, or Just a Program?: A Taxonomy for Autonomous Agents". In: *Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Lan-*

guages. ECAI '96. London, UK: Springer-Verlag, 1997, pp. 21–35 (cit. on p. 27).

[9] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989 (cit. on p. 22).

[10] Xin Yao Herbert Dawid Han La Poutre. "Computational Intelligence in Economic Games and Policy Design". In: *IEEE Computational Intelligence Magazine*. Nov. 2008, p. 75 (cit. on pp. 7, 19, 33).

[11] Bernd Lehahn. "The open universe in X rebirth". In: *Making Games* 6 (2013), pp. 32–35 (cit. on p. 5).

[12] Vili. & Ernkvist Mirko Lehdonvirta. *Converting the Virtual Economy into Development Potential: Knowledge Map of the Virtual Economy*. Washington, DC; infoDev World Bank, 2011. URL: http://www.infodev.org/articles/converting-virtual-economy-development-potential-knowledge-map-virtual-economy (cit. on p. 15).

[13] Simon Ludgate. "Virtual Economic Theory: How MMOs Really Work". In: *Gamasutra* (Nov. 2010). URL: http://www.gamasutra.com/view/feature/134576/virtual_economic_theory_how_mmos_.php? (cit. on pp. 2, 3, 15, 17).

[14] Craig Reynolds. *Steering Behaviors For Autonomous Characters*. 1999 (cit. on pp. 28, 29).

[15] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 2nd ed. Pearson Education, 2003 (cit. on p. 28).

[16] Christoph A. Scherbaum. *So funktioniert die Boerse*. Haufe Lexware, Feb. 2013 (cit. on pp. 11, 12).

[17] William John Teahan. *Artificial Intelligence - Agent Behaviour*. Bookboon.com, 2010 (cit. on p. 29).

[18] Michael Wellman and Peter R. Wurman. "A Trading Agent Competition for the Research Community". In: *Proc. of IJCAI-99 Workshop on Agent-Mediated Electronic Trading*. 1999 (cit. on p. 8).

[19] J.M. Wooldridge. *Econometric Analysis of Cross Section and Panel Data*. MIT Press, 2002 (cit. on pp. 28, 30).

[20] Lotfi A. Zadeh. "Fuzzy sets". In: Information and Control. University of California, Berkley, California, 1965, pp. 338–353 (cit. on p. 25).

[21] D. Zhang and K. Zhao. "Economic model of TAC SCM game". In: *Proc. of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*. Beijing, China. Sept. 2004, pp. 273–280 (cit. on p. 8).

Online sources

[22] Bundeszentrale für politische Bildung. URL: http://www.bpb.de/nachschlagen/lexika/lexikon-der-wirtschaft/19723/inflation (cit. on p. 12).

[23] Kyle Chayka. *The Very Real Value of Gaming's Virtual Economies*. Aug. 2013. URL: http://www.psmag.com/navigation/business-economics/the-real-value-of-virtual-economies-eve-world-of-warcraft-64593 (cit. on pp. 4, 15).

[24] Aaran Fronda. *Virtual worlds and broken economic models*. Aug. 2013. URL: http://www.theneweconomy.com/technology/virtual-worlds-and-broken-models (cit. on pp. 2–4, 6).

[25] Dan Hart. *Balancing Your Game Economy, and profiting from*. Oct. 2011. URL: http://www.gdcvault.com/play/1015151/Balancing-Your-Game-Economy-Lessons (cit. on p. 15).

[26] S. D.. Kaehler. *Fuzzy Logic Tutorial*. Encoder: The Newsletter of the Seattle Robotic Society. URL: http://www.seattlerobotics.org/encoder/mar98/fuz/flindex.html (cit. on p. 23).

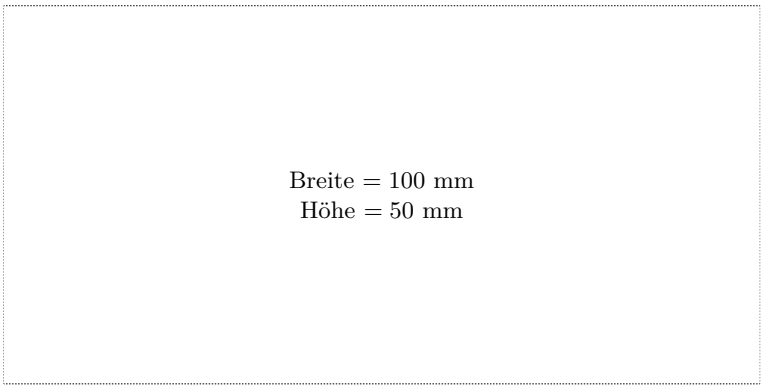
[27] Adi Robertson. *Blizzard shutting down 'Diablo III' auction house, wants players to get back to killing monsters*. Sept. 2013. URL: http://www.theverge.com/2013/9/17/4742120/blizzard-shutting-down-diablo-iii-auction-houses (cit. on p. 5).

[28] Václav Slavíček. *FuzzyFramework*. Jan. 2011. URL: http://www.codeproject.com/Articles/151161/Fuzzy-Framework (cit. on pp. 23, 24, 39).

[29] Europäische Zentralbank. URL: http://www.ecb.europa.eu/ecb/educational/hicp/html/index.de.html (cit. on p. 12).

Messbox zur Druckkontrolle

— Druckgröße kontrollieren! —



— Diese Seite nach dem Druck entfernen! —