Master Thesis

Rivalry AI: A Motivating and Socially Interactive Non Playing Character

Masterarbeit zur Erlangung des akademischen Grades

„Master of Science“

Verfasserin/Verfasser : **Alexander Cerny, BSc**

Vorgelegt am FH-Masterstudiengang MultiMediaTechnology, Fachhochschule Salzburg

Begutachtet durch:
Wolfgang Litzlbauer, MSc

Salzburg, 24.11.2014

Eidesstattliche Erklärung

Hiermit versichere ich, Alexander Cerny, geboren am **16.03.1990** in **Zell am See**, dass ich die Grundsätze wissenschaftlichen Arbeitens nach bestem Wissen und Gewissen eingehalten habe und die vorliegende Masterarbeit von mir selbstständig verfasst wurde. Zur Erstellung wurden von mir keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.
Ich versichere, dass ich die Masterarbeit weder im In- noch Ausland bisher in irgendeiner Form als Prüfungsarbeit vorgelegt habe und dass diese Arbeit mit der den BegutachterInnen vorgelegten Arbeit übereinstimmt.

Salzburg, am **24.11.2014**

_____	_____
Vorname Familienname	Personenkennzeichen

Zusammenfassung

Diese Masterarbeit behandelt, wie es möglich ist, ein sozial interaktives Wesen zu erschaffen, das eine Rivalität aufbaut, wenn zwischen zwei Charakteren ein Konflikt entsteht. Zu Beginn werden die vier Emotionstypen vorgestellt, die man in einem Spiel ohne Handlung empfinden kann. Weiters werden psychologische Theorien zu aggressivem Verhalten und Wettkampfsituationen behandelt. Aufbauend auf diesen Theorien werden Ziele für die Rivalry künstlicher Intelligenz (KI) gesteckt. Danach werden gängige Methoden zum Erstellen eines sozial interaktiven Wesens verglichen. Um ein solches Wesen für die Rivalry KI zu erschaffen, wird das µ-SIC System verwendet. Dieses System wird weiters genutzt, um den emotionalen Zustand und die einzelnen Beziehungen zu den Kontrahenten vom Spieler für den Computer verständlich abzubilden. Das erstellte Konzept der Rivalry KI wird im Zuge der Masterarbeit in ein bestehendes Spiel implementiert. Dabei handelt es sich um das Spiel BareFoot, das in dem Genre Fun Racer angesiedelt ist. Als Grundlage für die Implementierung der Rivalry KI in das Spiel wird eine gängige Methode vorgestellt, mittels welcher KIs für Rennspiele erstellt werden können. Zusätzlich wird erklärt, wie eine Rennstrecke automatisch geparst wird, damit die KI darauf fahren kann. Danach wird die Struktur der implementierten Rivalry KI erklärt. Abschließend wurden User Tests durchgeführt, um zu überprüfen, ob die gesetzten Ziele der Rivalry KI erreicht wurden. Die Tests haben ergeben, dass die Ziele mit Einschränkungen erfüllt wurden.

Schlüsselwörter: Rivalry KI, Personalized Games, Socially Interactive Characters, Player Modelling, µ-SIC System, Emotionen, Aggressives Verhalten, Wettkampfsituation, Motivation, Racing AI, Track Parsing

Abstract

This thesis describes how to create a socially interactive character, which engages into a rivalry with another game agent, if a conflict developed between the two of them. The thesis covers a summary about the four key emotions without story in games and introduces the psychological theories of aggressive behaviour and competitive situations. These theories has been used as foundation to set up goals, which the Rivalry Artificial Intelligence (AI) should accomplish. Further different techniques to implement socially interactive characters into games are discussed. To provide a socially interactive character for the concept of the Rivalry AI, the µ-SIC System is used and adapted. This system is also used to model the player's emotional state and relationships. The concept of the Rivalry AI will be implemented in an existing game. The game is called BareFoot and is a fun race. To be able to understand further implementations of the Rivalry AI into the game, a common method to create a racing AI is introduced. Additionally to the race track representation for an AI, the thesis briefly covers an approach to automatically parse a race track for a racing AI. Afterwards the architecture of the Rivalry AI is presented and further describes how it was implemented into the game. To evaluate if the Rivalry AI accomplished the prior set goals, user tests were conducted and the results confirmed or at least partially confirmed the set hypotheses of the Rivalry AI.

Key words: Rivalry AI, Personalized Games, Socially Interactive Characters, Player Modelling, µ-SIC System, Emotions, Aggressive Behaviour, Competitive Situations, Motivation, Racing AI, Track Parsing

Acknowledgements

First, I would like to thank my thesis supervisor, Wolfgang Litzlbauer, for his guidance, support and ideas during the process of creating this master thesis. Next, I would like to thank my family and friends for supporting me during my graduate education. Thank you all so much.

Table of Contents

1. Introduction

In the game industry there is a trend to adaptive and personalized games. From scripted ally Non Player Characters (NPC) in World of Warcraft (Blizzard 2006) over No One Lives Forever (Sierra 2006) to the Forza's Drivatar (Playground Games 2014) use adaptive or personalized Artificial Intelligence (AI). The Drivatar is the recent example of personalization in commercial games. The racing AI is able to adapt and learn the driving skills and styles from the player.

Bakkes, Tan, and Pisan (2012) have a definition for the difference between adaptive and personalized AI. It is called strictly adaptive, when game adaptions are made without being informed by the actual player in one way or the other. If the adaptions are informed by the player, we refer to the game as being personalized. For example if the game's difficulty is dynamically adapted to the player, the game is called personalized. Bakkes, Tan, and Pisan (2012) are providing an overview of the field of personalized games. They summaries and cover the motivations and the required components for personalized games.

Bakkes, Tan, and Pisan (2012) also cover the psychological foundation for the motivation to create personalized games. Studies show that subjects in a high personal relevance condition will exhibit stronger emotional reactions (Darley and Lim 1992). Emotional theory states the hypothesis that emotions will have substantial and measurable effects, when the situation is a personally or socially significant one (Izard 2009). One possible way of personalizing a game is to create socially interactive characters. Socially interactive characters simulate emotions for NPCs and Player Characters (PC). Additionally they are able to represent the relationships between characters. The goal of this master thesis is to create a Rivalry AI, which is able to socially interact with other fictional characters, regardless if they are NPCs or PCs. According to the social interactions and the relationships the AI should be able to start a rivalry with another player. The Rivalry AI will be affected by the player's interactions and can therefore be considered as personalization of the game. The goal of the Rivalry AI is to create an opponent, who starts a rivalry, if the game progress leads to a conflict between the two characters. The game should continuously model the player's current emotional state and adapt the AI according to these emotions. Player modelling is a sub domain of the personalization field.

There are existing concepts to create a socially interactive AI: The model of personality and emotion (Elliott 1994), personality-based adaption of game agents (Cheng and Tan 2007), believable agents with narrative intelligence (Stern 2002) and socially interactive non player characters (MacNamee and Cunningham 2003). These concepts offer solutions to create emotionally characters.

Another possible purpose of the Rivalry AI is to create an opponent who starts challenging and distracting the player, when this user lost the chance to win the main goal of the match. The main goal could be to win the Grand Prix ranking of a racing game. A sub goal would be to win one race of the Grand Prix. The rival should build up a competitive situation and give the player a new goal to beat the Rivalry AI. This should give the user a reason to keep playing the game.

As part of this master thesis the concept of the Rivalry AI will be implemented in an existing game. The game is called BareFoot and is a online multiplayer Fun Racer. The business model of the game is free to play. With an estimated sales volume of 1.4 billion in the United States in 2014 the free to play sector is a profitable market („North America Digital Games Market Report 2014“ 2014). Successful free to play games require a huge active user base. It needs some time to build up such a huge community for a game. The AI is a possible substitute as an opponent, until there are enough active user.

To bypass the time without many active user an AI is needed, which entertains and motivates the users. Lazzaro (2004) run a study on how to create emotions without story. These findings are the foundation to entertain the user with the Rivalry AI. Additionally the psychological basics for aggressive behaviour and competitive situations are reflected, to find ideas to motivate the player via the Rivalry AI. According to be able to implement the Rivalry AI into a racing game, the basics for a race track representation and a racing AI have to be introduced.

In context of this master thesis a user test is run to evaluate, if the Rivalry AI is remembered after a race and if the rival sticks out compared to the other NPCs. Further the performance of the player modelling and if the Rivalry AI benefits the game are discussed.

This thesis will start with an introduction of hypotheses, which the Rivalry AI should fulfil. The basics of possible emotions in games to entertain the user are presented. Afterwards the fundamentals of the psychological general aggression model is summarised, to explain how the AI can trigger aggressive behaviour. Additionally the psychological background for competitive situations is examined and extended to motivate the player through the Rivalry AI. Finally possibilities are introduced to make the Rivalry AI recognizable.

State of the art concepts are discussed, which are able to create socially interactive characters. Afterwards the selected µ-SIC System (MacNamee and Cunningham 2003) is introduced and extended to fit the requirements of the Rivalry AI. Additionally the thesis covers, how the rivalry is triggered considering the current mood state and relationships of the character. The thesis describes different possibilities to interact with the mood system and the relationship model.

Afterward the game is introduced, in which the concept of the Rivalry AI will be implemented. Before going into detail of the Rivalry AI implementation, the basic techniques to represent a racetrack for an AI and how to create a racing AI itself are summarised. Additionally a brief introduction is given, on how to automatically parse a newly created race track. Further the architecture is presented, which is used for the Racing Rivalry AI. The events are described, which can affect the character's mood and relationships. Thereby methods to annoy and cheer up the opponent are presented.

The last part are the user tests, which evaluate the Rivalry AI. The results should prove whether or not the previous hypotheses concerning the Rivalry AI are accomplished. The user tests consist of an A/B test, where one test group played with the enabled Rivalry AI and the other group with the normal racing AI. Afterwards the results are discussed and interpreted. As a conclusion the advantages and disadvantages of the Rivalry AI are analysed and possible adaptions, extensions and improvements are presented.

2. Rivalry AI

The goal of the Rivalry AI is to build a better opponent to entertain the player. To accomplish this task, we have to clarify, what makes video games fun and what motivates a user to keep playing a game. Thus gives a foundation to create ideas for the Rivalry AI.

2.1. Emotions in Video Games

Lazzaro (2004) conducted an extensive research study to find out, what other factors than story triggers emotions. They video recorded 45 hours of users playing various video games and analysed, what the participants said and did. Further they evaluated the player's questionnaires and interpreted their verbal and non-verbal emotional cues during play.

The study determined, there are four keys to more emotions in games without a story. Each key is a reason why people play video games. The four keys are: Hard Fun, Easy Fun, Altered States and the People Factor. Analyses showed that highly successful games meet at least three out of these four keys. These keys combined make a deeply enjoyable game for a wide market. For the Rivalry AI only three keys are considered and the Easy Fun category is ignored. The Rivalry AI's goal is to be able to fulfil and trigger the remaining three keys.

2.1.1. Hard Fun

The first key is Hard Fun, which results from challenges, strategies or puzzles. For many games and players this is the main reason to play games and to enjoy them. The player has to overcome challenges. The user runs through different learning stages to accomplish a goal. The failing, learning and accomplish phases create emotions such as frustration and Fiero. The term Fiero is Italian and means personal triumph (Eckman 2003). The player receives continuously rewards through the game progress and his own success. In a jump and run game this would be the mastering of hard jumps and in a puzzle game this would be the solution to a tricky riddle after many failing attempts. Users, who enjoy Hard Fun, love to test their own skill and to accomplish something. Games with this key offer balanced game difficulty for different skilled users. The Rivalry AI can trigger Hard Fun, when the player triumphs over the AI. It is possible to establish frustration over a mid-term relationship, where the Rivalry AI wins sub goals or side quests.

The study by Lazzaro (2004) observed seven different emotions. The Rivalry AI should trigger mainly frustration, Fiero and Schadenfreude. Schadenfreude is a German word and expresses the joy over misfortune or failure of an opponent. User experienced the tension of frustration

followed by the feeling of victory as an exciting gameplay element. Competition is an important factor when playing games. Especially the emotion Schadenfreude appears, when observing players compete in games against others (Lazzaro 2004).

2.1.2. Altered States

Players enjoy changes in their internal state, when they play games. They like to be influenced in their behaviour, thoughts and perception. They like to move from one mental state to another, while playing a video game. They want to feel something different. (Lazzaro 2004) This can be accomplished with the Rivalry AI by sending the player on an emotional rollercoaster. The player starts the game excited. Over time the rivalry is established. After some time the rival frustrates and annoys the player and keeps driving him insane. After an ongoing competition the NPC starts to cheer up the player by making mistakes. The AI alternately switches between triggering frustration and joy. In the end the AI should lose and bring the player to a tight victory, leaving him in an emotional state of joy, Fiero and satisfaction.

2.1.3. The People Factor

The observation from Lazzaro (2004) brought up, that people enjoyed playing with others. Multiplayer games enhance by better player to player interactions. For example people keep playing games, even if they don't like them, to spend time with their friends. The social interaction during a game is an important factor. Cook (2009) states, when friends compete against each other, mutual smack talk is a form of bonding. In order to include smack talk to the Rivalry AI, each AI character needs a recognizable voice, which is played at suitable occasions. For example when the player is attacked, the attacking AI should play a mocking or insulting sound. On the other hand when the NPC is attacked by the player, it should start swearing. By integrating smack talk to the Rivalry AI, the player's aggression can be increased.

2.1.1. Easy Fun

In order to provide a complete overview of the four key emotions, the last remaining key emotion is briefly summarised. Easy fun is the excitement of the sheer enjoyment of the game activities. The player likes to explore and exploit the virtual game world. This emotion isn't linked to a winning condition. The player likes to become immersed in the experience of the game. The repetition and rhythm of the gameplay can become hypnotic. Emotions trigger by this category are curiosity and wonder. Example genres for this type of fun are open world or Zen games. (Lazzaro 2004)

2.1.2. Motivation in Sport

Other research fields were considered, which were concerning motivation for competitive situations, to find ideas for the Rivalry AI. The most promising field therefore is motivation in sports. Research shows that athletes in sport are motivated by two main types of motivation (Vallerand, Deci, and Ryan 1987). The first one is intrinsic motivation, this is when someone plays out of joy and fun. The second one is extrinsic motivation, which means playing for benefits like trophies, prestige or glory.

Whereas extrinsic motivation can be achieved by video games with achievements, tournaments or high score lists, an AI opponent isn't predestined to fulfil this motivator. On the other hand intrinsic motivation is in reach for the Rivalry AI.

Vallerand and Losier (1999) claims that the social environment has effects on the motivation. Research found social psychological factors, which influence one's motivation. Vallerand and Losier (1999) point out three factors: success/failure, competition/cooperation and the coach's behaviour towards the athlete. The Rivalry AI only focuses on the first two factors. Especially in games with sub goals or tournament like competitions is it possible to motivate via success and failure. This can be accomplished by letting the player and his rival alternatively win. The competition as motivator is an important topic for the Rivalry AI. This topic will be covered in the following chapter Aggression and Competition.

During a competitive game a player often focuses on beating the opponent and not on the main task itself (Vallerand and Losier 1999). This is one possible field of application for the Rivalry AI. When a player loses touch with the leading group and has no chance of winning, the AI should establish a competition between the user and the AI. When focused on the competition or rivalry against an opponent, the user should lose focus on the main goal. Further he will start concentrating, hating and beating his new rival. This gives the user a new goal and therefore motivation to finish the game. This aspect meets two theoretical motivators: Hard Fun and motivation through competition.

Another similar reason, why the game benefits through the Rivalry AI, can be proven by the testosterone level. When people play against each other, the losers testosterone level decreases. The losing player attempts to avoid fighting the same opponent. In games where only a few people can be winners, like in a racing game, the player's retention will suffer (Cook 2009).

Therefore the Rivalry AI can help motivate inexperienced users, who often have a hard time starting an online multiplayer game. With the Rival AI the players don't have the goal to win against the experienced users, but to triumph over their rivals. This keeps them motivated until they are experienced and skilled enough to compete against human opponents. Thus increases the retention rate of new users, which is one of the key figures in free to play games.

Remaining problems are how the AI can establish a rivalry and how to force the user to hate the NPC. Therefore basic understanding of emotions, aggression and competitive situations is necessary.

2.2. Aggression and Competition

In order to be able to build up and trigger certain emotions with the Rivalry AI, the term emotions has to be defined. Elliott (1994) provides a suitable definition for emotions.

> *"What we refer to as 'emotions' in this paper arise naturally in many human social situations as byproduct of goal-driven and principled (or unprincipled) behaviour, simple preferences, and relationships with other agents. This applies to many situations that one would not ordinary refer to as emotional: a social agent becoming annoyed with someone who is wasting her time (a mild case of that person violating the agent's principle of social efficiency thus blocking of one of the agent's goals through the reduction of a valued resource, time) […]. "*
>
> (Elliott 1994)

This definition states that an interaction between two characters creates emotions, if the principles of one character are violated and further conflicts with the character's goals. An example for a situation, which creates an emotion according to this definition, would be in a fun racer, when the character gets stunned by a power up attack. The stunned agent's goal to win the race is violated. Therefore his sensation can be classified as emotional.

In order to create an exciting opponent, a definition for the term excitement is required. One possible explanation could be physical arousal. Anderson et al. (2010) states that aggressive behaviour increases the likelihood of creating a higher level of physiological arousal. In other words the player gets excited, when he gets physiological aroused.

Another point of view to determine the excitement of a player is the testosterone level. Playing with strangers affects the testosterone level of winners and losers. The winner's testosterone increases and in order his dominance and aggressive behaviour increases. Thereby increases

the physical energy and people can become aroused (Cook 2009). Therefore winning is exciting.

Aggression increases the physiological arousal. The problem with this approach is, that this arousal can turn into violent behaviour, when it exceeds a certain degree of aggression (Anderson et al. 2010). To control the intensity of the aggression an understanding of the fundamentals of aggressive behaviour is required.

2.2.1. General Aggression Model

The General Aggression Model (GAM) has been created to provide a simplified overview of the common elements among prior models of the development of aggressive behaviour (Anderson et al. 2010).

The GAM focuses on one cycle of an ongoing social interaction of a person in a certain situation. There are three main foci of the cycle. The first uses a person and a situation as inputs. The second are cognitive, affective and arousal routes, which the inputs have to pass. The third are the outcomes of the underlying appraisal, which is passed in the decision making processes. (Anderson and Bushman 2002) This model explains that the intensity and outcome of the aggression depends on the person's personality, current mood and the current situation. The input has to run through certain routines and structures to appraise it. The structures depend on the person and the current situation. Afterwards the decision system is able to perform a choice of action.

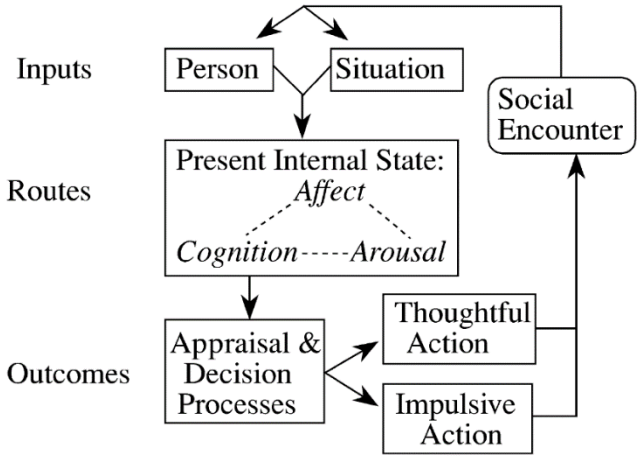

Fig. 1: The general aggression model episodic processes (Anderson and Bushman 2002)

2.2.2. Script Theory

There are a few social-cognitive models of aggression, which cover the cognitive, affective and arousal routes of the GAM in detail. Those are for example the Cognitive Neoassociation Theory, Social Learning Theory and Script Theory (Anderson and Bushman 2002). Script Theory states, when children observe violence in media, they learn aggressive scripts. It memorizes a connection between the situation, the roles participating and the actions, which were taken. Once a script is learned, it can be retrieved at some later point and used as a guide for behaviour. (Anderson and Bushman 2002) For example a child, which has seen several thousand times to settle a dispute by using a gun, will have a very accessible script stored in its head. This model is needed as a basis to understand and to establish competitive situations, which are handled in the following chapter. The Script Theory is then used to build up aggression in the player, by using accessible scripts for competitive situations.

2.2.3. Facilitate Aggression Variables

There are environmental variables, which facilitate the emergence of aggression, like insult or pain. An example for insult through an NPC is the prior mentioned smack talk. The effects of aggression are categorized in short-term and long-term effects. The long-term effects appear, when a human is exposed over months or years to violence. These effects contain permanent changes in beliefs, expectancies, scripts, attitudes and other related personal factors. (Anderson et al. 2010) Those effects must be avoided at all costs, when creating a Rivalry AI.

Short-term effects appear, when a person plays a video game for a brief time like 15 minutes (Anderson et al. 2010). Those effects use the previous mentioned Script Theory. The player has a social interaction and the resulting reaction is chosen depending on existing knowledge structures, such as various types of schemata and scripts. (Anderson et al. 2010)

Many situational factors can increase arousal and anger, even in nonviolent video games. For example racing games or sports games can increase heart rate and blood pressure. Unbalanced games, which are too difficult, can increase frustration and anger. The physiological arousal instigated by a video game is fairly short persistent and are less likely to leave long term traces in the brain. (Anderson et al. 2010) This is the physiological arousal, which the Rivalry AI should trigger. Because of these consequences, the Rivalry AI shouldn’t use explicit violence or too difficult and unbalanced game parts to frustrate the player. Other possibilities must be found and considered to engage the player in aggressive behaviour, such as competitive situations.

2.2.4. Competition

Anderson and Morrow (1995) state that competition leads to aggression. The classical Robber's Cave studies (Sherif and Sherif 1953) is about two groups of boys in a summer camp, competing against each other. The study shows that competitive games can break up initial friendships, create hostilities and induce aggressive behaviours. Nelson, Gelfand, and Hartmann (1969) found that 5 to 6 year old children show more aggressive behaviour, when participating in a competitive contest, than after a non-competitive contest. Even though, when they have won the most games of this contest, they still show aggressive behaviour. This supports the hypothesis that Rivalry AI could be established as a mid or long-term relationship. The player feels still aggressive towards the rival, even if he has won a lot of sub goals during the match. An example for a sub goal is to win a race in a Grand Prix of a racing game. The player and the AI can alternatively win races. The main goal is to be victories in the Grand Prix.

There are two factors to determine a competitive game (Berkowitz 1962). First there have to be two or more participants reaching for the same reward. Second, the participants stand in conflict to each other to receiving this reward. *"Competitive encounters are at least partly frustrating as the contestants block each other's attempts to reach the disputed goal. "* (Berkowitz 1989, 66)

As mentioned above in the Script Theory a human being develops structures and scripts. For example one script defines to develop aggressive behaviour in competitive situations. As soon as this script exists in the head of players, they act aggressive in a certain situation, because they interpret the situation as competitive. On the other hand in cooperative situations, players built structures that lead them to less aggressive behaviour. When someone has built such structures, it is a purely cognitive effect to react aggressive in competitive situations. (Anderson and Morrow 1995)

As stated prior, when describing the Script Theory, people start to build such structures when they are children. Therefore it should be possible for the Rivalry AI to establish and simulate competitive situations to initialize aggressive behaviour for adult players. When a player defines a situation as competitive, he will react aggressively (Anderson and Morrow 1995). Competitive situations don't need an interpersonal interaction in order to produce aggressive behaviour. The player simply has to define the situation as competitive. (Anderson and Morrow 1995) Therefore it is possible to create competitive situations by the Rivalry AI, although it isn't a human being.

2.3. Concept of Rivalry AI

As a summary after this excursion in psychological fields, the goal is to create a Rivalry AI, which is able to entertain and motivate the user. A rivalry between a player and a NPC should build up slowly over time. When the player starts to engage into a rivalry, the AI should start to establish a competitive situation. This is possible by sticking close to player and starting to win sub goals and side quests.

As soon as the player interprets the situations as a competitive one, he develops an aggressive behaviour, which leads to physiological arousal. That means the player gets excited. By letting the player and the Rivalry AI alternatively win sub goals, different feelings of the player are triggered. When the NPC beats the user in a sub goal, the player gets frustrated, when the NPC makes a mistake, the player feels Schadenfreude. When the user beats his rival in a sub goal, he feels Fiero. With this concept the player experiences Altering States, which was one of the four previous mentioned key emotions.

The player should get frustrated when engaged into rivalry by using so called environmental variables, mentioned prior in cognitive psychology. This means insulting, provoking or attacking the player. How this part will actually look, depends on the game, for which the Rivalry AI will be created. For example in a hack and slay game the AI can start a lot of attacks and force the player to lose a side quest. In a real time strategy game with multiple enemies the NPC could help another opponent to defeat the user. After winning a side quest the rival could start smack talking. There are a lot interactions the player can use to annoy and facilitate the user's aggressive behaviour.

When the Rivalry AI tries to systematically annoy and frustrate the user, it is important to be extremely careful to not cross the player's line. Therefore the rival has to cheer up the user once in a while. Again interactive events from the game are used, so the rival can amuse the player. One way is to let the AI make vital mistakes. For example in a jump and run where the NPC is chasing the user, the AI could jump off a platform too early and fail to get over a difficult gap. This action triggers Schadenfreude in the player.

The player is sent on an emotional rollercoaster by the Rivalry AI. First tension is built up. This built up tension should be released in the end, when the player wins the main goal and beats the NPC. Thus transforms the tension into sensations of satisfaction, happiness and joy, simply called Fiero.

As mentioned previously, both in the field of competitive psychology and motivational factors in sport, a rivalry can be a motivator. The player forgets about the main goal and just concentrates on beating the rival. This is especially motivating in games, where there are only one or a few winners per match. As mentioned above, players don't want to play against people they have lost in the past. By creating a competitive situation the Rivalry AI establishes a new goal for the player, which is to beat the new competitor. Before the rivalry there were only one or a few possible winners to triumph chasing the main goal. By establishing new goals for the players, the Rivalry AI creates more possible winners for the game. Inexperienced players get the chance to win against the rival instead of winning the main goal. Thus motivates inexperienced users. Even if the player loses against the Rivalry AI, it is possible to make it a tight victory. If the player has the sense that there was a chance to win, he will try again.

In order to build a Rivalry AI a competitive situation has to be established. As soon as the situation is interpreted as competitive, the rival can start to facilitate the user's aggressive behaviour. The NPC has to switch between annoying and cheering up the player. During this phase, tension should be built up and released in the end of the game by the player triumphing over the AI. This lets the player feel Fiero and motivates the user to play another match.

Williams and Clippinger (2002) found that users playing against the computer experienced higher levels of aggression than playing against a stranger. They conducted a user test. The participants had to play monopoly face to face against another participant. Afterwards they had to play a second time against a computer opponent. The subjects had to stop playing once in a while and answer a questionnaire about how aggressive they feel in the current situation. The results showed that users playing against computer opponents were significantly more aggressive than when playing against other users. As mentioned before the Rivalry AI tries to build up aggressive behaviour in the player, but it has to be avoided to annoy the player too much and make the user violent. Therefore a possible solution to reduce the aggression could be to humanize the computer opponents. In order to accomplish this task the Rivalry AI could be represented by portrait picture of a human being.

2.4. Character Recognition

To be able to build up a relationship between the player and a NPC, it must be guaranteed that the player is capable to differentiate the NPCs. The player has to be able to recognize each opponent. In many games it is possible to create a unique character for the AI. For example in Mario Kart (Nintendo 1992) there are eight unique characters and only one instance per character is allowed in the same race. It is easy to differentiate between these unique characters. In games where it isn't possible to use unique characters, it is necessary to use

other methods to provide a better character recognition. Possible methods are to use visual and audible material to make the Rivalry AI recognizable.

2.4.1. Avatars

A common technique to represent players in games are avatars. Avatars are considered as computer generated visual representation of people (Nowak and Rauh 2005, Schroeder 2002). Chan and Vorderer (2006) expanded this definition of avatars to defining avatars as characters that represent the users in a virtual environment and community. The origin of the term avatar refers to the Sanskrit language and means the god Vishnu's manifestations on earth (Castronova 2002).

To increase the chance of character recognition each NPC gets a unique avatar and a unique name to represent him in the game. This concept was already used in the game Micro Machines (Codemasters 1991), where the player could decide, who the opponents would be. Each character had a unique avatar and a name.

Fig. 2: Avatars and names of the NPCs in Micro Machines (Codemasters 1991)

Users consider predefined game requirements, when they design an avatar for a game (Trepte, Reinecke, and Behr 2009). This means players can be expected to show stronger tendency to create avatars with features very close to the game requirements (Trepte and Reinecke 2010). A study was conducted to prove this hypothesis. Users had to create avatars after reading seven different game descriptions (Trepte, Reinecke, and Behr 2009). For example users created more manly avatars, when the game description contained pre tested male characteristics and vice versa for pre tested female characteristics, regardless to the user's gender.

The requirements for the Rivalry AI are to trigger aggression and establish competitive situations. Therefore we decided to create grim and mean looking avatars for the AI, which represent the characters for a competitive game. In order to create the avatars, pictures of the development team were altered. These images were transformed into cartoon illustrations of certain stereotypes like a captain, an army sergeant or an agent. Each stereotypical avatar got a corresponding name. Additional to increase the chance of recognition, each character got a different plain-coloured background.

Fig. 3: Avatars designed for the game BareFoot

The visual representation of the NPCs consists of stereotypical characters, suitable names and a defined background colour. These components should increase the chance to recognize the character.

2.4.2. Verbal Feedback

In order to boost the chance of character recognition and to heat up the relationship between NPCs and users, the NPCs get a set of voice recordings for certain situations. In Mario Kart the characters have audible reactions to different game situations. For example when the character overtakes an opponent, a positive sound like laughter or a catch phrase is triggered. On the other hand when something bad happens, a negative sound is triggered that implies anger and frustration.

Additionally to these events the Rivalry AI should include smack talk. As mentioned before smack talk is a form of bonding, when playing games (Cook 2009). Suitable situations for smack talk is the start of the game or a sub goal and right after the game or a sub goal ended. To increase the chance of recognizing the character, it is important that each voice file from one character is recorded from a single person. Thus makes it possible that the user builds a connection between the voice and the character's avatar and name. In order to decrease the aggressive behaviour of the player towards a NPC, the NPCs should be humanized. To accomplish this goal the avatars use pictures and voices of real human beings.

The base concept of the Rivalry AI is constructed. The next problem is, to start the rivalry according to the emotions of the player. It is planned to engage in rivalry after a certain period of time. The AI shouldn't start a rivalry randomly. As stated before, a rivalry builds up on emotions. Therefore the rivalry should only start, when two players built up a conflict between them. At some point the AI should start to explicitly fight against the player. To accomplish this goal, the NPC's mood has to be simulated, therefore a socially interactive character is required.

3. Socially Interactive Game Agents

Imagine you are playing a Grand Prix in Mario Kart. In the first level you are constantly switching position with Toad. He beats you in a photo finish. The rubber band technique, where the maximum speed of the leading character is limited, manages that Toad sticks around your character the whole race. He attacks you in the worst moments possible. He throws you off cliffs, stuns you in front of an important jump or blocks you along a narrow path. In the third race your single remaining goal is to crush this tiny little mushroom. Every single power up you collect, will be used to make the rest of his pathetic life to hell on earth. But all he does is keep trying to compete in the race and pretending nothing ever happened in the prior races.

The idea of a Rivalry AI is to create game agents, which adapt to the current mood of the player. A game agent is defined as every fictional character in the game world. It can be either a Player Controlled (PC) character or a Non Player Character (NPC). When the player is engaging into a rivalry with a NPC, the AI should provoke and bother him, until the PC is getting upset by these ongoing annoying events. Afterwards the rival should start to cheer up the player by attacking other opponents or start making mistakes like driving too fast into a curve and falling off a cliff. The Rivalry AI should put the player on an emotional rollercoaster, whereby the game should benefit.

To assure such a behaviour a system is required, which is able to gather the player's current emotional state. A model is needed to simulate socially interactive NPCs and a technique to transform virtual actions and events into emotional sensations. The system should be simple to implement, because it's used for a casual game with a few limited emotional events. Additionally it should be easy to understand to enable non-technical people like game designer the planning and maintaining of the AI (MacNamee and Cunningham 2003).

Suitable concepts are a shallow model of personality and emotion (Elliott 1994), personality-based adaption of game agents (Cheng and Tan 2007), believable agents with narrative intelligence (Stern 2002) and socially interactive non player characters (MacNamee and Cunningham 2003).

3.1. Related Work

Elliott (1994) discussed the problems of a system for agents to be capable of emotional interaction with users. Based on the discussion a model was created, which consists of a set of appraisal frames, which represents the character's individual goals, principles, preferences, and moods. Further it has a set of channels to express emotions. The appraisal frames are

used to interpret the situation and the channels consider the interpretation and react accordingly. In return the reaction is observable by other agents. The model considers seven emotion categories: anger, hatred, sadness, love, joy, fear, and neutral.

The system uses a modified set of emotion intensity variables, based on the work of Ortony, Clore and Collins (1988) and modified by Elliott (1992). The intensity of an emotion can be determined by three parameters: The short-term state of an agent, the long-term disposition of an agent, or the emotion-eliciting situation itself.

One goal of this approach was to work out a detailed computational method for representing the antecedents and expression of human emotion (Elliott 1994). This is a suitable method to simulate complex emotions. However considered in casual game aspect, it is a far too complex system. Using seven emotion categories and three different states, which influence the intensity of each emotion, would be an overkill for game designers and programmers to maintain. Especially for a casual game which has only a limited set of actions and decisions, like picking up a power up or attacking an opponent. These are only a few social interaction possibilities and therefore such a complex system would be inappropriate. Whereas it is an adequate model for simulations with complex emotional character like The Sims (EA Maxis 2002).

The next model, which fits our criteria, is a technique with a personality adaption module (Cheng and Tan 2007). This module allows adjustments to individual styles and strategies of different players. The adaption is decoupled from action selection and created thereby a modular adaptive system. It is possible to use this module for either a player PC or a NPC. The model is designed for NPCs to adapt their teamwork skills according to the PC. However the framework can be easily extended to be adaptive referring to opponents instead of team members and thereby suitable to create the Rivalry AI.

This technique makes it possible that a game agent adapts according to actions of other agents, regardless whether they are PCs or NPCs. The agent’s personality is defined as a set of allowable actions. Each actions is tagged with a value between 0 and 1. The personality is actually a function, which returns a value for each action. These values are used by an action selection mechanism, which determines the chance to choose that action. The personality is therefore a combination of different action values. These values change constantly over time, which results in different action outcomes.

The concept is comparable with a fuzzy logic decision making system. In aspect of the Rivalry AI, where the game agent should start the rivalry at a certain moment during the simulation, it

is improper to constantly change the decision making process. Another problem is that the personality is encapsulated in the decision management. One goal of the Rivalry AI is to create recognizable character and therefore the NPCs should have rich personalities, which remain the same during the game. The last issue of this concept is, the model is designed to adapt to individual player styles and strategies, which are non-existing aspects for the Rivalry AI.

The third approach is from a commercial project. The game Virtual Babyz (PF Magic 1999) tried to create an interactive narrative experience. Similar to the more famous game Tamagotchi (Bandai 1996), in which the user has to play, feed and raise babies. One goal of the game is to provide a powerful interactive narrative experience, which should be built up according to emotions and relationships between the characters (Stern 2002). A baby can socialize with both, other NPCs and the players. Therefore the Babyz' characters needed rich personalities, emotions and the ability to express themselves through actions and behaviours. Virtual Babyz uses a behaviour-based architecture with a model of personality and emotions. Each character has a persistent fuzzy association matrix memory, which expresses the positive or negative feelings to other babies or the player. The memory is constantly updated by actions, which take place in the virtual world, accordingly to their unique personalities. A drama manager decides based on the current memory, what the NPC should do next.

This model meets a lot of the requirements for the Rivalry AI. First it is possible to set up personalities. Second the emotional state of the game agent is stored. Next it keeps track of the relationships between virtual characters. The chosen actions consider the current mood, the relationship to other game agents and are goal orientated. One minor problem is that the emotional state is encapsulated with the individual relationships.

The last concept is the µ-SIC System, which is used to simulate personalities, moods and relationships of NPCs and chooses the character's behaviour according to the current simulated parameters (MacNamee and Cunningham 2003). This approach evaluated a number of psychological solutions to model a character's persona and selected sufficient solutions in considerations for computational purposes. This is another concept, in which it is possible that the game agent interacts with both, other NPCs and PCs as well. It is split up in three modular parts: personality, mood and relationship. The personality is predefined and will not change during the simulation. The mood is constantly updated and manages mood changes independent to who interacted with the game agent. The relationship module manages, who interfered with the character and if it affected the mood positively or negatively.

This system convinces on the one hand with its simplicity and on the other hand with scientific psychological background. It is simple to design and maintain during developing phase and can be easily understood by non-technical team members. Which is an important aspect in the game development industry because the AI is often developed by game designers (MacNamee and Cunningham 2003). It meets all criteria of the Rivalry AI with its modular architecture, predefined personalities, separated mood from relationship and it is possible to model the player's mood with this system. With this technique it is possible to gathers and models the current emotional state of the user. Therefore the µ-SIC System was chosen as the base concept, which should be adapted and extended for the Rivalry AI.

3.2. µ-SIC System

The µ-SIC System is designed to create socially interactive game agents. It uses psychological models to simulate the personality, mood and relationship of a virtual characters. These models use the game agent's social interactions in the virtual game world, which then drives the character's behaviour. The personality model defines, how stable or unstable and how introvert or extrovert a character is. It is possible to model personality types such as aggressive, sociable or moody. The mood model constantly changes depending on actions, which involve the character and occur in the virtual world. The relationship model also reacts to social interactions, which involve the character and another game agent. The Proactive Persistent Agent (PPA) architecture is used to manage these models.

The PPA architecture (MacNamee and Cunningham 2001) was developed for the purpose to overcome the limitations typically associated with creating NPCs. The architecture is used by µ-SIC system. It was created as part of the TCD Game AI Project (Fairclough et al. 2001).

3.2.1. Proactive Persistent Agent Architecture

Agents in the PPA architecture are proactive, because they are able to follow their own goals. It is persistent, because each agent is modelled at all times during the simulation. Even if the player isn't close to the character. (MacNamee and Cunningham 2003) You can see a schematic illustration of a PPA architecture in the figure below. The architecture has a modular structure and is made up of four key elements: Schedule, Role Passing System, Social Unit and Goal Based Planning Unit.

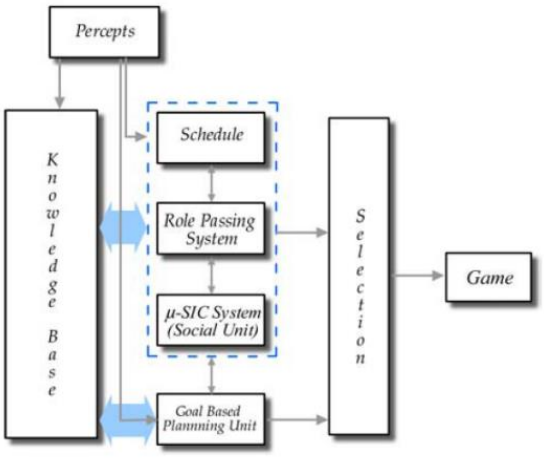

Fig. 4: Proactive Persistent Agent Architecture used in the μ-SIC System (MacNamee and Cunningham 2003)

3.2.1.1. Schedule

The NPCs which use the PPA architecture are persistent, as mentioned above. Therefore each character runs continuously a schedule, no matter if the player is close to the character or not. A schedule can be seen as the daily routine of a working person. He gets up in the morning, makes breakfast, drives to work, works, goes on a lunch break, returns to work and so on. Each state of the day is one part of the schedule. This is a very high level sequence and can be achieved without any intense computation. (MacNamee and Cunningham 2003)

3.2.1.2. Role Passing System

Depending on the current schedule state the AI has to make decisions in different situations. That means the schedule decides, which role the character is adapting. In other words the schedule module passes a role to the NPC (Mac Namee and Cunningham 2001). The character uses simple behaviours, which can be layered, to handle more complex situations. To put it in another AI topic context, the role passing system could be implemented using fuzzy logic. In this case a role would be a state, which uses certain fuzzy rules out of a basic set of fuzzy rules to decide, which action should be chosen. This module defines which rules have to be considered by the decision making progress. (MacNamee and Cunningham 2003)

To continue with the schedule example from above, the character would first adapt the wake up role. This could include turning off the alarm clock, standing up and getting dressed. The drive to work part could include to start the car, drive off and find the shortest path. Each role in this schedule would significantly change the behaviour of the NPC.

3.2.1.3. Social Unit

The social unit keeps track of all social interactions of all game agents. Furthermore it updates the inter-personal relationships between all game agents (MacNamee and Cunningham 2003). For example when someone is cutting off the character on his way to work or if a co-worker helps him solve a problem, the social unit keeps track of this interaction and with whom they were. The µ-SIC system is used as the social unit.

3.2.1.4. Goal Based Planning Unit

The goal base planning unit is used to pursuing long term goals. For example the long term goal of our previous mentioned character could be to survive. Therefore the game agent needs to drink water, when he is thirsty or has to eat once in a while.

3.2.2. Psychological Models used by the µ-SIC System

The µ-SIC System has taken existing models from psychology to model the NPC's personalities, moods and relationships. MacNamee and Cunningham (2003) considered three selection criteria to find suitable psychological models.

> *"The first selection criterion is that the models chosen need not necessarily represent the current state of the art in cognitive science. Our goal is to create characters which behave plausibly at all times within a simulation, so models which achieve this are enough.*
>
> *The second important criterion for model selection is that the models used should be as simple enough that the designer can understand how they work, and more importantly, how changing a model's parameters might affect a character's behaviour.*
>
> *In addition to the concern for usability, any system for use in games must be efficient both in terms of memory usage and the computation required. "*
>
> (MacNamee and Cunningham 2003, 2)

The psychological models, which meet these criteria, are the Eysenck's two dimensional classification (Eysenck and Rachman 1965) for the personality model, the Lang mood model (Lang 1995) to simulate the emotional state of the NPC and a relationship model, which has its psychological basis in (Wish, Deutsch, and Kaplan 1976).

3.2.3. Personality Model

The Eysenck model plots personality across two orthogonal axes. The introversion-extroversion values are plotted along the x-axis and the neuroticism-stability values along the y-axis.

This method makes it possible to create different personality types such as aggressive, sociable or moody. Psychologists generally agree that two axes are not enough to accurately model personalities, but referring to the selection criteria above, this model is sufficient enough for the µ-SIC System. Another reason to choose the Eysenck model is the fact that it is still one of the most respected and well established personality models in psychological theory (Lloyd et al. 1984). In following figure the Eysenck model is displayed with some example personality types plotted along the two axes. The personality is a predefined part of the social unit and shouldn't change over time.

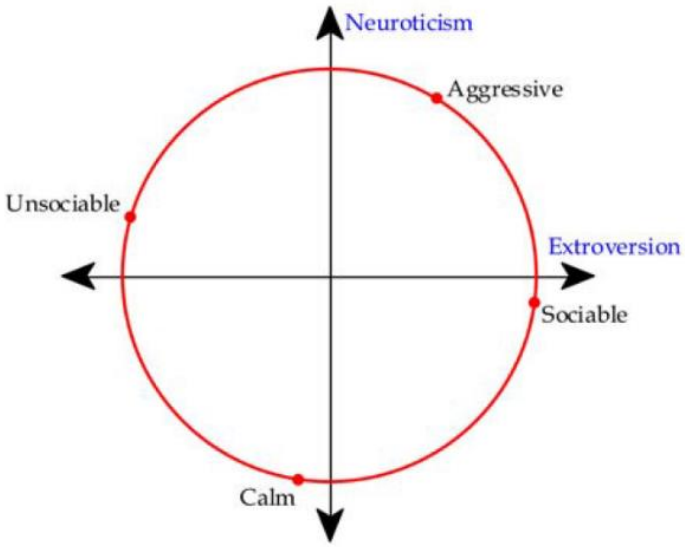

Fig. 5: The Eysenck Personality Model with example personality types (MacNamee and Cunningham 2003)

3.2.4. Mood Model

The mood of a character changes constantly. It is influenced by interactions with other characters or the virtual game word, but the mood isn't considered related to any other game agent. Similar to the Eysenck model, the chosen mood model consists of two axes. The model was created by Lang (1995). The axes are measured in valance and arousal. Valance simply determines whether the mood is positive or negative. The term valance stands in psychology for the emotional value associated with a stimulus. The arousal value refers to the intensity of the emotion. This mood model has been already used in computing applications by Picard (1995).

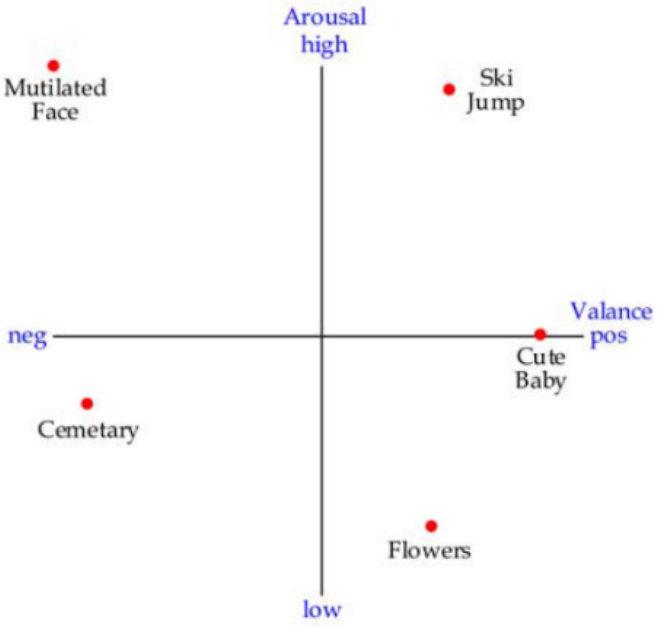

Fig. 6: The Lang mood model with example reaction values to pictures shown in a user test (MacNamee and Cunningham 2003)

3.2.5. Relationship Model

Similar to the mood model the relationship model changes constantly over time. The difference compared to the mood is, that each emotional event is associated and tracked to another game agent. This model is originally due to Wish, Deutsch, and Kaplan (1976). It has been used in a number of other entertainment projects: Oz Project (Reilly 1996) an interactive drama, TALE SPIN (Meehan 1976) an interactive program that writes stories and UNIVERSE (Lebowitz 1985) a program that generates melodrama plot outlines. There are four values to model the relationship of two game agents: like, attractedness, dominant and intimacy. The μ-SIC System extended this model by the fifth value interestedness.

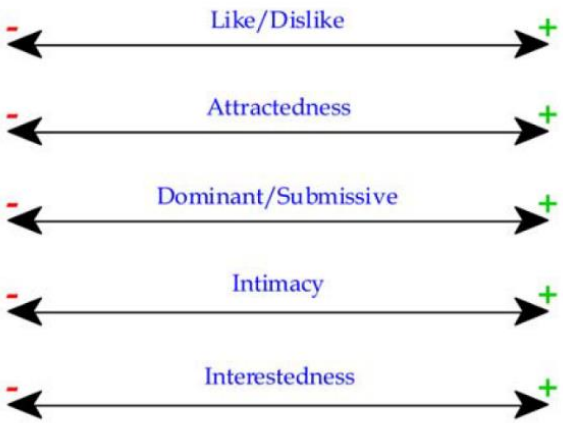

Fig. 7: The relationship model

The µ-SIC System is originally design for conversations between game agents. The five categories, in which the relationships get plotted, are created for topics of the conversation. For example if the characters are attracted to each other, they start to flirt, when they are intimate, they share personal secrets. If the characters aren't interested into each other, they won't start a conversation. For the Rivalry AI restrict these categories to three: Like/Dislike, Dominant/Submissive and Interestedness. How these categories are affected by actions will be covered in the implementation part of the thesis.

The µ-SIC System offers a simple approach to create a socially interactive character. The persona of a character is separated into three different modules. These modules are encapsulated in the PPA architecture. The system is a modular and can be easily implemented and maintained. It is possible to create game agents as a non-technical person.

3.3. Engage in Rivalry considering the Mood

By using the µ-SIC System it is possible to create a socially interactive game agent. The next question is how the µ-SIC System can be extended and modified to include the Rivalry AI. It is important to consider, when the Rivalry AI should be started. The Rivalry AI should adapt according to emotions. The question is in which emotional state someone would start a rivalry. As stated before, a rivalry is similar to a competition. A competition creates aggressive behaviour and aggressive actions are linked to a bad mood. An aggressive behaviour increases the physiological arousal, therefore the arousal should be high or at least be positive. A negative arousal implies that the character is bored. By considering these thoughts the rules to start a rivalry are that the arousal has to be positive and the valance has to be negative. When these rules are visualised in the mood model, which plots its mood values along two orthogonal axes, it turns out that the rivalry could only start, when the current mood is plotted in the second quadrant. The rules are extended by the fact, that the rivalry only starts, when a specific threshold is exceeded. The visualised result of this rule set is shown in the figure below. The marked area is called the rivalry trigger zone.

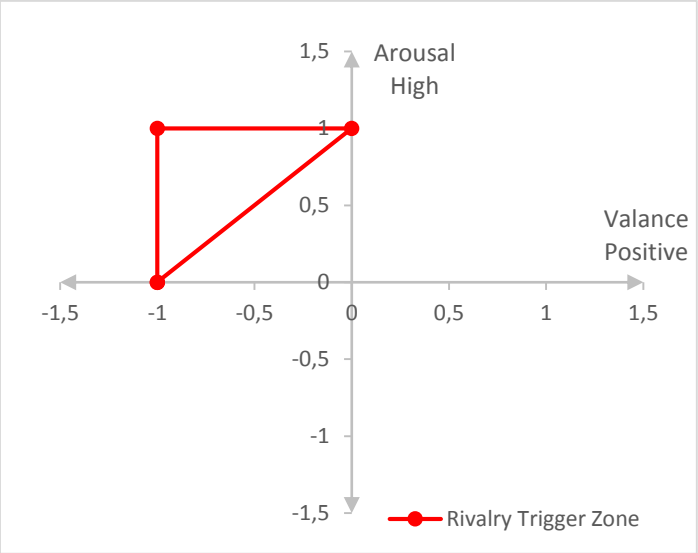

Fig. 8: Rivalry Trigger Zone in the second quadrant. Valance is negative and arousal is positive

The µ-SIC System allows each game agent to have a unique personality. The Rivalry AI also considers the personality as a parameter to adapt the rivalry trigger zone. The idea is to separate the rivalry trigger zone into two pieces. The area which is farther away from the graphs origin stays the rivalry trigger zone. If the mood reaches this area, the rivalry is engaged. The character immediately starts the rivalry. The second part of the divided area is the snap zone. When the mood enters the snap zone, it is decided by chance, if the rivalry is engaged. How large the two areas will be is decided by the character's personality. The personality is composed of a stable/unstable and an introvert/extrovert value. The stable/unstable value influences the size of the snap zone and the chance of the character to snap. When a game agent snaps, the rivalry will start. This means, if a character is unstable, he is more likely to snap and start a rivalry. The introvert/extrovert spectrum defines the size of the rivalry trigger zone. This value can be seen as a difficult adjustment value. A character with a higher extrovert value is more likely to start a rivalry. An introvert character is very hard to arouse and to annoy. Therefore it is hard to start a rivalry with an introvert.

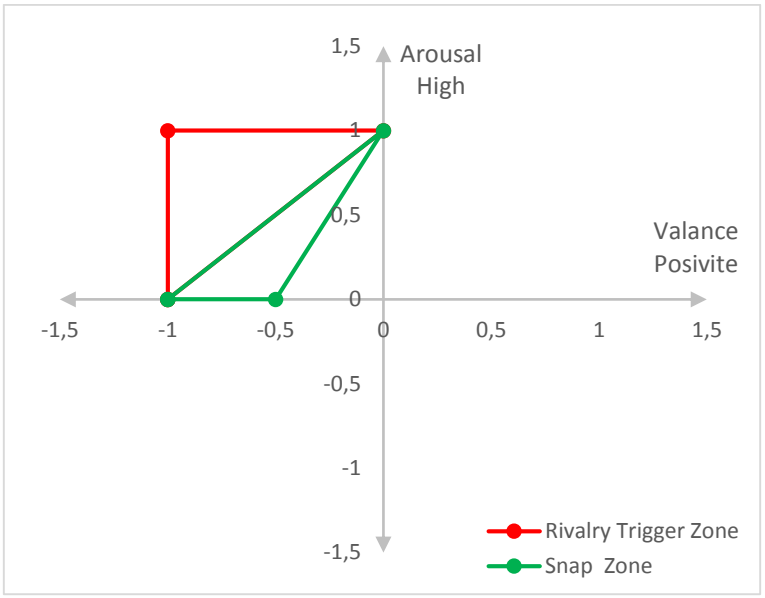

Fig. 9: The mood model with displayed rivalry trigger zone and snap zone

The approach, which is used in the project BareFoot, works as follows. First two lines are defined and the slope for each line is calculated. Each line defines the threshold for one of the two zones. To calculate the slope of a line, two points need to be defined along the line. The labelled points of the two lines are displayed in the figure below. $R1$ and $R2$ are the two points along the rivalry trigger zone line and $S1$ and $S2$ are the points along the snap zone.

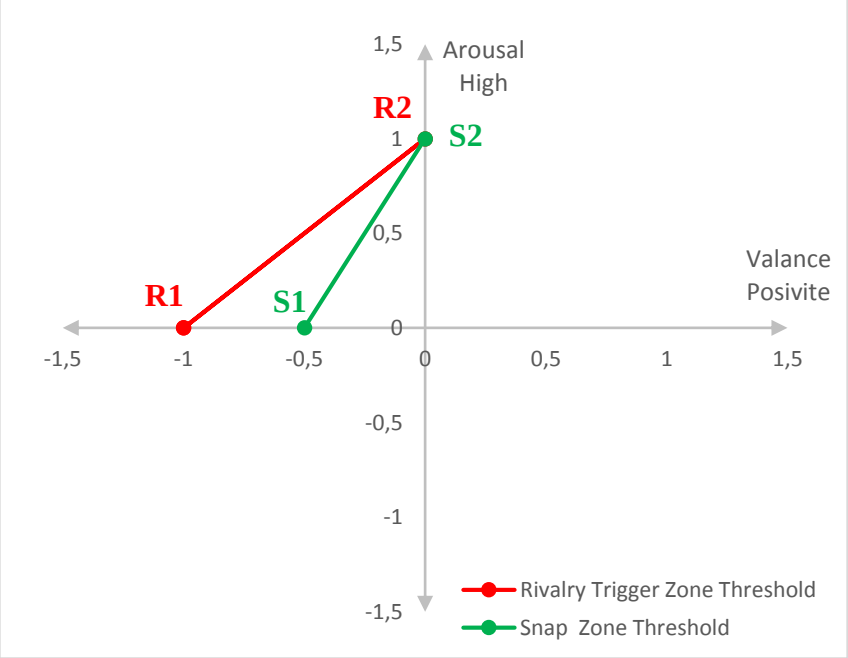

Fig. 10: The four points along the trigger zone threshold lines

A slope is defined as:

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$
$$S1 = (x_1/y_1)$$
$$S2 = (x_2/y_2)$$
$$R1 = (x_1/y_1)$$
$$R2 = (x_2/y_2)$$

Then m is the slope, y_1 is the arousal value of the start point and y_2 of the end point of the line, from which we want to calculate the slope. x_1 is the current negative valance of the start point and x_2 of the end point.

3.3.1. Calculate Rivalry Trigger Zone

The start point $R1$ of the rivalry trigger zone line in the second quadrant is always the same.

$$R1 = (-1/0)$$

Which means the character starts a rivalry, when he is zero aroused, only when the current negative valance value exceeds a threshold of -1. The second point of the line $R2$ depends on the personality. The point lays in a possible range of $E1 = (-0.5/1)$ and $E2 = (0/1)$. Where the point lays exactly, depends on the introvert/extrovert value of the personality. An introvert is hard to provoke and has therefore a smaller rivalry trigger zone. The end point of an introvert lays close to (-0.5/1). The extrovert is easier to arouse. Because of that he has a larger rivalry trigger zone. This means such a personality lays close to the (0/1) point.

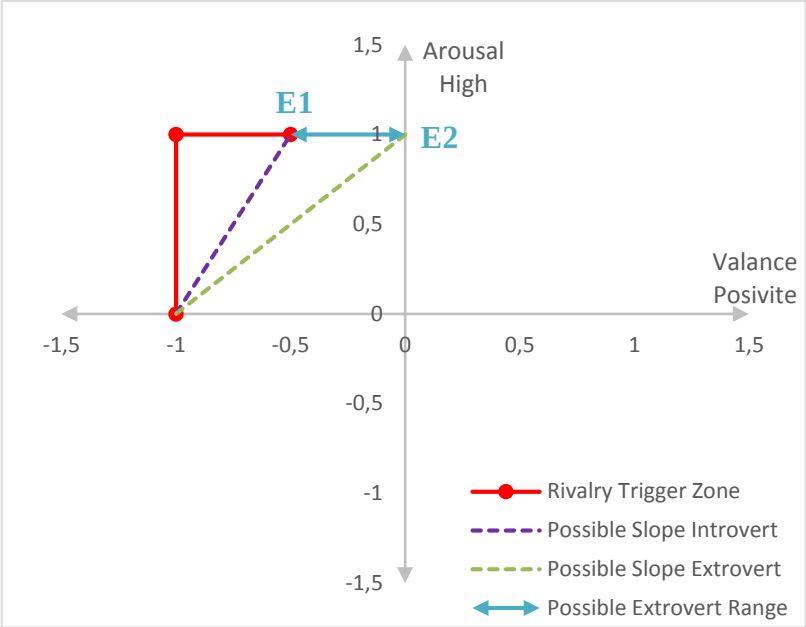

Fig. 11: The possible range of the point R2, depending on the personality's extrovert value

To calculate the second point $R2$ we use the following equation:

$$R2 = (x/y)$$

$$x = \frac{rangeX}{2} - \frac{rangeX}{2} * extrovert$$

$$rangeX = x_{E1} - x_{E2}$$

$$y = 1$$

$rangeX$ is the mood range, where the second point of the rivalry trigger zone line can lay. The extrovert value of the character's personality defines the size of the trigger zone.

3.3.2. Calculate Snape Zone

$S1$ is the starting point of the snap zone in the second quadrant. Where it starts depends on the stable/unstable value of the character's personality. The possible range of the starting point is from $U1 = (-0.75/0)$ to $U2 = (-0.25/0)$. The starting point is calculated as follows.

$$S1 = (x/y)$$

$$x = -rangeX + \frac{rangeX}{2} - \frac{rangeX}{2} * unstable$$

$$rangeX = x_{S1} - x_{S2}$$

$$y = 0$$

Again $rangeX$ is the possible range of the starting point. $unstable$ is the stable/unstable value of the character's personality. This means when the character is unstable the snap zone is larger and vice versa the area is smaller, when the character is stable. The end point of the snap line $S2$ is identical with the end point of the rivalry trigger zone $R2$.

$$S2 = R2$$

Now we are able to calculate the slope of the rivalry trigger zone m_r and the slope of the snap zone m_s.

$$m_r = \frac{y_{P2} - y_{P1}}{x_{P2} - x_{P1}}$$
$$m_s = \frac{y_{P4} - y_{P3}}{x_{P4} - x_{P3}}$$

In order to visualize the different rivalry trigger zones, three different personality types are used to calculate possible areas. The personality types are used from the examples from the µ-SIC System personality system.

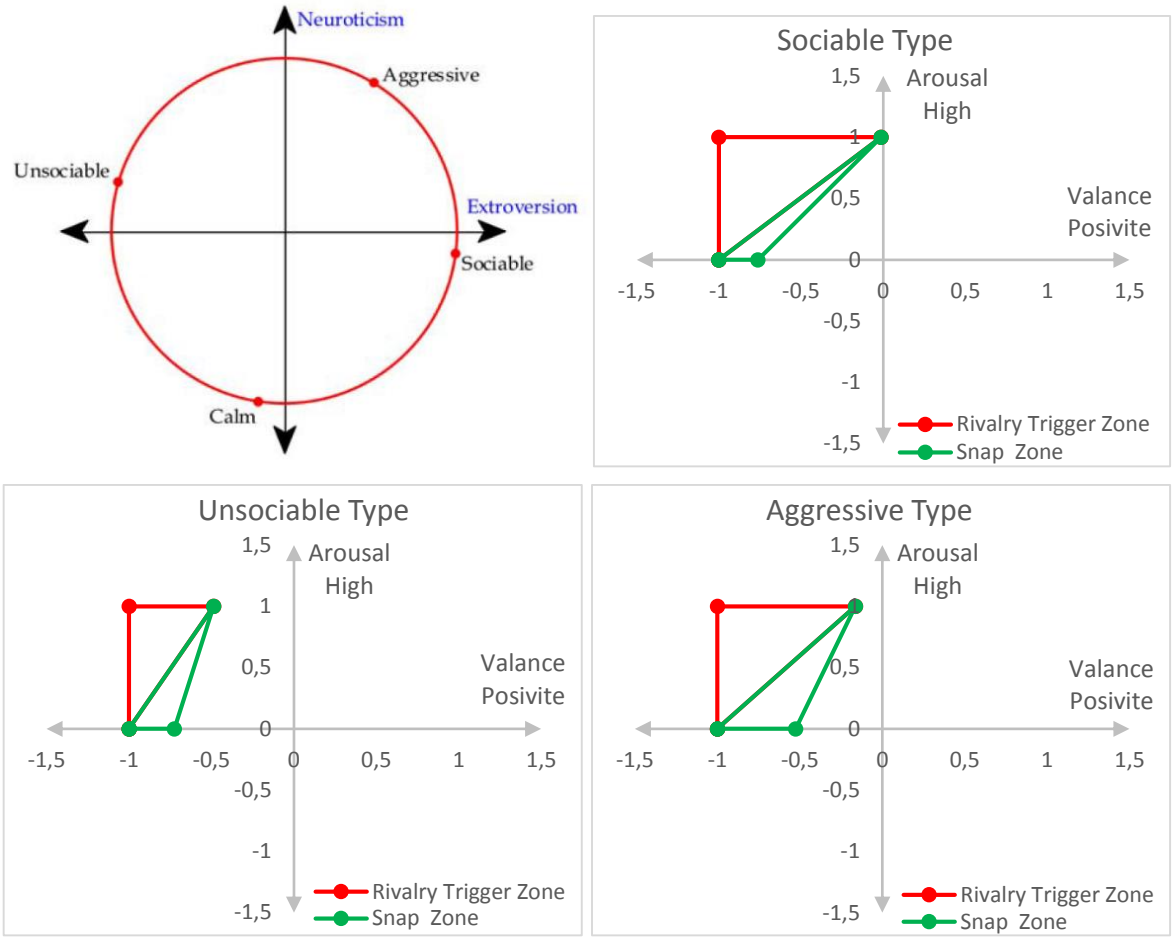

Fig. 12: Examples for different personality types (MacNamee and Cunningham 2003) and the corresponding rivalry trigger zone and snap zone areas

3.3.3. Check if mood triggered rivalry

The last step is to check if the current mood exceeded one of the defined slopes. With a given current negative valance value, it is possible to calculate the corresponding arousal values, which the current arousal value needs to exceeded to trigger the area.

$$y = m * x + b$$

In this equation y is the target arousal value, which has to be exceeded by the current arousal value. m is the slope of the zone we want to check, this can be either the slope of the rivalry trigger zone or the slope of the snap zone. The variable x is the current positive/negative valance value of the character. The variable b can be interpreted as the y-intersect of the line. With the points $R1$ and the slope m_r from the rivalry trigger zone and S1 and m_s from the snap zone it is possible to calculate the corresponding b_r and b_s.

$$b_r = y_{P1} - m_r * x_{P1}$$
$$b_s = y_{P3} - m_s * x_{P3}$$

Now it is possible to calculate the corresponding arousal threshold, corresponding to the current negative valance value. If the current arousal value exceeds the calculated threshold, the rivalry is triggered or in case we are checking the snap zone, we have to execute the random function, if the rivalry should be started.

The character's personality determines whether it is easy to start a rivalry with this characters. The two different personality values affect two different approaches to start the rivalry. The extrovert/introvert value determines how big the rivalry trigger zone will be. Therefore this values decides, if it is hard to start e rivalry or not. The second personality value is the stable/unstable value. This value determines if the character snaps easily and start early a rivalry. This value affects the size of the snap zone. If the rivalry is triggered depends on a random function.

3.4. Socially Interactive Events

The µ-SIC System consists of three modules. The personality system is defined when the game agent is initialized. On the other hand the mood system and the relationship model change continuously. The mood system must be updated whenever an event happens, which affects the character's emotions. It doesn't matter if there is a social interaction with another game agent. Each event needs a data structure, which defines an arousal and a valance value. These values define how the mood is affected by the event. Additionally an event list for all possible or relevant social interactions is needed. These need another structures as the mood affecting events. The structure has to store a value for each relationship category. As soon as an event is triggered, the mood system and the relationship model should be updated.

The problem is to gather and balance these values. The values shouldn't be static, because the most interactions depend on the current situation. For example in a racing game, when someone is attacked, the emotional effect is different, if it happens in sight of the finish line, than right after the start. One possible solution is to let a quality assurance (QA) team or the developer team test the game. Every time one of the above mentioned events happens, the game will be paused and the tester can input the current felt emotion or reaction to the event. For example a value on a scale from 0 to 10 is selected, which defines how intense the event was and if he enjoyed or hated it.

Frijda et al. (1992) referred to the overall felt intensity of an emotion as comprising the following. *"Whatever would go into the generation of a response to a global question such as this: 'How*

intense was your emotional reaction to situation S?' Physical manifestations, neural processes, and much about duration are not included in the model." (Frijda et al. 1992)

Elliott (1992) limited the consideration of emotion states and intensity to Frijda et al. (1992) definition. This is a sufficient enough method to limit the intensity of emotions and is therefore suitable for the Rivalry AI. Additionally to the appearing question, how intense this event felt, the game should keep track of current virtual environment figures, like the health points of the character, how long the game session is running or the current position in the leader board. With a small test team and a few test runs it is possible to gather a test set.

The µ-SIC System uses a small test set to train an Artificial Neural Network (ANN), if a game agent should engage in an interaction with another character. This system could be adapted to return a balanced emotion value for a certain event, which depends on the current game environment. For example with data like how intense and annoying an attack one meter in front of the finish line compared to an attack right after the start, it is possible to train the ANN to output a result, when the player gets attacked after the first lap. The trained ANN would be able to return arousal and valance values according to the current game state and the type of the event.

3.5. Player Modelling

As mentioned before the µ-SIC System is designed for all game agents. This means it is possible to simulate both, the emotions of NPCs and of PCs. This is a shallow form of player modelling. Player modelling is a term used in personalization of games. The goal is a track the player's strategy, style or habits and try to model these findings in computable structures. For the Rivalry AI the user's current emotional state should be modelled. It is possible to model the player's emotional state with the µ-SIC System. If it is already implemented for NPCs, no extra work is required to implement it for PCs. One negative aspect of this system is, that the human mind is a complex mechanism and the µ-SIC System uses rather simplified systems of psychological models to model the human mind. Therefore deviations will be inevitable. This means the µ-SIC System is made for non-complex emotional simulations. It is suitable for casual and arcade games, which want to extend their NPCs by personalized behaviour. It isn't suitable for complex game simulations like the SIMS (EA Maxis 2000). The advantages for the µ-SIC System bear the fact that it is made for casual and arcade games. The system is easy to understand, fast to implement and models the emotional state according to psychological models. It is predestined to model the emotional state for the Rivalry AI.

3.6. Trigger Rivalry considering the Relationship

The reason to model the player's emotional state is to trigger the rivalry in the correct moment. The first rule is, the rivalry can only be triggered, when a socially interactive event happened. This means, when two agents are involved in an event like an attack. It would be nonsense to trigger a rivalry, when only the mood of a character is altered and not his relationship to another agent. If there isn't another character involved in the event, there is no opponent to start the rivalry with. An example for affecting only the mood of a character is, if the character dies because of his own fault. A too deep drop and he dies from the impact on the ground.

It is already clarified, how to technically start the rivalry considering the character's current mood. This theory needs to be extended, by considering the two involved agents' relationship. Therefore rules are required, at which state of the relationship the rivalry could be started. For example when there are the relationship categories Like/Dislike, Attractedness, Dominant, Intimacy and Interestedness at least three of these five categories have to be negative. Another possible rule could be, that the values have to exceed a given threshold. If these rules apply and the mood system entered the rivalry trigger zone, we can consider the next step in the logic to start the rivalry.

There are three different possibilities how to trigger the Rivalry AI. The first one is, if at least one of the two involved agents meets the mood and relationship requirements to start the rivalry. This approach brings up the problem, that when agent A attacks agent B and only agent A meets the requirements to start the rivalry, the character which experienced an emotionally positive event could start the rivalry. Because agent A is in a good mood, when he successfully attacked an opponent. Therefore the rivalry should only start, when the character, which gets annoyed, meets the requirements to start a rivalry. This concept brings up an implication, which should be considered. When the NPC is in a bad mood and the user is in a good mood, the NPC is able to start a rivalry. The NPC will then attack and annoy the player and the user's emotions will change rapidly. The game adaption will not meet the user's emotions, because the rivalry should start when the player is in a bad mood. The game isn't adapting to the player emotions, but to the NPCs emotion. The Rivalry AI is triggered by the NPC. This is not a problem, but it should be considered, when using this approach.

The second option is to start the rivalry, when both agents are aggressive enough. This approach brings up the problem, if one of the agents has a very bad match and the other one experiences an amusing match. In this case the annoyed player wants to start the rivalry, but the other player is too happy to confirm this request. This can lead to blocking the rivalry and therefore less rivalries could be started, compared to the first approach.

The last possibility is to only consider the player. Whenever an annoying event happens to the player, the logic will check if he meets the requirements to start a rivalry. This has the advantage to directly react and adapt to the players current emotional state. The disadvantage of this option is that we don’t simulate the NPCs emotions. This means, if the AI is annoyed and wants to start it rivalry, it has to wait for the player until he meets the rivalry requirements.

4. Methods

The concept of the Rivalry AI should be implemented into a game. The Rivalry AI concept was used in the project BareFoot, which has been developed by students at the University of Applied Science in Salzburg for the past two year.

4.1. BareFoot

BareFoot is a classical Fun Racer like Mario Kart (Nintendo 1992) or Diddy Kong Racing (Rare Ltd. 1997) for up to eight concurrent players. The player controls a fictional creature, which is running on a race track, has to collect power ups and can attack other players. The user can freely build his own creature before each race. A creature consists of five body parts: head, arms, legs, torso and tail. Every single body part affects the movement of the beast. Furthermore the body part decides, which type of power up the player can use during the race.

The game is designed as a free to play online multiplayer game. It is possible to play the online multiplayer or challenge your friends via split screen. When there are no human opponents accessible, the user can challenge the AI. The success of free to play games usually depend on a huge user base. The gameplay is designed for users to compete and interact with each other. When there are only a few daily active users, it is a problem for the game, because there are not enough players to challenge continuously. Especially independent companies can get problems, when they launch their game title. If there is only a small marketing budget, the user base will increase slowly. To bypass this time period a challenging and motivating AI can hold up the retention rate. A problem which should be solved by the Rivalry AI.

One problem of the Rivalry AI combined with the game BareFoot is the player recognition. For example in Mario Kart are eight unique characters and just one instance of the character is allowed per race. In BareFoot the user is able to build an individual creature out of a finite set of body parts, it is possible that two or more players have similar or identical creatures. As described in the theoretical part, it is essential to recognize the rival. Otherwise it is impossible to build any emotional relationship. Therefore BareFoot uses avatars, which are floating above each creature.

Fig. 13: Ingame Avatar floating above the creature in the game BareFoot

The Rivalry AI requires that the game agents stick close together. BareFoot uses the rubber band method to provide this requirement. The rubber band method is predestined for the Fun Racer genre. The idea is to slow down the character, which is in lead and speed up the characters behind. How strong the speed alterations will be, depends on the distance between the players. This assures that the game agents stick close together.

As it is a goal to extend the AI of a racing game, the basics on how a racing AI works are required. This is important for the following chapters, which cover specific actions, which the Rivalry AI is able to take. First the racetrack representation for the AI is described. Then the racing logic itself is covered, followed by a short excursion how it is possible to automatically parse a racetrack for an AI. This is important to keep the game expendable for additional level packs. Otherwise the AI will increase the workload for a new level significantly.

4.2. Racing AI

There are several state of the art ways how to implement an artificial intelligence (AI) for racing games (Schwab 2004). All employ a similar racetrack representation, which makes is possible for the AI to drive on the circuit. (Biasillo 2002a). There are no documented methods to automatically create the correct representation for the AI from a racetrack mesh. It is possible to set the track representation manually in an editor. If you have multiple racetracks in your game it is very time consuming to set up each track manually. Furthermore it’s not flexible to maintain

or update the track, i.e. if the level designer decides to make some modifications. An automatic track parser is a solution to this problem.

4.2.1. Racetrack Representation

The racetrack has to be represented in a practical way for the AI system. Otherwise the AI system can't decide, which path is the best, how fast to drive on the current path or if a curve is ahead.

4.2.1.1. Sector

The racetrack, on which the player can drive, is split up in sectors. Thus makes it possible for the AI to identify the on-track areas. A sector is composed of a leading edge and a trailing edge, which are called interfaces. Other parts of a sector are the racing line, the overtaking line and additional information for the AI. The sectors are stored in a linked list to keep track of the correct order. Each sector holds an array to the following sectors. With this technique it is possible to split up the racetrack in multiple paths, if there are more than one sector stored in this array. The last information the sector memorizes is the distance from the start of the race-track to the sector. With this data the AI system can calculate how far it has driven on the racetrack and compare it to opponents.

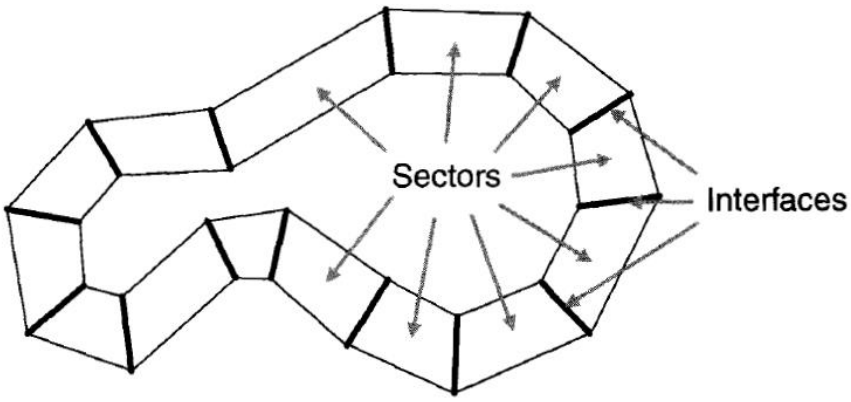

Fig. 14: Racetrack split up in sectors and interfaces (Biasillo 2002a, 439)

4.2.1.2. Interface

An interface is a straight line with four nodes along itself. The start node and end node of the straight line determine the left most and right most boundaries of the sector. Further there are a driving line node and overtaking line node stored in the interface structure.

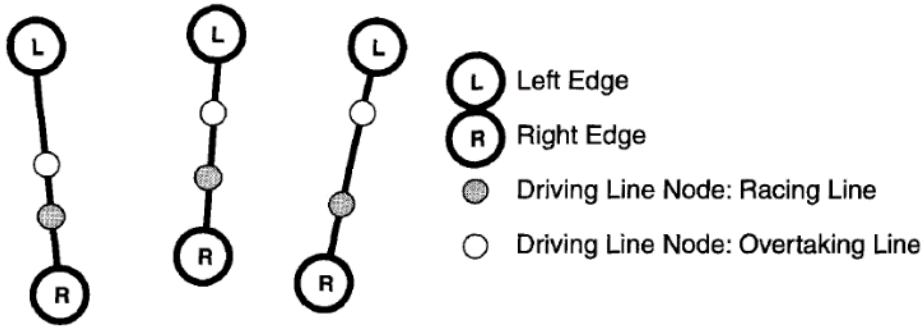

Fig. 15: 3 interfaces with 4 nodes (Biasillo 2002a, 440)

When the left end of the leading edge are connected with the left end of the trailing edge, the left border edge of the current sector is formed and vice versa for the right border edge. These edges combined with the two interfaces create a plane for the boundaries of the sector. With this plane the AI system can check, if the driver is on the sector. Similar to the left and right edges of the sector the same method works for the overtaking line, which is constructed of the overtaking line nodes of the leading and trailing interfaces. For the racing line the driving line nodes are connected.

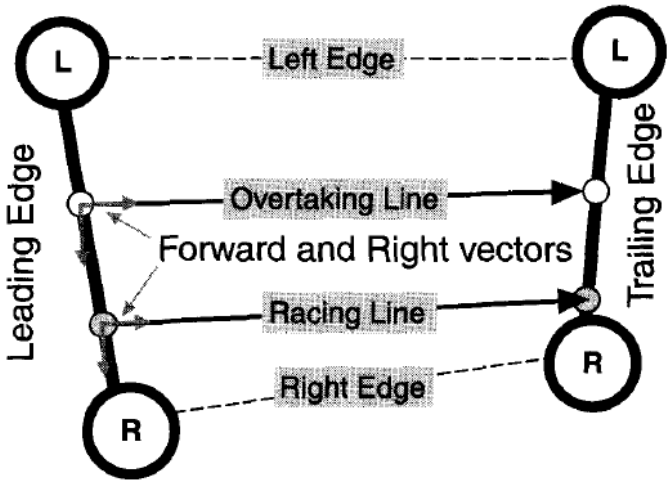

Fig. 16: Constructed sector (Biasillo 2002a, 440)

4.2.1.3. Racing Line

The racing line describes the ideal route along the racetrack. The racing line of a sector is the optimal path between two interfaces. Its class stores the starting position of the line, the line length of the current section, the forward direction vector and the right direction vector. The AI can check if the driver is heading in the correct direction with the forward direction vector of the racing lane and the forward vector of the racer. The right direction vector is perpendicular to the forward vector and is used to determine how far the driver is away from the racing line.

The overtaking line is used when a driver catches up with an opponent. If the driver wants to overtake, he switches to the overtaking line and tries to pass the other rider. The structure of the overtaking line is the same as the structure of the racing line.

4.2.1.4. Information for AI

Additional the sector structure stores several information for the AI system. First of all the path type, which marks a shortcut, a weapon pick-up route or simply a normal sector. Also the terrain type like mud, snow or grass is important for the AI. Another information is the possible danger of the sector, which contains gap-, hole- or obstacle warnings, so the AI can slow down before this sector. The break or throttle data is another indicator for the AI system to accelerate for example before a long straight part of the racetrack or break before a specific sector.

4.2.2. Racing AI Logic

At the start of a race the starting grid has to be defined. Then the AI has to wait for the green light to start the race. At this moment the state machine changes to the race state. The main task of this state is to traverse through the sectors, find and drive to a target along the racing line. It has to check, if it is necessary to overtake an opponent.

4.2.2.1. Traversing the Sectors

First of all the AI system has to find the current sector, on which it is driving. Then the sector ahead is checked and if a split decision is needed. To simulate a human behaviour the AI should anticipate the route ahead. Therefore the AI looks a few sectors ahead to find the target point along the race line. The look ahead distance is proportional to the current speed of the racer. If the character is fast, the distance is bigger and vice versa. With this anticipation it is possible to check, if there is a sharp curve or a long straight part ahead. The result defines if the character accelerates to the maximum speed or breaks in anticipation to the upcoming curve. The side effect of looking ahead is, when a hairpin turn comes up. The AI would ignore the curve and cut the corner.

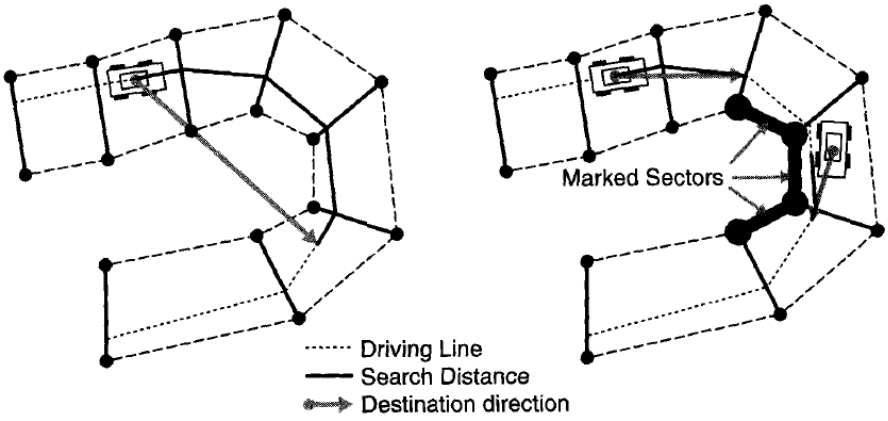

Fig. 17: AI cutting hairpin curve (Biasillo 2002b, 447)

The solution to this problem is that the sectors along a sharp curve are marked as hairpin turns. If the AI finds a marked sector, it stops looking any further. The AI also needs a minimal look ahead distance, otherwise it could stop on the road under special circumstances.

4.2.2.2. Driving to the Target

If the AI has picked the correct sector ahead, a point along the racing line has to be defined as target point. This spot marks the target position where the driver is heading. The target point affects the steering of the AI. If the target is left from the driver, it drives to the left and vice versa for the right side.

4.2.2.3. Overtaking

If the AI catches up with an opponent, it has to try an overtaking manoeuver. In this situation the AI has to choose a new driving target point, which lies along the overtaking line instead of the racing line. The new target point influences the steering behaviour of the AI.

4.2.2.4. Rubber Band

When an AI system is designed for a fun racer, a catch-up logic should be considered. That means, if the AI is in lead, it starts to act worse than usual. The difficulty should be adapted to the current race position of the AI. This can be accomplished by limiting the maximum speed, breaking earlier for corners or accelerate slower. Another approach is to get collected power ups according to the current position. If the AI is in the back it gets stronger power ups. The AI system could aim only for other AI players, if it is in front of the human player. The rubber band is a sub domain of the Dynamic Difficulty Adjustment (DDA).

4.2.3. Automatic Track Parsing

It is time consuming to set up the track representation manually. Especially when it comes to multiple tracks. The solution is to parse the racetrack using an algorithm. In the following a semi-automatic method is described.

4.2.3.1. Preparing Racetrack Mesh

The first step for parsing the racetrack is the generation of a simplified mesh of the race course. It would be too complex to parse the entire 3D mesh of the racetrack. In Fig. 18 the original racetrack of the game BareFoot is shown and next to it is the simplified mesh, which is used to demonstrate the parse algorithm.

a)

b)

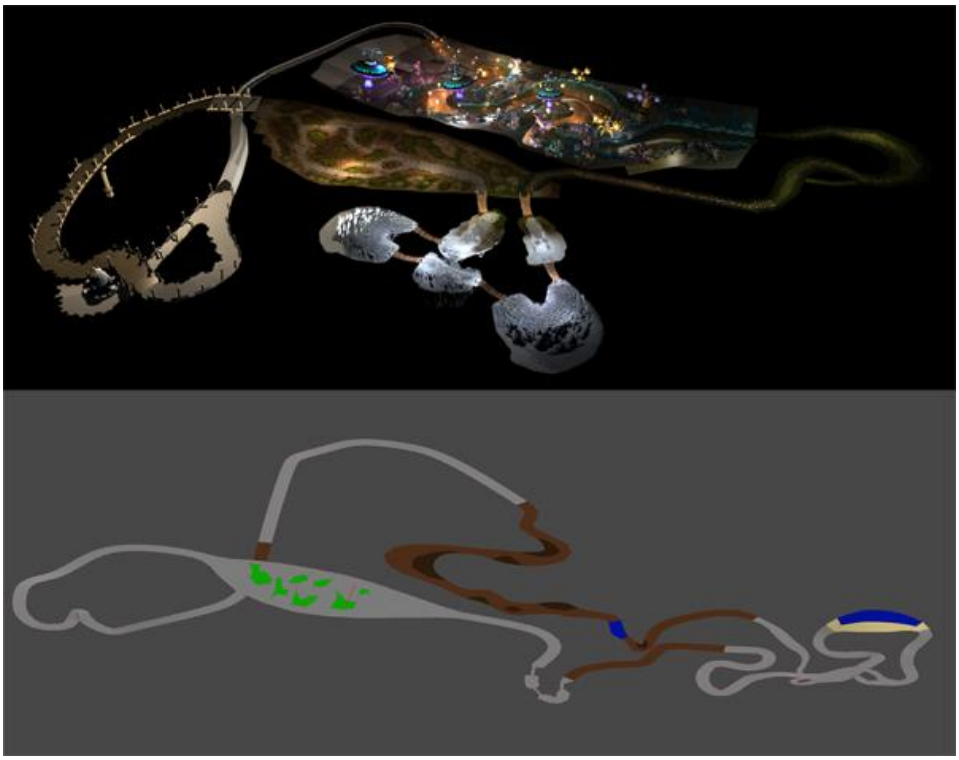

Fig. 18: a) Original Racetrack Mesh; b) Simplified Racetrack Mesh

The reduced mesh has less details than the original mesh and therefore less vertices. In the figure below a comparison of the vertex detail of the two meshes is represented. The next step is to find all vertices, which determine the bounds of the racetrack. Based on the simplified mesh it is easier to find these.

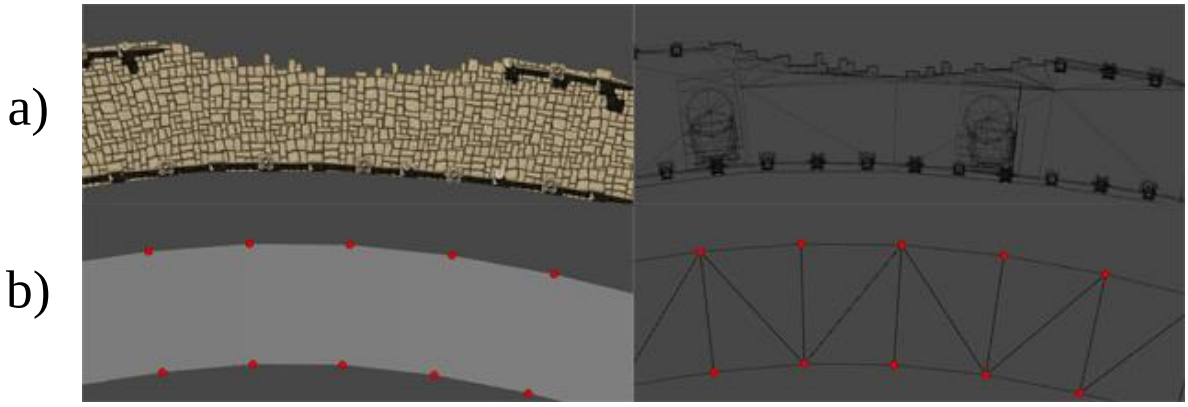

Fig. 19: Comparison (a) detailed and (b) simplified mesh

4.2.3.2. Finding Racetrack Bounds

We have to take a closer look at the geometric properties of the mesh. It is possible to find the mesh bounds by using the edges of the mesh. If an edge connects two vertices, which are not part of the same outline edge, then the edge is used by two different faces of the mesh. As shown in the figure below two faces use the same edge.

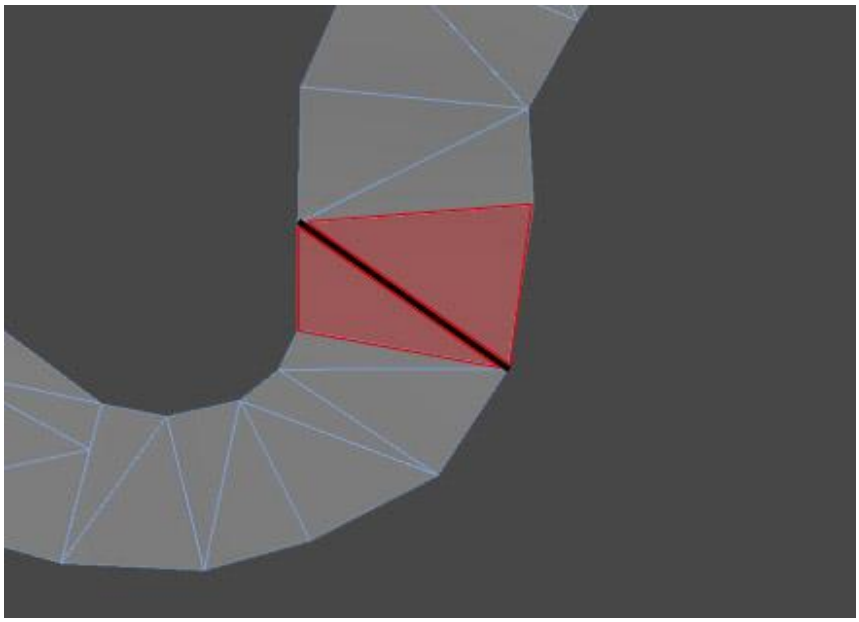

Fig. 20: Double used edge. Black Line: Edge. Red Areas: 2 Faces.

When taking a closer look at an outline edge it turns out, it only is used by a single face.

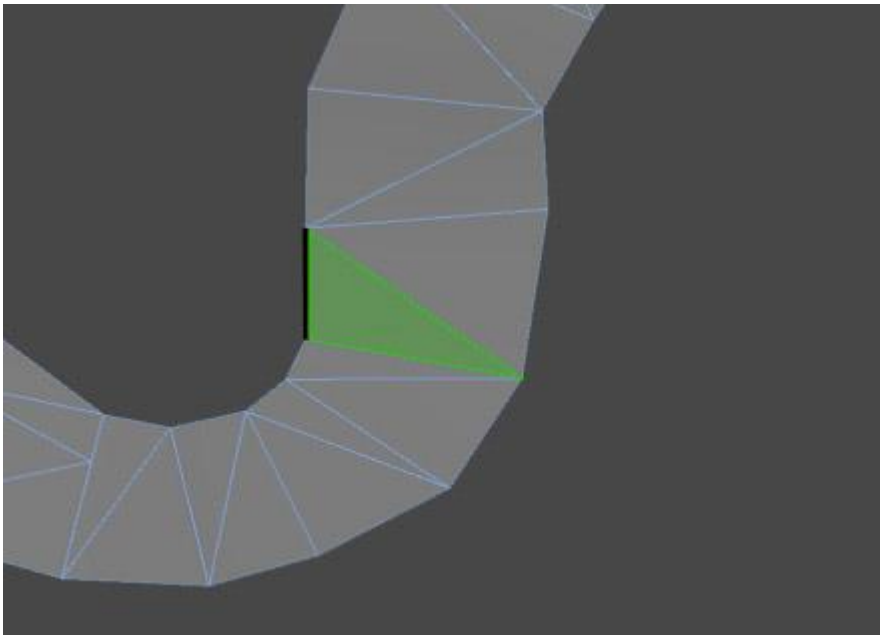

Fig. 21: Single used edge. Black Line: Edge. Green Area: Face.

The algorithm creates a map, in which the appearance count of each edge is stored. Then it loops over all faces in the mesh and creates an entry in the map for each edge. If there is already an entry the algorithm increments the counter for this edge. All edges with a higher count than one are deleted after the algorithm has looped over all faces. Afterwards the map contains all outline edges. Each edge has a reference to two vertices, which are the outline vertices. It is important that the mesh is triangulated. Otherwise the algorithm will detect some vertices in the middle of the lane as outline vertices as seen in the following figure.

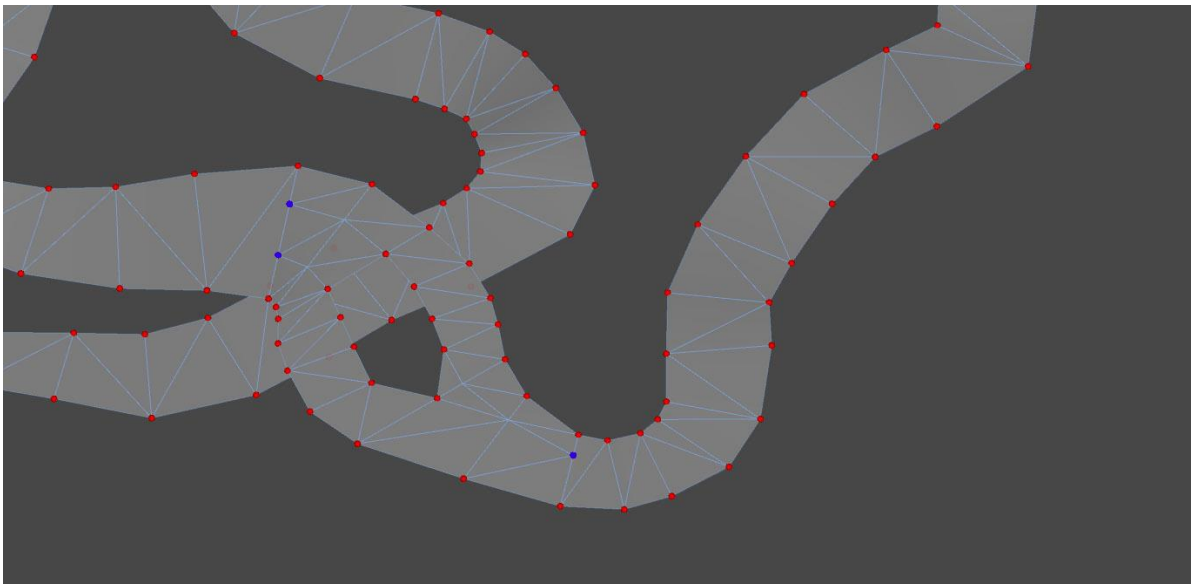

Fig. 22: Red Points: Parsed Outline Vertices. Blue Points: Vertices in middle of the track, parsed as outline vertices because of a not triangulated mesh.

4.2.3.3. Sort Vertices

The next step is to create the interfaces. In order to accomplish this step, the vertices have to be sorted in the right order. Each vertex stores a reference to its previous and next vertex along the mesh. Each outline edge has references to the two vertices, which the edge connects. Therefore it is possible to determine, which vertices are connected. The next step is to find out, which connected vertex is the previous and which is the next one. In order to order these vertices, the start and end edge of the track must be defined.

Set Start- and End-Edges

In order to save time the start and end edges are set manually instead of finding them with an algorithm. In the example racetrack there are different track parts with different physical surface types like stone, water or sand. The player movement depends on the ground type, because of that each track part with a different surface type has its own mesh. Therefore multiple start and end edges have to be defined, one for each track part. The result is shown in the following figure.

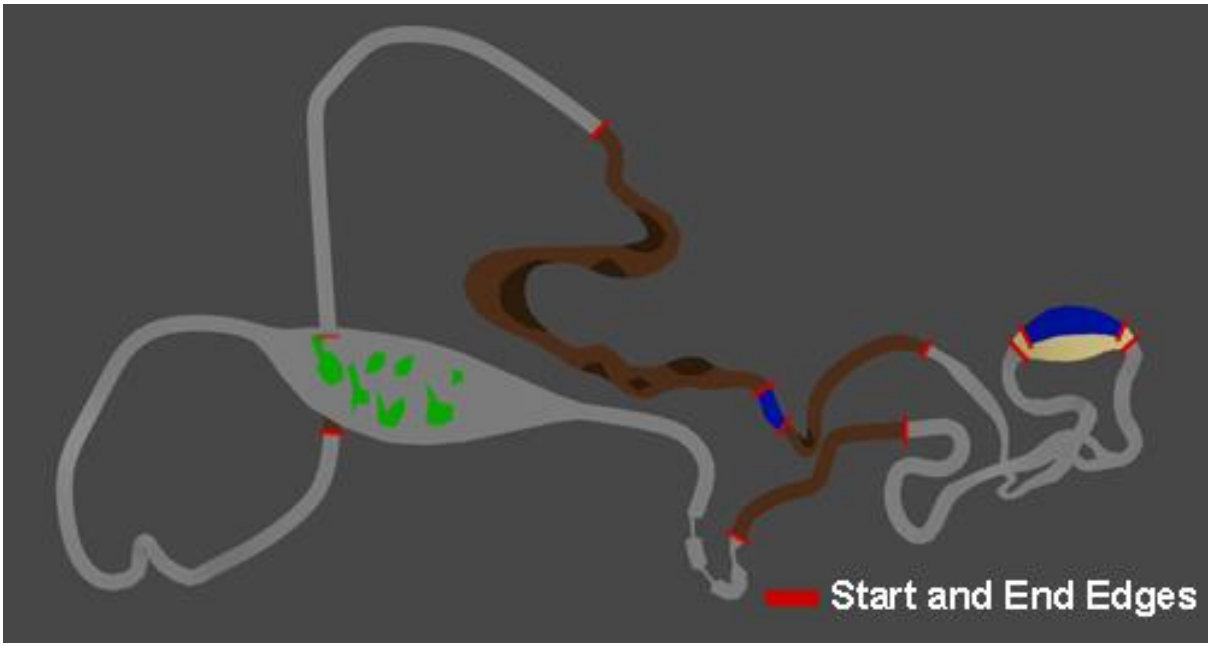

Fig. 23: Start and end edges of the example track

Find Left- and Right-Hand-Side of the Racetrack

Each outline vertex of the mesh will be used as a node of an interface. The defined start edge is already the first interface of the track. Each interface has a left and a right node. The next issue is to find out, which vertex of the start edge is the left and which is the right one.

Therefore a local coordinate system has to be defined for the left node. One of the two nodes is randomly chosen to be the left node of the interface. Then the x-axis of the local coordinate system is defined in the direction to the second node of the interface. This vector is called $\vec{x}$, as shown in the figure below. The next step is to calculate the z-axis. Therefore the vector $\vec{y}$ is required, which points to the next vertex of the left node. By calculating the cross product of $\vec{x}$ and $\vec{y}$ we get $\vec{z}$, which is the z-axis of the local coordinate system. Now the $\vec{y}$ is transformed to the defined local coordinate system. If the y value of the transformed vector is positive, the correct left node has been chosen by chance. Otherwise the right and left node of the interface have to be switched.

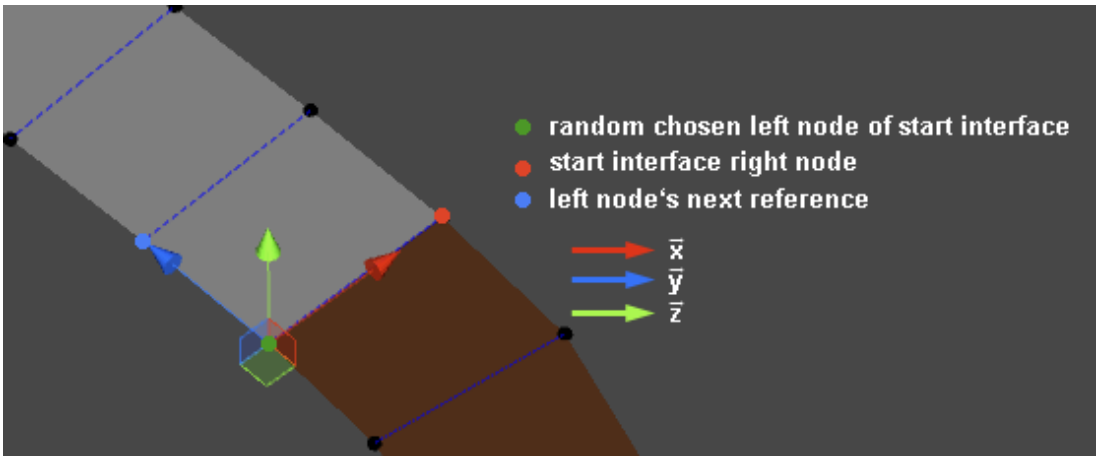

Fig. 24: Finding left and right node of the start interface

4.2.3.4. Creating Interfaces and Sectors

As last step a new interface is created for each next vertex of the left and right node of the current interface. This step will be repeated until a vertex is reached, which is part of the manually defined end edge. If there is an editor available to edit the parsed outline vertices it is possible to delete unwanted vertices before creating the interfaces. Otherwise the mesh should have the exact amount of right and left vertices. The vertices are already in the correct order and therefore the created interfaces are also sorted. Now the interfaces have to be summed up into sectors and the track is represented correctly and is ready to be used by the AI.

4.2.3.5. Outlook Track Parser

There are a lot of possible improvements to this parsing algorithm. The biggest improvement would be to define the start and end edges automatically. This approach doesn't yet consider when the racetrack splits up in multiple paths, how to parse additional information per sector for the AI or how to automatically calculate an optimal racing line.

4.3. Architecture

As explained previously, the µ-SIC System uses a PPA architecture. This architecture is adapted as hierarchical finite state machines (HFSM) for the implementation. Finite state machines (FSM) are the most commonly used technology in game AI programming today (Fu and Houlette 2004). A FSM is a concise, nonlinear description of how an object can change its state over time, possibly in response to events in its environment (Fu and Houlette 2004). In hierarchical FSM each state can actually invoke an entire FSM of its own (Houlette, Fu, and Ross 2000). In the figure below you can see an example of a hierarchical FSM.

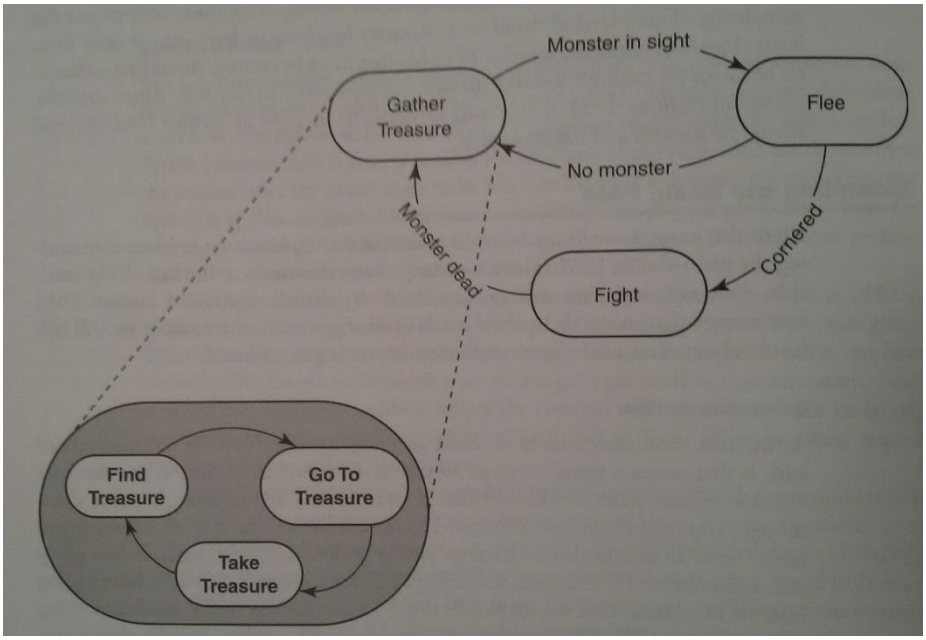

Fig. 25: An example for a hierarchical FSM (Fu and Houlette 2004)

The structure of the hierachical FSM, which is used for the Rivalry AI, looks as follows. On the top level is the Schedule module. The schedule module only uses simple logic to switch states. Below the schedule module comes the role passing system, which depends on the current schedule state. The logic for state transitions is complexer compared to the schedule module. Each role passing system consists of four additional FSMs. These four FSMs manage the logic of every virtual action the character can make. The structure of the hierarchical FSMs is displayed in the figure below. The last remaining module in the PPA architecture is the goal based planning unit. In a fun racer the only superior goal is to finish the race. To acomplish this goal, the AI has to stay on track. As soon as it leaves the track, the AI should find its way back on the track, regardless of the current schedule state or the current adapted role.

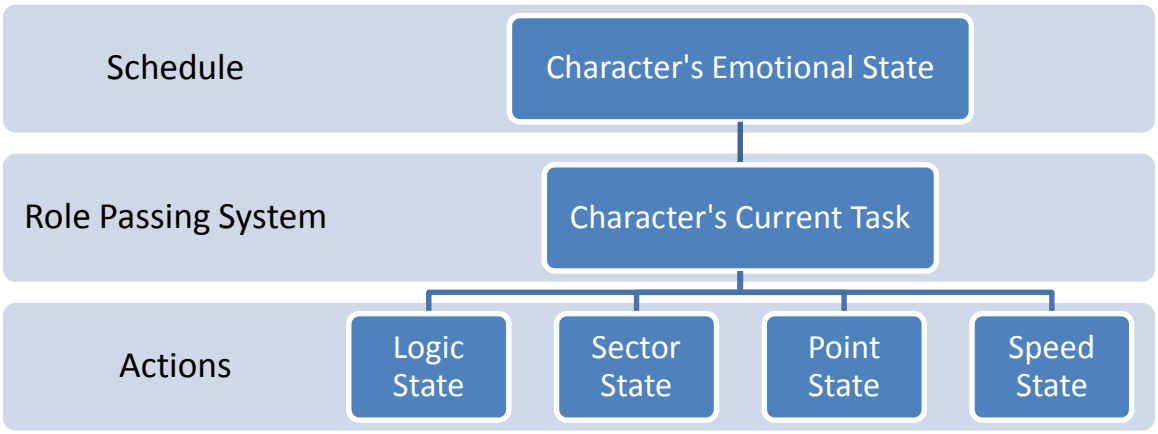

Fig. 26: PPA architecture adapted as hierarchical FSM

4.3.1. Schedule Module

The schedule manages the character's emotional state. This means for example, if he started a rivalry or is still in a normal racing AI condition. The schedule consists of five different states. The states and the possible state transitions are displayed in the figure below.

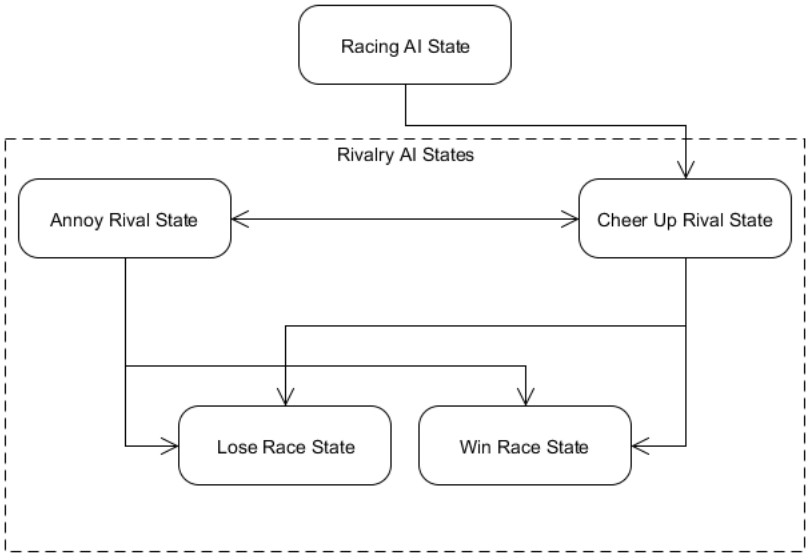

Fig. 27: Schedule States

4.3.1.1. Racing AI State

The Racing AI State is the initial state. The AI is a normal racing AI, which stays on track, tries to drive along the racing line, picks up power ups and attacks enemies. As soon as the player engages into a rivalry with another player, the FSM makes a state change to the Annoy Rival State.

4.3.1.2. Annoy Rival State

As the state name tells, the NPC's goal is to annoy his rival. He will excessivly attack the player. When the AI's character is infront of his rival, he will try to drive in front of his opponent. The NPC will concentrate on collecting offensive power ups, when his rival is in lead. He will try to collected defensive power ups or reverse offensive power ups like mines, when his rival is behind him. If the opponent is in close range behind him and collectable power ups are in reach, the Rivalry AI will try to snatch the collectable power up from under his rival's nose. The NPC will push his opponent aside, if he tries to overtake him. If the modelled mood state of the player exceeds a defined threshold, the FSM switches into the Cheer Up Rival State.

4.3.1.3. Cheer Up Rival State

The goal of the Cheer Up Rival State is to cheer up the annoyed opponent. As mentioned previous on the topic how to motivate the player, the key emotion Schadenfreude was introduced. This is the core emotion, which should be triggered in the player in this state. To accomplish this task, the player gets chances to revenge for all prior annoying actions of the rival. For example the NPC drives close to a cliff or in close range infront of the rival to praise himself for an attack. Another possible action is to spare the rival and shoot another opponent, who drives near the two rivals. The Rivalry AI can drive on purpose into mines on the track. A futher action could be to make mistakes. These moves should accure very rarely, otherwise the player wouldn't enjoy them, if the rival constantly makes mistakes. An example is when the NPC drives behind the player and a sharp curve is appearing ahead. When the player starts breaking for his turn, the Rivalry AI keeps driving at full speed, overtakes the player and in the next moment crashes or drives off the track.

4.3.1.4. Win/Lose Race State

The last remaining states are the Win and Lose Race States. These states depend on the current progress of the game. For example if the player started a Grand Prix mode, in which there are more races than one, the rival is allowed to win some of these races. As mentioned in the theoretical part, this races can be considered as sub goals. By winning these sub goals the rival establishes a competitive situation. When the race is coming to an end, for example when the player comleted 90 percent of the race track, the Rivalry AI has to check, if it should win or lose the current sub goal. Considering the current Grand Prix rank of the two rivals, the AI switches into the win or lose state. In the Win Race State the AI tries to catch up to the player, overtakes him and stays in front of him until the race is over. In the Lose Race State the NPC drops behind his rival and stays in this positions. The Rivarly AI stops to attack the

player until the race is over. There are no futher state transitions, these are the final states in a race.

The schedule starts of with a normal racing AI. When someone started a rivalry with the AI, it switches into the Annoy Rival State. The goal is to provoke the player and facilitate aggressive behaviour. Before this behavoiur transforms into violant behaviour, the AI starts to cheer up the opponent and changes the current state to the Cheer Up Rival State. When the player settled down the FSM makes a transition back to the Annoy Rival State. This cycle between the annoy and cheer up the rival repeats until the race is almost over. Depending on the progress in the game, the FSM switches into the Win Race State or Lose Race State. This is the final state of the race and decided whether the player or the Rivalry AI wins the race. These states will be covered in more detail in the next section.

4.3.2. Role Passing System

The role passing system depends on the current schedule state. The schedule state defines, which actions are possible. Each action state of the role passing system posesses four states, one state for each sub FSM. To clarify the tasks of these FSMs, the steering algorithm of the racing AI has to be briefly repeated. The AI searches for a target sector ahead, calculates a point on the sector and drives towards this point. If the point is left from the character, it steers left. The method to get these target point depends on the current situation of the character. The four FSM construct this method to determine the target point.

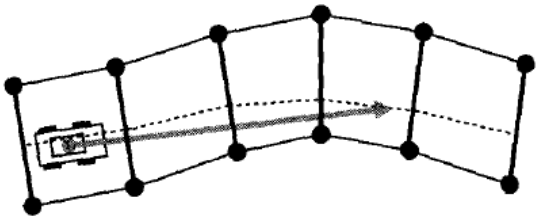

Fig. 28: Steering to target point (Biasillo 2002b)

4.3.2.1. Decision Making FSM

The first FSM of the four sub FSMs handles the transition logic between the action states. For example in certain situations it is not possible to change into another state. When the AI is trying to overtake the opponent, it is not possible to switch to the collect power up state, because overtaking has a higher priority than to collect a power up. Imagine the AI is trying for a longer period of time to overtake the enemy and finally gets the chance by passing him on

the inside of a curve. In this moment the possibility to collect a power up appears. The AI leaves the inside of the curve and steers to the outside. The Rivalry AI collected a power up, but missed the chance to overtake. To solve this problem, the transition from the overtaking state to the collect power up state is restricted. This restriction is defined in the Logic FSM of the overtaking state.

4.3.2.2. Target Sector FSM

The second state machine traverses the race track sectors in front of the AI to find the target sector according to the look ahead distance. The search algorithm depends on the current adapted role passed from the current schedule state. The different possible algorithms differ for example in split path decisions. When the AI is in the drop back state, it avoids short cuts. Another difference can be the look ahead distance. As explained in the racing AI chapter, the look ahead distance of the AI depends on the current speed of the character. In certain situation the look ahead distance shouldn't depend on the current speed, but should be set the to a certain value. For example, when the Rivalry AI tries to block his opponents, the look ahead distance should be shorter than usual. Thus makes it possible to react faster to direction changes of the opponent.

4.3.2.3. Target Point FSM

The goal of the thrid FSM is to define a target point along the previous found target sector. Usually this would be a point along the race line of the current sector, depending on the look ahead distance. But there are exceptions, for example when the character has to collect a power up, he has to leave the race line and aim for the collectable power up position. Another reason to drift apart the race line is the overtaking maneuver. The character has to find a point on the sector, which provides an open street and has to start steering towards this point.

4.3.2.4. Target Speed FSM

The last state machine manages the calculation of the target speed. Under normal circumstances the target speed depends on the race track and on the target point. The race track defines how fast the character can drive in a certain curve. These informations are stored in the target sector. The second parameter for the target speed is the target point. For example if the character runs at full speed and should turn 90 degrees in the next second, but is physically limited to only 70 degrees per second, the target speed will consider the physical limitations. Another example is, when the character tries to overtake an opponent. He will risk to exceed the maximum speed definded by the race track, to have a better chance to successfully overtake the enemy.

The action state defines for each of these four FSMs a state. This architecture can reuse the same states to find the target point. It is a modular system, different schedule states are able to work with the same action states. The states from the four sub FSMs drive the actual behavoiur of the character. The other two layers of the architecture are only contoiners to define, which states are used to drive a certain behaviour in a certain situation.

4.3.2.5. Racing AI State

Each Schedule state has a similar set of action states. The following chapters introduce the possible sub states from the Racing AI State and the transitions between these states. An overview of the states is displayed in the figure below.

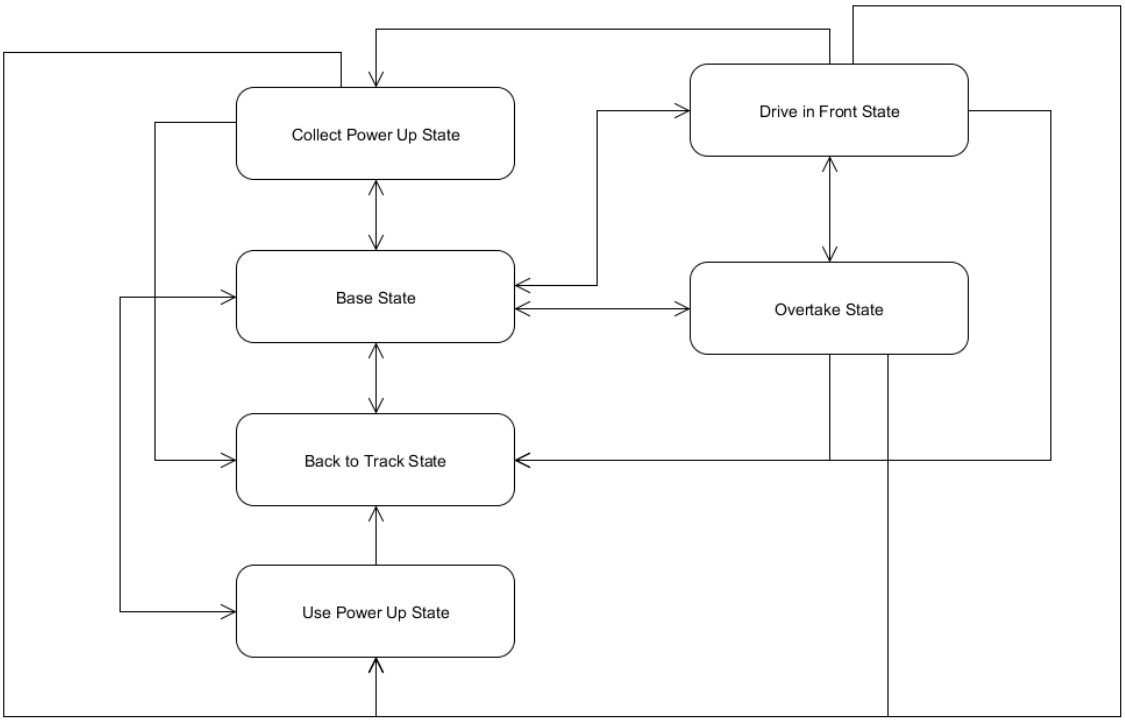

Fig. 29: Sub states of the Racing AI State

The first state of the Racing AI States is the Base State. The NPC is driving according to the racing AI logic. He looks a certain distance ahead, finds a point along the race line and keeps driving towards this point. The base state is able to switch into every other state and every other state can switch back to the base state.

Back To Track State

The next state is the Back To Track State. This state can be seen as the goal based planning unit. The long term goal of the AI is to finish the race, therefore it has to stay on the track. No

matter in which action state the FSM is, it always checks first, if the character is on track. When it isn't on the track, the state machine switches automatically in the Back To Track State. In this state the Sector FSM allways returns the last valid sector, on which the creature was before leaving the track. The logic state machine checks, if the creature got back on the track. When the character reaches a sector, the action state switches back to the base state.

Collect Power Up State

As soon as the defined target sector is marked as a collectable power up sector, the FSM switches to the Collect Power Up State. Depending on the schedule state a decision algorithm finds a target collectable power up and sets the target point to the position of the power up. For example when the AI is in front of his rival, it will primary choose defensive power ups as targets. On the other hand when the AI is behind the rival, it will select offensive power ups, in order to be able to attack him. If the character is trying to overtake the opponent, it will go for speed up power ups. As soon as the AI chose a collectable power up, it registers the character as a listner. When the power up is collected, it notifies all listeners. When the AI gets notified, it checks, if the collector was the Rivalry AI itself or if an enemy snatches away the power up. When the AI successfully collected the power up, the FSM switches back to base state. When someone snatched the power up away, the AI checks, if the physical steering limitations allow to go for another power up on the target sector, without decreasing the speed. When it isn't possible, the state machine returns to the Base State, otherwise the procedure starts all over again.

Overtake State

When any opponent enters a defined Region of Interest (ROI) in front of the AI, it compares the speed difference of the two characters. If the character is faster than the opponent, it switches into the Overtake State. An example for a ROI is shown in the figure below.

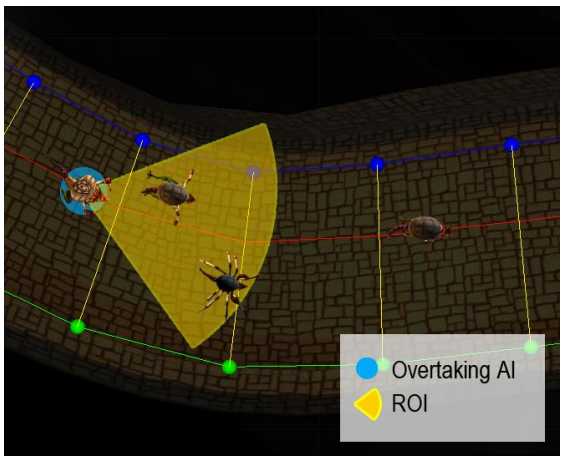

Fig. 30: Region of Interest (ROI) to start overtaking

After entering the Overtake State the AI has to find a cleared lane infront of it. This means AI needs a modified racing line. The goal is to latterally shift the race line and find a possible overtaking line, which isn't blocked by an opponent in the ROI. Therefore we store all opponents in a list and search for a overtaking line, which still lays on the target sector.

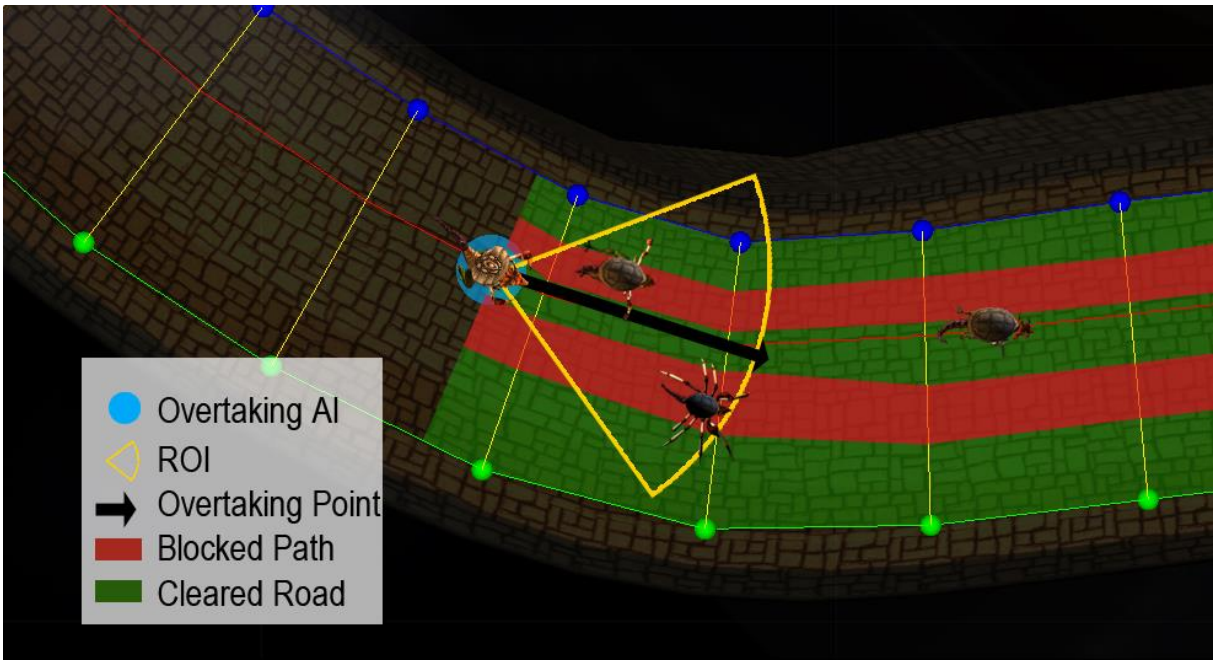

Fig. 31: Blocked Path and Cleared Road ahead

If the target sector is part of a curve, the search algorithm for the overtaking line starts searching in the inward direction of the curve. If there is no space for the AI to overtake, the algorithm continues to search in the outward direction of the curve. If all possibilities in front of the AI are blocked, it sticks to the last calculated overtaking point. The algorithm searches constantly for an overtaking line, if there isn't any cleared road ahead on the inward site, the algorithm starts to search on the other side. This could force the problem that the AI starts to overtake on the outward side and when the AI has almost taken the lead, the searching algorithm could find a possible overtaking line on the inward side. The character automatically steers in the inward direction and fails thereby the overtaking manoeuver. To prevent the character to constantly switch the side, on which he wants to overtake the opponent, the direction of the first determined overtaking line is stored. Additionally a timer manages to abort this overtaking side, if the AI hasn't taken the lead within ten seconds. The look ahead distance is limited to achieve a more realistic overtaking behaviour. As soon as the creature has taken the lead, and no opponents are remaining in the overtaking ROI, the state machine switches back to the Base State.

Drive In Front State

The Drive In Front State is triggered with a similar concept to the Overtake State. When someone starts driving in a ROI behind the AI, it switches to the Drive in Front State.

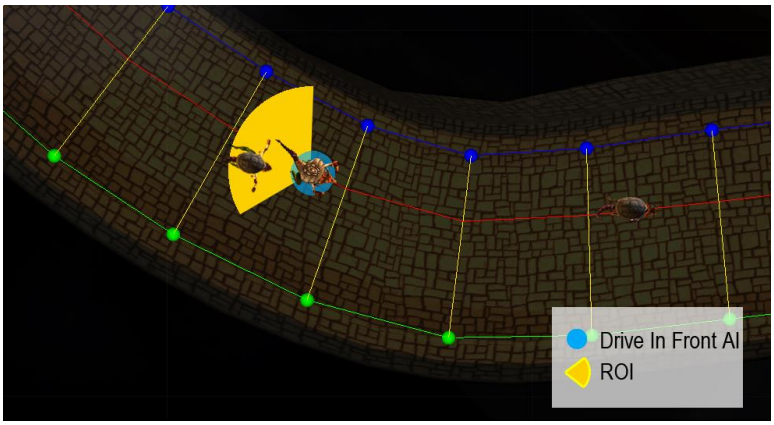

Fig. 32: Drive in Front ROI

The goal of this state is to complicate the overtaking for the opponent behind. It is not the goal to prevent or prohibit the overtaking, this comes up later, when the Rivalry AI states are presented. The offset of the opponent in the ROI is calculated and this offset is added to the target point. With this approach the AI drives to the position, where the opponent was in the last frame, therefore it isn't too hard and complicated to overtake the AI, while it is in the Drive In Front State. Additionally a timer was created, which determines how long the AI drives in front of the opponent. After the maximum time was reached, the state machine returns to the Base State and the AI lets the opponent pass. The other option to return to the base state is, when the opponent left the ROI, by overtaking or by falling behind.

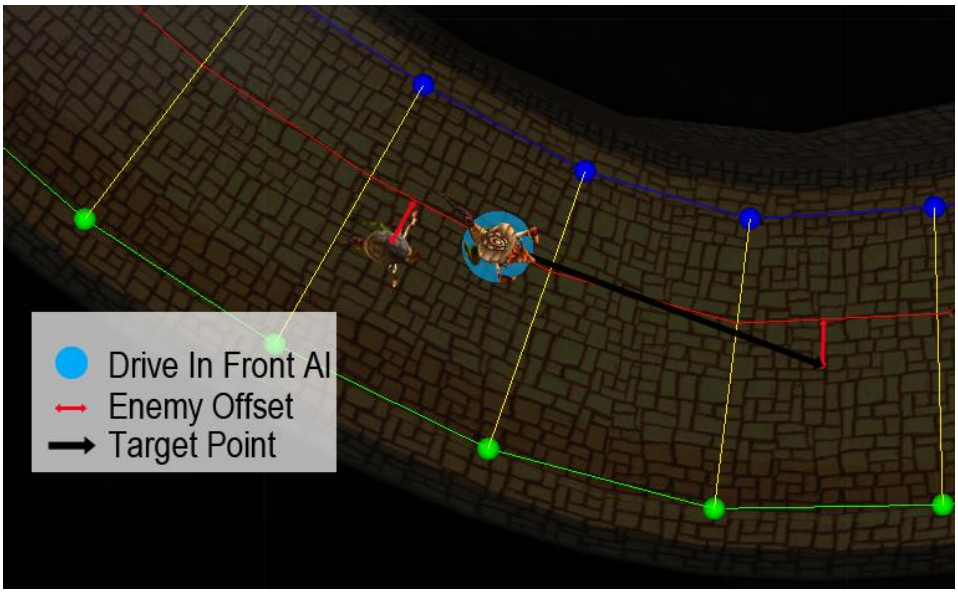

Fig. 33: Target Point depends of the enemy's offset to the race line

Use Power Up

The remaining transition between the states is the transition into a power up state. Certain power ups require a state. For example when an opponent is in shooting range, but the AI has to aim before it is able to shoot. In this example the state machine would change to the Use Power Up State and as soon the AI aims at the desired opponent, the AI attacks. If the enemy exits the shooting range, the state machine returns to the Base State without shooting.

4.3.2.6. Annoy Rival State

As mentioned before the following states look all similar to the Racing AI State. The rivalry states have additionally a few more states, for example to switch the position and to annoy the player.

Collect Power Up State

As soon as the state machine enters the Annoy Rival State a rivalry is going on. The Back To Track State remains the identical state as in the Racing AI State. The first state, which differs from the previous introduced states, is the Collect Power Up State. When the target sector is marked as collectable power up sector, the FSM switches into the Collect Power Up State. If the rival of the AI is driving in a defined ROI behind the AI, the AI starts to drive in front of the rival. The goal is to snatch away the power up from under the rival's nose. Except for the logic state of the logic FSM, the other three states are the exact same state as in the Drive In Front State from the Racing AI State. The transition logic for this state is different, but the executed behaviour remains the same. The logic state has to decide if the normal behaviour of the Collect Power Up State or the snatch away the power up behaviour should be executed.

Stay In Lead State

The next state is the Stay In Lead State, which is also related to the Drive In Front State. In order to stay in lead an additional rubber band system is added for the Rivalry AI. The rubber band assures, that the AI stays five to 15 meters in front of his rival. The next step is to block the enemy. Therefore not the offset of the rival to the racing line is considered, but the offset from the position where the rival is looking at. The rival's offset has to be calculated and the forward vector of the rival's character is added to the offset. Thus gives defines the position where the rival will drive to. In the Drive In Front state, the position where the rival was in the last frame was calculated. With this technique it is possible to calculate the future position of the rival. Therefore it will be harder to overtake the AI compared to the Drive In Front State.

Stay Behind State

The Stay Behind State also activates the additional rubber band. In this state the goal of the rubber band is to keep the AI five to 15 meter behind the rival. The pending state transitions, which we haven't explained yet, are state changes, when the player should switch the race position with his rival. As mentioned in the theoretical part, one goal of the rivalry is to establish a competitive situation to facilitate the aggressive behaviour. To build up a competitive situation in a race game, it is suitable to start fighting for a certain position. When the player is holding the third place in the race, the rival should challenge him for this position. The best way to simulate a position battle is, when the two rivals constantly are switching positions. Therefore the rival has to catch up and drop behind alternatively.

Catch Up State

In the Catch Up State the NPC tries to overtake his rival. Because of unexpted circumstances the Rivalry AI could be far behind. To ensure the AI catches up fast, a second rubber band system has been added, which is extremer than the normal one. At this point the Rivalry AI starts to cheat. Most players prefer an AI that cheats than a brain-dead one. As long as the cheating is not obvious (Scott 2002). Therefore it is allowed to cheat by using the rubber band system, but the speed boost, which is gained through the rubber band, shouldn't be faster than the power up speed bost. Otherwise the player could run with the speed boost and the Rivalry AI would still be faster and could overtake him without using any power up.

Drop Behind State

The Drop Behind State is similar to the catch up state. As soon as it's the players turn to take the lead, the Rivalry AI switches into the Drop Behind State. When the NPC enters the state, the additional rubber band system is started, which begins to slow down the AI. The user should be able to catch up and overtake his rival.

4.3.2.7. Cheer Up Rival State

In the Cheer Up State the emotion Schadenfreude should be triggered in the rival. The Rivalry AI built up aggression and hate in the Annoy Rival State and the best way to cheer up an opponent, who hates the Rivalry AI, is by creating situations with Schadenfreude. The basic sub states are the same as the sub states from the Racing AI State.

Make Mistakes

One option to amuse the rival is to start mistakes. It is not allowed to use this option very often. If it is used too often, the behaviour is predictable and thus destroys the Schadenfreude, because the element of surprise in the actions is lost. An example for a mistake could be to drive

off a cliff or purposely drive into a mine. Another possibility is to drive at full speed into a curve and crash. This can only happen, when the AI is in the Stay Behind State and a curve is coming up. When the rival starts to break to slow down for the curve, the AI starts to accelerate to full speed. The AI will overtake the rival on high speed, the rival will get annoyed for a brief moment and in the next moment the AI crashes into a barrier of the curve. This triggers Schadenfreude in the player.

Drive Close To Cliff

Another approach is to offer the player the chance to execute the Schadenfreude on his own. For example when there appears a cliff along the racetrack, the AI could drive close to the edge, if the rival is behind him and in sight. Now it is the decision of the rival, if he tries to attack the AI and throw it off the cliff. Another example is to drive in the rivals shooting range, if he has power ups collected. This lets the rival feel Fiero because he triumphed over the AI.

Power Ups

The last idea concerns the power ups. For example the AI can start to clear the road for the rival. When a third racer drives in front of the rival, the Rivalry AI should attack the third racer. This means the rival is in shooting range of the AI, but the AI spares the rival and attacks the character in front of him. Another option is to attack the rival right after he activated a defensive power up. This brings up a superior feeling in the rival.

4.4. Modified µ-SIC System

It was already explained how the mood system and the personality module work together in the Rivalry AI. The next topic is how the relationship system is modified for the Rivalry AI and what the relationship rules are to trigger a rivalry.

4.4.1. Relationship System

In the µ-SIC System the relationship system consists of five categories. For the concept of the Rivalry AI his number was reduced to three: Like/Dislike, Interestedness and Dominant/Submissive. The other two categories are not relevant for the concept of the Rivalry AI. There are certain rules for the relationship system required to start the rivalry. The following rules were implemented in the Rivalry AI of the fun racer. The player must dislike the opponent, this means the value of the like variable has to be negative. The second rule is that the player has to be either dominant or interested in the opponent. If at least one of the two categories is positive and the like category is negative, the relationship rules allow the rivalry to start.

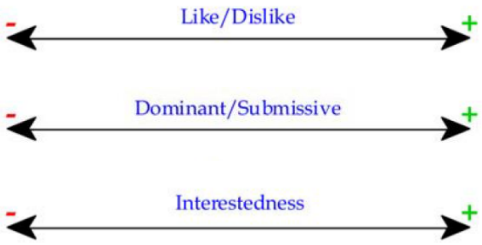

Fig. 34: The three relationship categories used for the Rivalry AI

4.4.2. Emotional Events

In order to change the mood of a game agent, there have to occur certain events, which affect the current emotional state of the agent. Each event changes the valance in positive or negative direction and the arousal according to the fact, if the event is exciting or boring. The following event list was implemented in the fun racer BareFoot.

4.4.2.1. Positive Events

The following events cheer up the character.

- **Hit opponent**
  When a character attacks an opponent with a power up and successfully hits him
- **Fend off attack**
  When the opponent attacks the character and the character has a defensive power up activated
- **Throw opponent off track**
  If the character pushes an opponent off the track or throws the opponents off the track using an attack
- **Block opponent**
  When the opponent drives behind the character in a defined ROI and the character blocks him from overtaking
- **Rival spared character**
  When the character is in the attacking/shooting range of an opponent, but the opponents attacks another creature and spares the character. This event is only considered with an ongoing rivalry.
- **Rival drives into wall**
  The character drives in lead and the opponent accelerates to full speed, when a curve is coming up. The opponent is too fast for the curve, overtakes the character and drives into a track boundary. This event is only available with an ongoing rivalry.

- **Snatched power up away from opponent**
  If the character manages it to snatch a collectable power up away from his opponent. The opponent has to drive in a ROI behind the character in the moment, when the character collects the power up
- **Overtake opponent**
  When the character is able to win back the leading position and overtakes an opponent
- **Win race**
  The character cheers up, when he finishes better ranked than his rivalry. This event is only available with an ongoing rivalry.

4.4.2.2. Negative Events

The following events annoy the character.

- **Get hit**
  When an opponent successfully attacks the character
- **Miss attack**
  When the character attacks the opponent, the opponent has activated a defensive power up and fends off the attack or the character simply misses his attack
- **Fell off track**
  The character drives off the track by mistake and has to respawn to get back to track
- **Get pushed off track**
  When the character falls off the track, forced by an attack of an enemy or an opponent rammed the character off the track
- **Get blocked**
  The character tries to overtake, but the opponent blocks his way
- **Opponent stole power up**
  When an opponent drives in a ROI in front of the character and collects a power up
- **Get overtaken**
  If an opponent overtakes the character
- **Lose race**
  After finishing the race, the character is worse ranked than his rival. This event is only possible, when a rivalry has already started.

4.4.2.3. Neutral Event

The Rivalry AI considers to converge the current valance value towards zero, if nothing is happening. For example in a race where the character isn't attacked or has seen an opponent for a certain time, the valance and arousal is reduced. Therefore a timer is reset, whenever an emotional event occurs. This timer constantly reduces the valance and arousal value. The current valance converges towards zero, no matter if the value is positive or negative. The current arousal is decreased and can fall below zero, this means the character gets bored, when nothing is happening.

4.4.3. Socially Interactive Events

The second category of events are socially interactive events. In socially interactive events have to be at least two game agents involved. Each event affects one to three relationship categories. The events used in BareFoot are listed below. The most events are identical to the emotional events, which affect the mood system.

4.4.3.1. Positive Events

The following events affect at least one of the three character's relationship values. All stated events in the list below increase or at least don't change the like value of the relationship. The most events are congruent to the emotional events.

- **Hit opponent**
  When the character successfully attacks an opponent, the opponent loses dominance and the character loses interest in him. The like value remains unchanged.
- **Fend off attack**
  If the character prevents an opponent from an attack, the opponent becomes less dominant. The other values remain untouched.
- **Push enemy off track**
  When the character manages to push an enemy off the track, the character gains dominance and loses interest in the opponent.
- **Rival spared character**
  When the rival is able to attack the character, but chooses to attack another opponent, the character starts to like him and loses interest in the enemy.
- **Rival drives into wall**
  When the rival makes a mistake, he loses dominance over the character.

- **Opponent drives in Field of View (FOV)**
  When an opponent drives in the FOV of character, the character gets interested in the opponent.
- **Better ranked than rival**
  When the character is better ranked as his rival, he gains dominance over the rival.
- **Overtake Opponent**
  When the character manages to overtake an opponent, he becomes more dominant.
- **Win race**
  When the character finishes the race and is better ranked than his rival, he increases his dominance and becomes less interested in the rival.

4.4.3.2. Negative Events

The negative events affect at least one of the character's relationship values. To be defined as negative events they have to affect the like value negatively. Also the negative events are congruent to the negative emotional events.

- **Get hit**
  When someone successfully attack the character, the dominance and the interestedness of the opponent rises.
- **Get pushed off track**
  When someone pushes the character off the track and the character needs a respawn to get back on track, the opponent gets more dominant and the character gets interested in the opponent.
- **Get blocked**
  When an enemy blocks the character, the character gets interested in him.
- **Opponent snatched power up away**
  By snatching a power up away from the opponent, the character's interestedness in the enemy increases.
- **Lose race**
  When the rival finishes the race better ranked than the character, the opponent becomes dominant and the character gets interested in his rival.

4.4.3.3. Neutral Event

Similar to the emotional events, when nothing happens the relationship values converge towards zero. When there is no interaction between characters over a defined period of time, the relationships cool down.

4.4.4. Switch Annoy and Cheer Up Rival

The rules to switch from the Annoy Rival State to the Cheer Up Rival State are similar to the mood requirements to start a rivalry. As soon as the rivalry has started, the mood values are plotted in the second quadrant of the mood system graph. When the current valance exceeds the negative threshold of -1.5 it is time to cheer up the rival. At the moment when the valance decreases under a value of -0.5, the rivalry switches back to the Annoy Rival State. With this approach the Rivalry AI is continuously adapting according to the players emotional state. This personalizes the game and increases the emotional sensations, because the game reacts to the player's behaviour (Izard 2009).

5. Results

In the introduction of this master thesis we have defined three hypotheses, which the concept of the Rivalry AI should accomplish. H_1: The first hypothesis is that the user remembers and recognizes the Rivalry AI after playing the game. H_2: The second hypothesis is that the Rivalry AI sticks out compared to the normal AI. H_3: The last hypothesis is that we are able to model the player's emotional state with the µ-SIC System. To be able to evaluate these hypotheses, user tests were conducted.

The user tests were executed as A/B tests. An A/B test is a test with two variants, A and B. The participants are split into two groups, into the control and treatment group. This type of test is used to test hypothesis and has therefore the technical term two-sample hypothesis testing. (Hartung, Elpelt, and Klösener 2005)

The participants had to play one race in the game BareFoot. One race consists of three laps. Four creatures attend the race, three creatures controlled by the AI and a human controlled the last one. The treatment group tested the game, in which one AI opponent started a rivalry with the player. The other two AI controlled creatures used the normal racing AI. This group tested the Rivalry AI. The control group played the game without the Rivalry AI. Therefore the player had three normal AI controlled opponents.

As mentioned in the description of the game BareFoot, a creature consists of five different body parts. Each player can individually construct his creature. The creatures for the AI were randomly constructed. Each creature got an avatar assigned. Every match had the three identical avatars for the AI opponents. Each opponent got one of the three avatars randomly assigned. The Rivalry AI for the treatment group was chosen randomly out of these three avatars.

The rivalry between the player and an AI opponent didn't build up during the match, but started after 30 seconds into the game. Which one of the three opponents started the rivalry with the user was decided randomly. The reason, to automatically start the rivalry right after the race start, was to assure that there will be a rivalry in the race. Otherwise it could have happened, that there aren't enough interactions between the player and the opponents. Thereby the rivalry wouldn't start and we couldn't assure that there is a satisfactory amount of participants in the treatment group. Another problem that could appear, without starting the rivalry automatically after the race start, is that the rivalry could start too late. When the rivalry would start in the last lap of the race, there wouldn't be enough time to leave a permanent impression to the user. Thereby the participants would be categorised in the treatment group, but wouldn't

be able to answer questions concerning the Rivalry AI, because they had no time to notice the ongoing rivalry.

After finishing the game, the participants had to answer a questionnaire. Additionally the game stored match relevant data in an xml file. The file contained information like a snap shot of the µ-SIC System at the exact moment, when the characters finished the race.

5.1. User Tests

49 people participated in the user test. 26 tester are categorized into the treatment group and 23 into the control group. 81 % of user test are male participants and 19 % are female users. The age range of the subjects lays between 17 and 62. The average age is 28.2 years and the median age is 26 years.

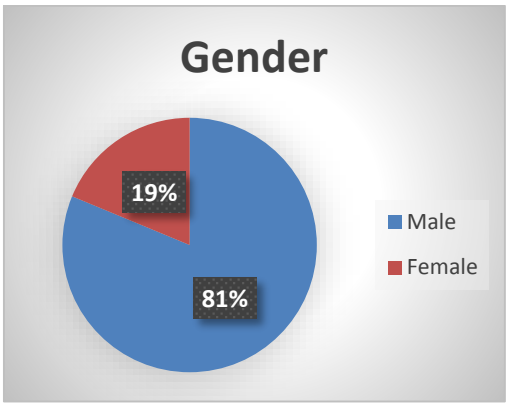

Fig. 35: Gender arrangement of the user test

5.2. Hypothesis one: Recognize Rivalry AI

Hypothesis one predicts that the player remembers and recognizes the Rivalry AI after playing the game. To evaluate this hypothesis the questionnaire has five different avatar images and the subjects must decide, which one was an opponent in the race. Three avatars had really participated in the race, the other two images displayed avatars, which the participants haven't seen anywhere before. One of the three avatars is randomly assigned to be the Rivalry AI. It is important, to randomly choose one of the three avatars to be the Rivalry AI, otherwise one avatar could have been more memorable than the others and could distort the results. The subjects had to choose for each avatar, which one was an opponent in the race.

Fig. 36: The three participating avatars

Fig. 37: The two avatars, which didn't participate in the test

In order to get a comparative value, the first analysis concerns the two avatars, which didn't start in the race.

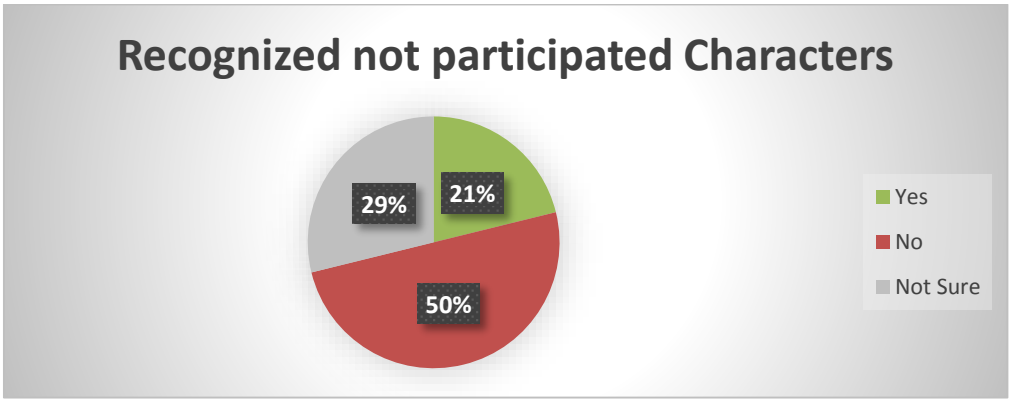

Fig. 38: The recognition rate of the two avatars, which weren't participating in the race

The subjects answered with 50 % correctly, that the two avatars didn't start in the race. 29 % were uncertain and 21 % answered the question wrong. This is the first target value, which the recognition rate of the Rivalry AI has to exceed to support the first hypothesis.

When the question, if this avatar was participating in the race, was answered with “Yes” or “Not sure” the participants got the follow up question on how sure they are, that this avatar was an opponent in the last race. 69 % of the participants answered the question, if the avatar of the Rivalry AI was participating in the race with a “Yes”. Only 12 % didn’t recognize the Rivalry AI and answered with “No”. 19 % of the participants were uncertain.

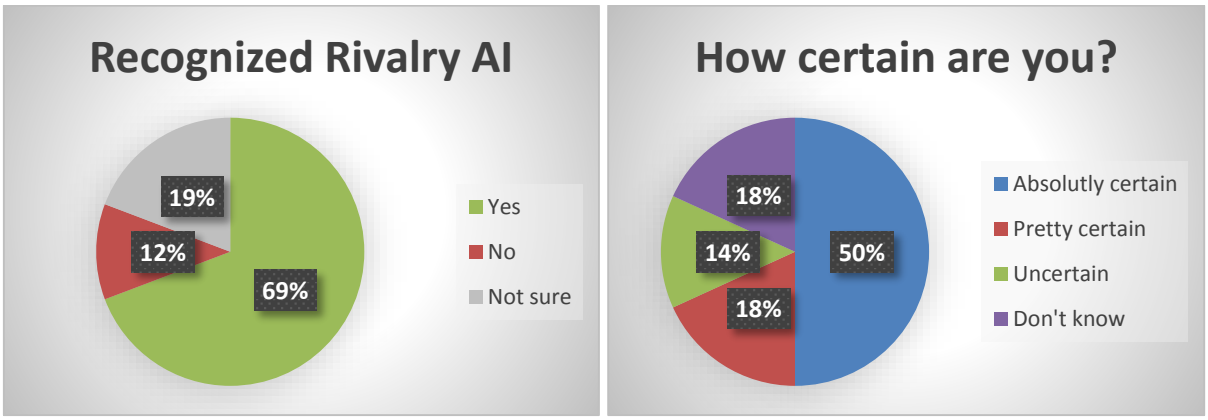

Fig. 39: Recognition rate of the Rivalry AI and how certain the subjects were

When we compare these results to the null hypothesis, which would be, when the subject would just guess for each avatar, if he had participated. The null hypothesis would have a chance of 60 %, because three avatars out of five have participated in the race. Therefore the result of 69 % is just slightly above the null hypothesis.

However, when each avatar image is checked individually, it turns out, that the Captain and the Hippie avatar were recognized significantly more often than the Sergeant avatar.

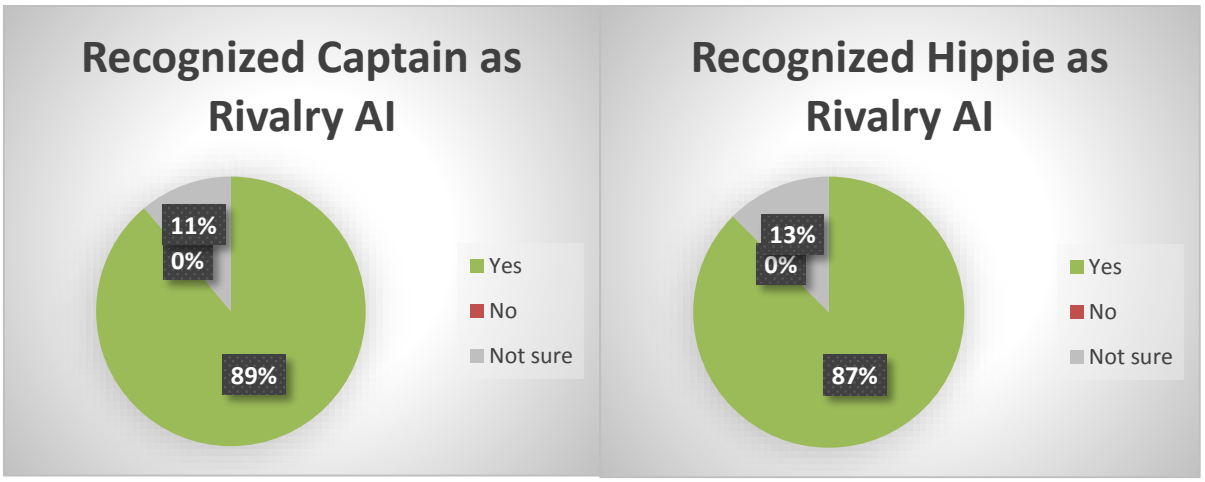

Fig. 40: Individual recognition rate of Captain and Hippie

Both, the Captain and the Hippie avatar, are with 89 % and 87 % precisely better than the null hypothesis. This supports the Hypothesis one, that the player memorizes and recognizes the Rivalry AI. However, the Sergeant avatar decreases the overall recognition rate.

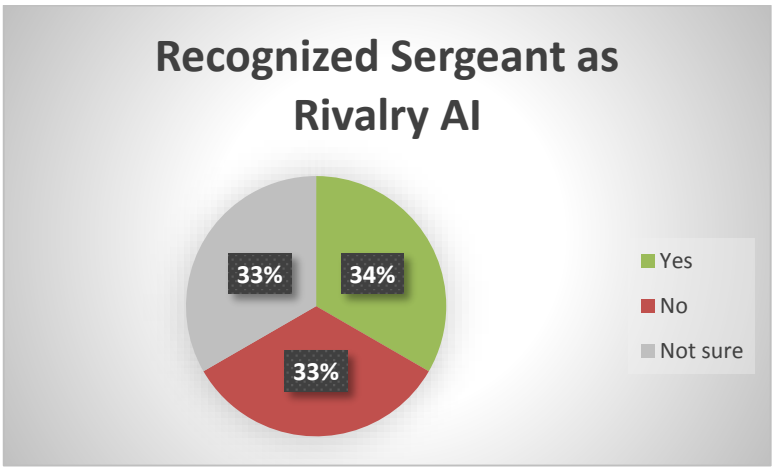

Fig. 41: Recognition rate of the Sergeant avatar

With only 34 % the recognition rate is significantly lower than the null hypothesis rate. Considering the recognition rate of each avatar, we see that the Sergeant avatar is significantly less often recognised than the other two avatars. Both at the treatment group and the control group, the participants didn't as often recognize the Sergeant as the Hippie or the Captain as an opponent in the race.

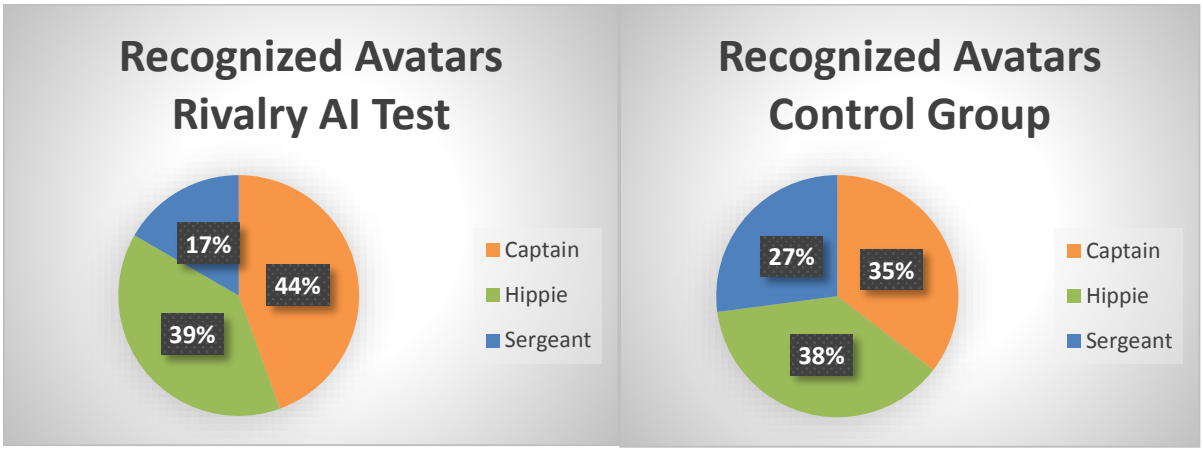

Fig. 42: In both tests, the Sergeant avatar wasn't as often recognized as the Hippie or the Captain avatar

This supports the hypothesis, that the Captain avatar isn't as memorable as the other two avatars. Therefore we limited the recognition rate of the Rivalry AI to the Captain and Hippie avatar.

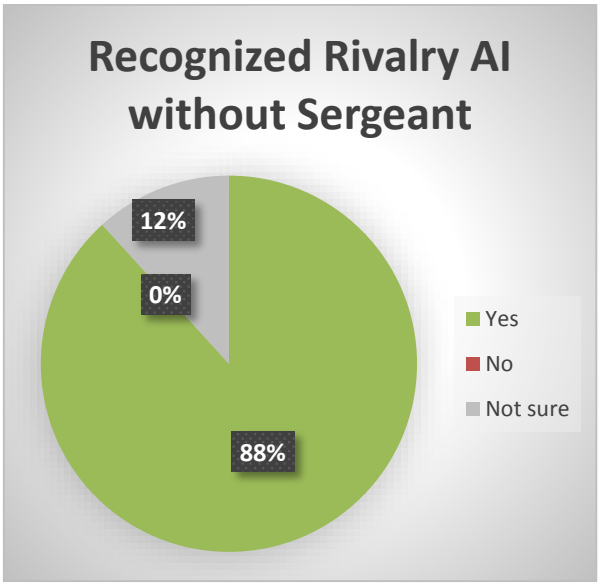

Fig. 43: Recognition rate of the Rivalry AI, ignoring the Sergeant avatar

When we ignore the deviations caused by the Sergeant avatar, we are able to confirm the Hypothesis one, that the player memorizes and recognizes the Rivalry AI.

5.3. Hypothesis two: Rivalry AI sticks out

The Hypothesis two predicts, that the Rivalry AI sticks out compared to the normal AI. The user test brought up different figures to evaluate this thesis. The first one is a comparison between how often the Rivalry AI was recognized compared to the other two normal AI characters.

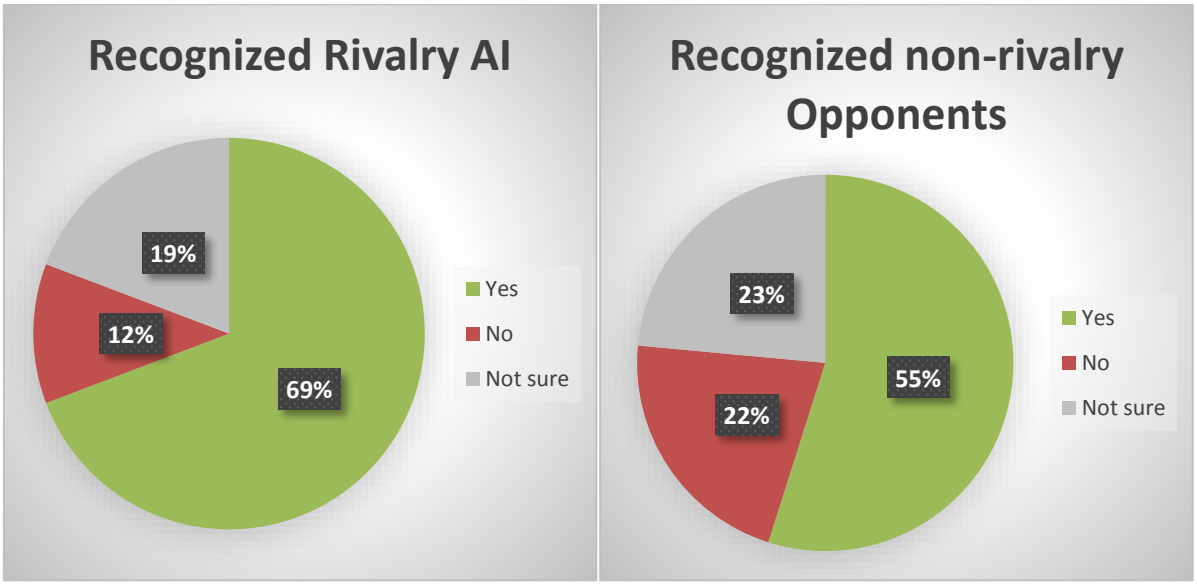

Fig. 44: Comparison recognition rate Rivalry AI and normal AI

Compared to the normal AI opponents the Rivalry AI was recognized significantly more often. With a 14 % better recognition rate the Rivalry AI is more noticeable and more memorable compared to the normal AI in the same race. Even without paying attention to the previously found fact, that the Sergeant avatar is decreasing the recognition rate. On the other hand, if the Rivalry AI recognition rate is compared to the opponent recognition rate from the control group, where only normal AI opponents participated in the race, the recognition rate is identical.

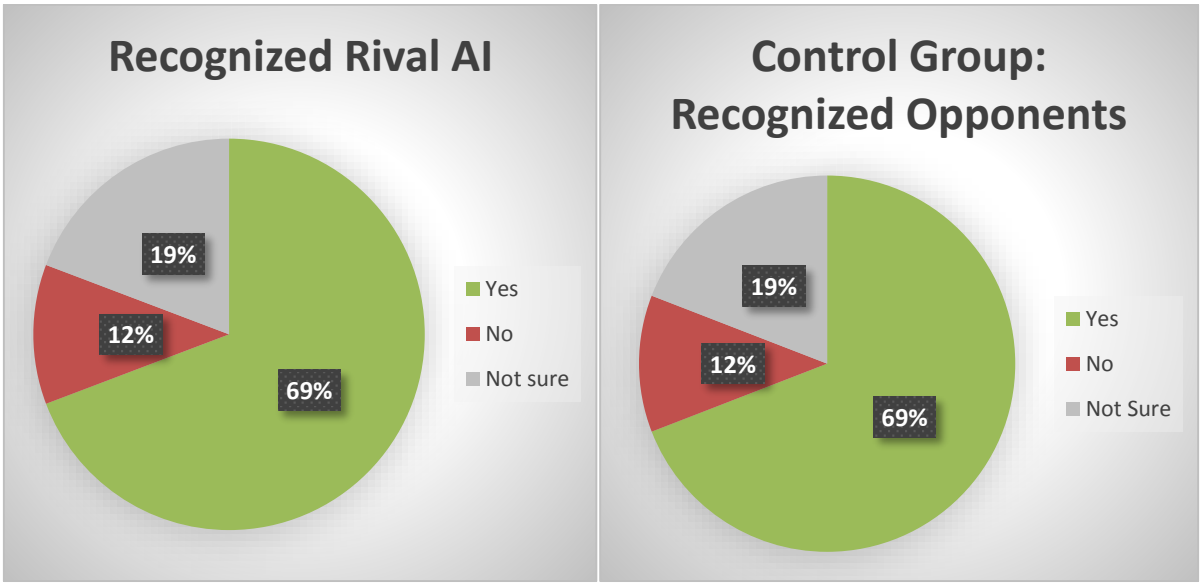

Fig. 45: Comparison recognition rate Rivalry AI and normal AI from the control group

Although the recognition rate of the Rivalry AI compared to normal AI opponents from the same race is significantly better, the Rivalry AI rate is identical to the recognition rate of the normal AI from the control group. The control group played the game without a Rivalry AI, all opponents were controlled by the normal AI. This fact doesn't support the hypothesis. When the Sergeant avatar effect is considered and the data of the treatment group and the control group is compared, without considering the Sergeant opponent, recognition rate is better for both groups. The recognition of the Rivalry AI is significantly better than the rate from the control group. This result supports the hypothesis.

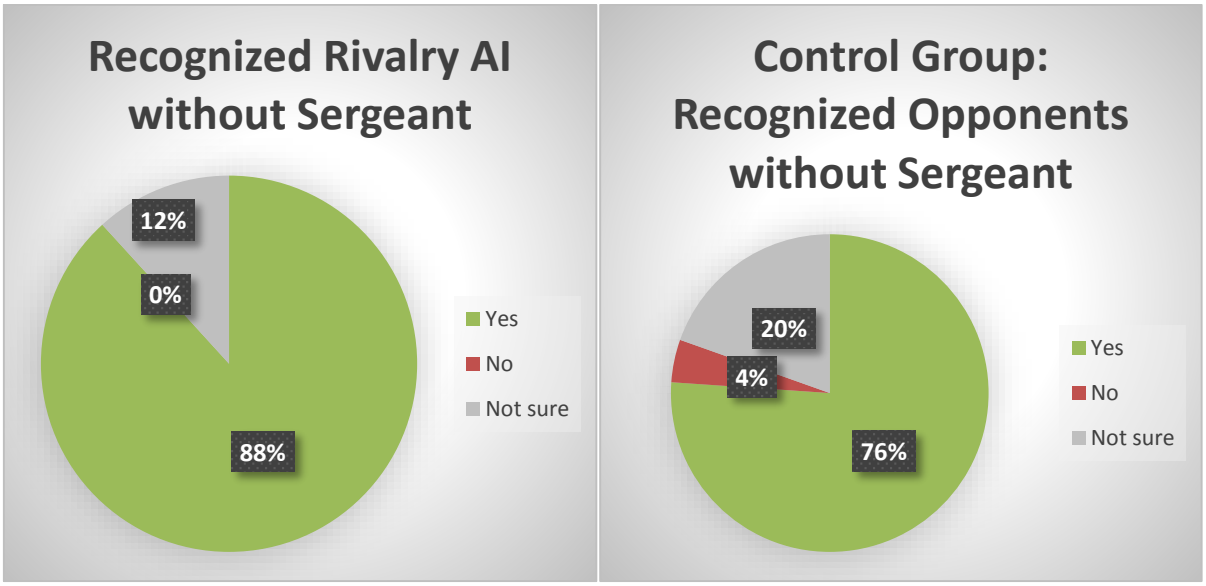

Fig. 46: Comparison recognition rate without the Sergeant avatar

The next option to evaluate the second hypothesis is how many subjects named the Rivalry AI, when asked, if one opponent attacked the player excessively.

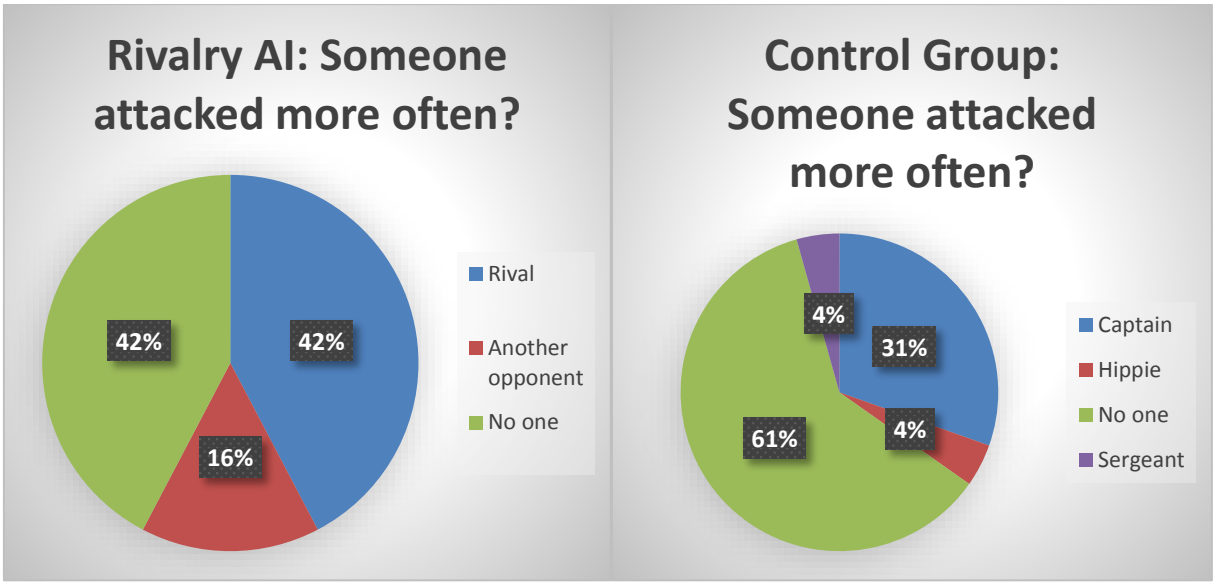

Fig. 47: Comparison attack rate of treatment and control group

42 % noticed that the Rivalry AI attacked the subject more often, compared to the other opponents. 16 % experienced another opponent than the Rivalry AI had attacked excessively the subject. By comparing those two rates, the rate of the Rivalry AI is explicitly higher than the rate of the normal AI. 42 % didn't noticed any opponent, who attacked the subject more often. Compared to the control group, where 61 % didn't experience any opponent, who excessively

attacked the subject. The null hypothesis is defined as the 61 %, where the subjects chose there was no one who attacked him more often than other opponents. Compared to the null hypothesis the 42 % recognition rate is explicitly better. Thus supports the hypothesis two.

The third analysis of the second hypothesis concerns the question, if the subject perceived any opponent as his personal competitor during the race. The participants could choose between the Rivalry AI, the two normal AI opponents, the two not participating avatars and nobody.

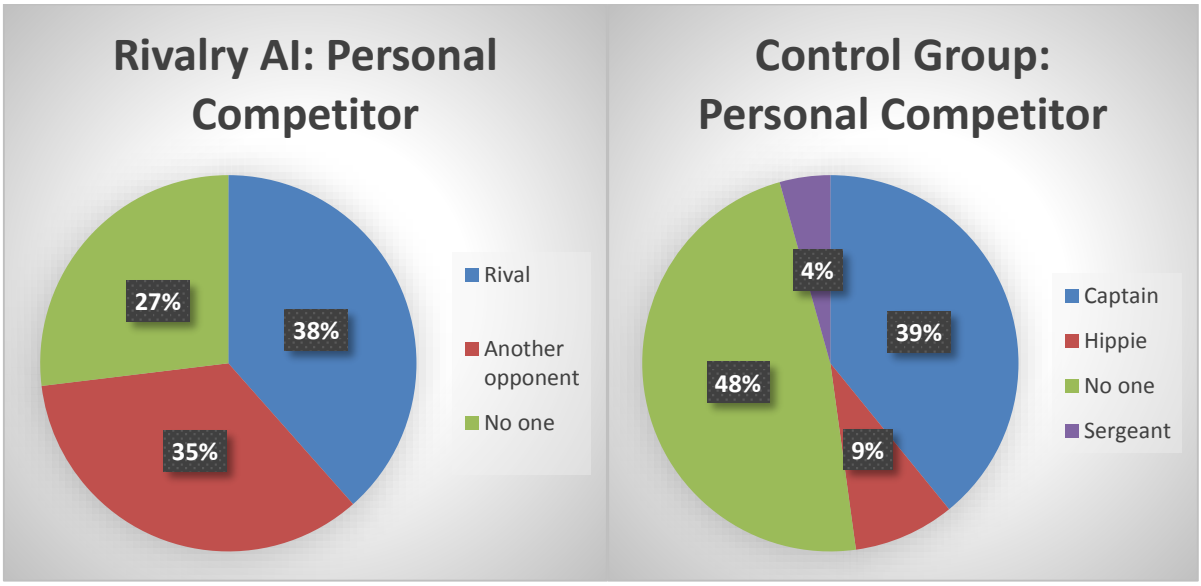

Fig. 48: Comparison personal competitor

In the treatment group only 27 % didn't experience one of the three opponents as a personal competitor. In contrast in control group 48 % didn't experience a personal competitor. On the other hand roughly the same amount of subjects experienced the normal AI as their personal competitor compared to the Rivalry AI.

By limiting the results of the last to analyses to subjects, which declared themselves as advanced or professional racing game players, the results turns out more promising for the second hypothesis.

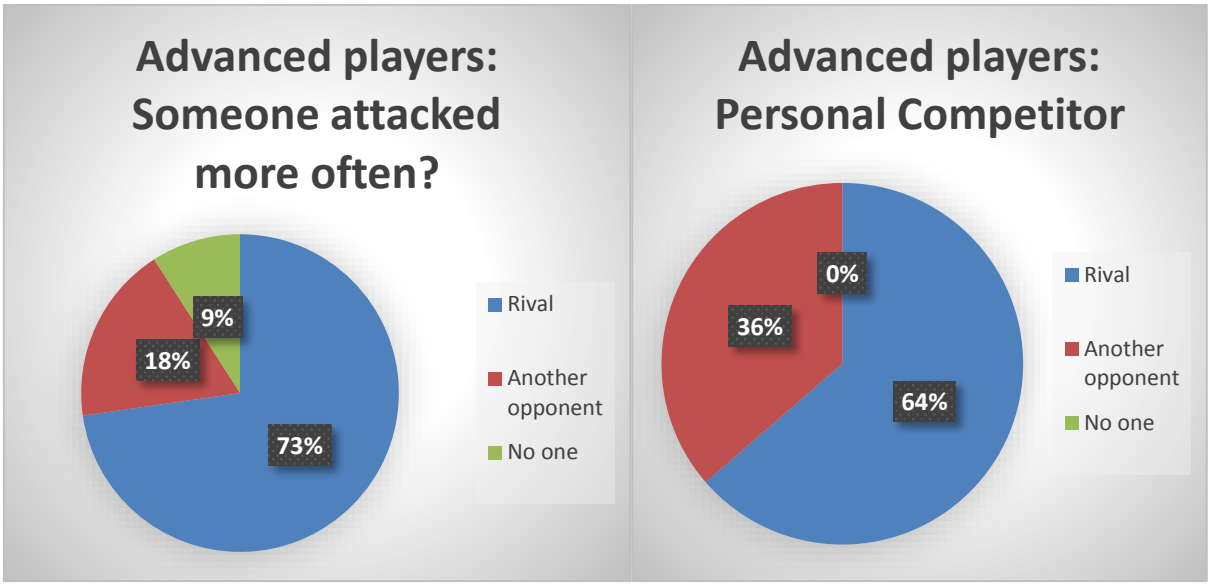

Fig. 49: Results of the advanced categorized players

64 % of the advanced players experienced the Rivalry AI as their personal competitor. By recognizing the Rivalry AI twice as often as an excessive attacker and recognizing it nearly twice as often as their personal competitor, the results confirm the hypothesis two partially. The hypothesis is true, when we ignore the Sergeant as opponent and when we limit the test groups to advanced players.

5.4. Hypothesis three: Model Player's Emotions with µ-SIC System

The last hypothesis states that it is possible to model the player's emotional state and his relationships by using the µ-SIC System. The first step to evaluate this hypothesis, is comparing the tracked values of the mood system with the answers to the question, if someone attacked the subject excessively.

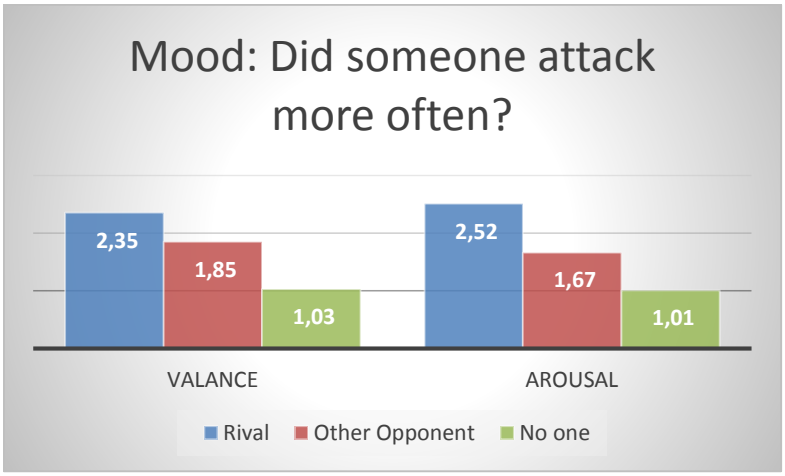

Fig. 50: Comparison between tracked mood of player and who attacked the player excessively

We have to note, that we used the absolute amount of the valance values for representational purposes. The results show that the subjects were more annoyed, when the Rivalry AI attacked them excessively. On the other hand the subjects did not notice any excessive attacks, when the valance value was low. The same applies to the arousal value. This supports the hypothesis three. No matter if the excessive attacker is the Rivalry AI or another opponent, when there are a lot of attacks, the subjects were annoyed and aroused.

A second indicator to support the hypothesis is the comparison of the mood values to the question, how the subject perceived the amount of attacks during the race. The subjects had to answer the question, if the amount of attacks was too high, appropriate or too low.

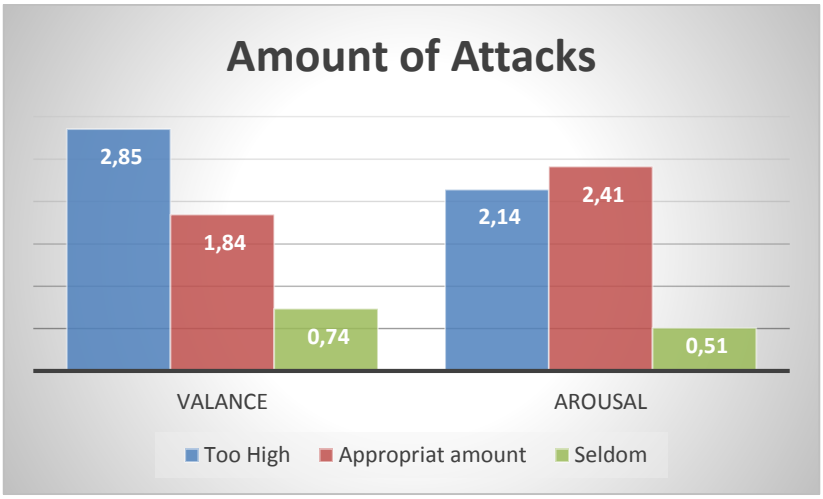

Fig. 51: Comparison between tracked mood of player and the amount of attacks

This comparison shows that the valance and arousal depend on the amount of attacks. When the subject was often attacked, the values were high. When there were too few attacks, the subjects had low valance and arousal values. A possible reason, that the arousal value of the appropriate category is slightly higher than the value of the too high category, is that there are other events than attacks, which affect the arousal of the player. The two shown results support the hypothesis and confirm that it is possible to model the player's mood using the µ-SIC System.

The second part of the hypothesis concerns the relationship model of the µ-SIC System. Therefore we compare the tracked relationship values and the question, what was the emotional relationship to each opponent. There were five choices: the subject appreciated the opponent, the opponent was an enrichment for the race, the subject developed an antipathy against the opponent, the player hated the opponent or the subject isn't sure.

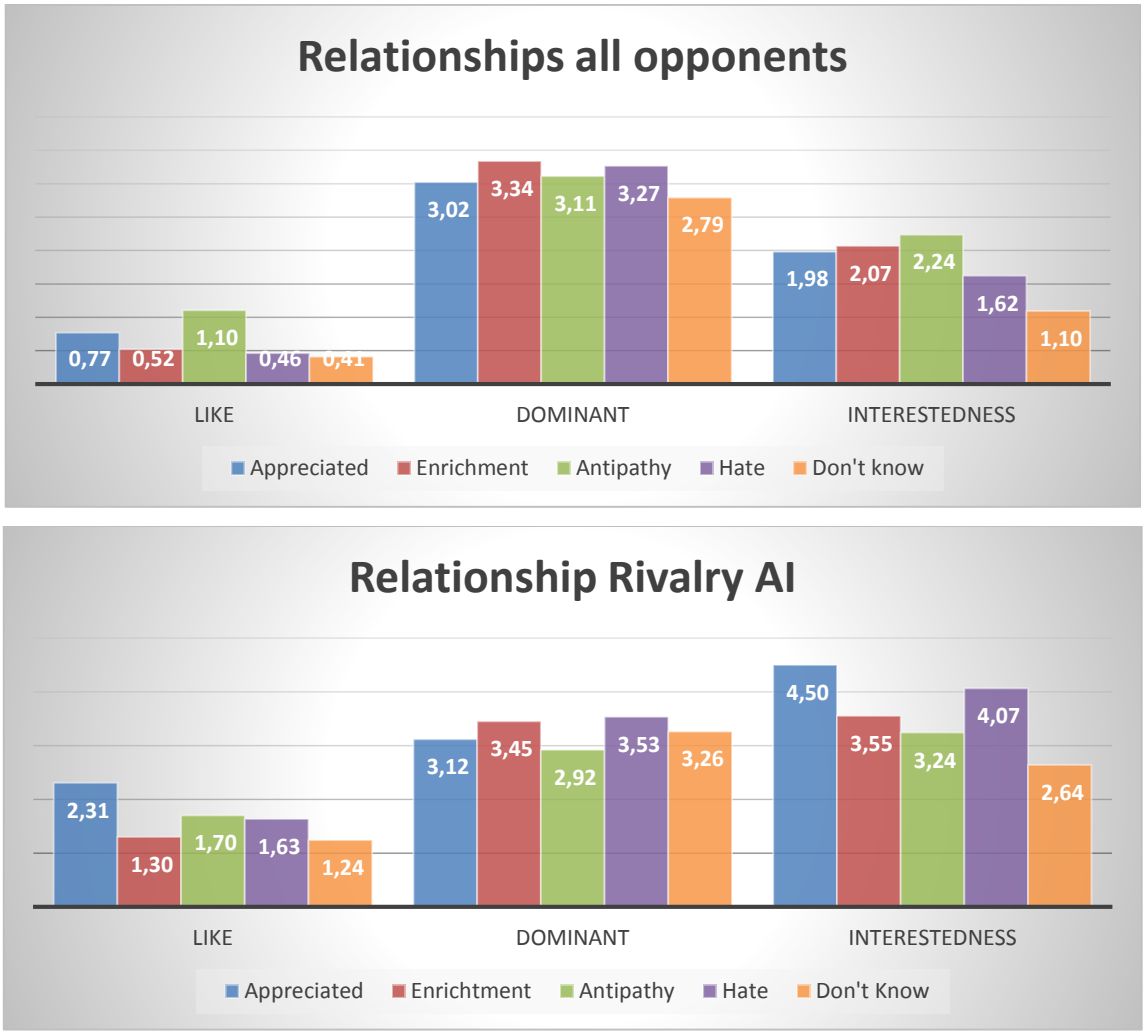

Fig. 52: Tracked relationships of player compared to the emotional relationship to the opponents

Neither the comparison to the Rivalry AI nor to all opponents brought up any useful results. This disproves the hypothesis that it is possible to model the player's relationships using the µ-SIC System. Therefore the hypothesis three, which states that it is possible to model the player's emotional state with the µ-SIC System, is only partially true.

5.5. Result

The hypothesis one is proven, which states that the player recognizes the Rivalry AI. The Rivalry AI has a higher recognition rate than the two not participating characters. The rate is higher than the random choice chance. The recognition reaches nearly 90 %, when the sergeant character is ignored.

The second hypothesis is only partially proven, which claims that the Rivalry AI sticks out compared to normal AI opponents. The Rivalry AI is significantly more noticeable and memorable than the two normal AI opponents from the same race. Compared to normal AI opponents from the control group the recognition rate of the Rivalry AI is still better, when the answers concerning the Sergeant avatar are ignored. Compared to the attack rate during the match, the Rivalry AI sticks out and has a better recognition rate as an excessive attacker. On the other hand the Rivalry AI doesn't stick out, when compared to normal AI the personal competitor of the tester. However when the test group is filtered and only advanced and professional players are considered, the results prove the second hypothesis.

The third hypothesis is half proven. The hypothesis states that the emotional state of the player can be modelled with the µ-SIC System. On the one hand it is proven that it is possible to model the players mood using the µ-SIC System. On the other hand it was not possible to model the players relationships with the µ-SIC System.

6. Discussion

One huge problem of the conducted user test was that the test only consists of one race, which takes three laps to finish. This is not really a long time to establish relationships and emotions between game agents. The concept of the Rivalry AI is designed to build up the rivalry slowly over a period of time. Normally the player has time to get to know each opponent. From race to race the rivalry builds up. When the rivalry starts, the player already knows his opponent. During the rivalry the AI annoys and cheers up the player. They have enough time to build up a relationship. In the conducted user tests, the rivalry starts randomly after 30 seconds. The player has no idea who his opponents are. There are less than three laps remaining to attract the player's attention and to get him to remember the Rivalry AI after the match. Therefore the Rivalry AI was balanced to be very noticeable. The balancing of the Rivalry AI and the events were partially obtrusive. The problem to conduct a longer user test with more races was that currently there is just one single race track available. It is not possible to demand from voluntary testers to conduct the survey and play five times in a row the same level. Therefore it was decided to execute the test with just one race.

When interpreting the results it appeared the so called Sergeant avatar effect. The results of the Character with the Sergeant avatar departed from the results of the other characters. There are two possible reasons for this effect. First it could be possible, that the image has less details and accessories than the other avatars. Because of that the Sergeant avatar is less memorable, which results in a lower recognition rate. The other reason could be the questionnaire. The questions to the Sergeant are the last of a five identical question sequence. For each character the subjects have to answer, if the displayed avatar was an opponent in the race. When the subjects answer with yes, three follow up questions appear. When they choose no, they skip these follow up questions. When they reach the same question for the fifth time, if this avatar was an opponent, maybe the subjects got annoyed or bored and just wanted to skip the follow up questions. No matter what are the reasons for the Sergeant effect, by ignoring answers concerning the Sergeant, it was possible to prove the hypothesis.

One reason for the results, that the recognition rate of the normal AI from the control group is identical to the recognition rate of the Rivalry AI, could be the attack amount. The two tests show different results, when asked how they experienced the attack amount during the race.

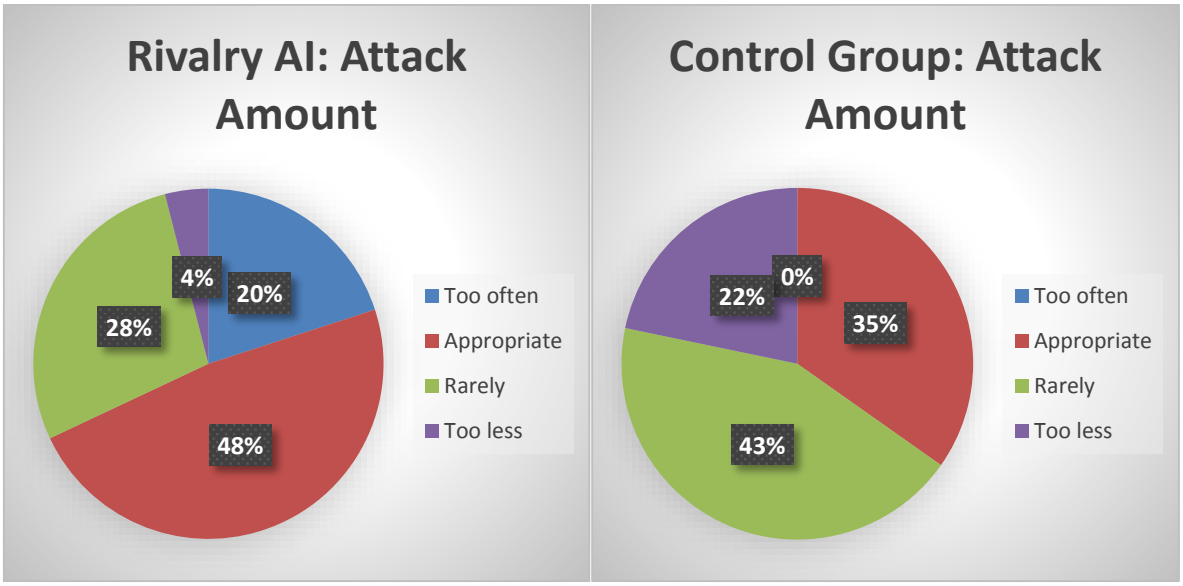

Fig. 53: Attack amount compared Test A and B

As shown in the diagram above, the control group claims that in 65 % of the games there were too little action going on. On the other hand the treatment group experienced in 68 % of the games enough or too much action. This could be the reason that subjects from the control group had enough time to concentrate on other things than fighting with the opponents. For example looking frequently on the mini map, where all avatars of the participating characters are displayed. Therefore the control group could have had the same recognition rate as the Rivalry AI. Another indicator to support this theory is that advanced players did recognize the Rivalry AI as their personal competitor more often. Advanced gamers are more likely to keep the overview of the game and notice more details, like the flying avatars over the creatures.

Another interesting result was that advanced players still chose other opponents than the Rivalry AI as their personal competitor. This isn't a negative result for the Rivalry AI, it shows that it is possible to start a rivalry with one opponent, but there are still other opponents participating and interacting with the player. This provides the possibility that a normal player is still a competitor additional to the Rivalry AI. The important result is that the Rivalry AI has a higher recognition rate as personal competitor than the normal AI does.

The problem with the third hypothesis, which stated that the player's emotions can be modelled with the µ-SIC System, was that the event parameters weren't enough balanced nor tested. With a set of more balanced parameter it could be possible to model the player's relationship with the µ-SIC System. It is not that different to the mood system, which was able to model the player's emotional state. Therefore it should also be possible to model the player's relationships. Another problem is that it is hard to express emotions only using a questionnaire. It

would be better to run a more complex user test like Lazzaro (2004) did to find the four key emotions in games.

The results of the user test were very promising. They proved one hypothesis and at least partially proved the other two hypotheses. The problems with the player modelling of the relationship should be possible to correct, similar to the mood system, which already was proven to work.

7. Conclusion and Outlook

The concept to personalize a game brings a lot of benefits. For example a personalized game builds virtual worlds with personal relevance conditions for the player. A player in such conditions will exhibit stronger emotional reactions. (Darley and Lim 1992) The Rivalry AI is a method to personalize a game. The results of the conducted user tests proved or at least partially proved the hypothesis, which were constructed for the Rivalry AI.

There are a lot of possibilities to improve the concept of the Rivalry AI. For example it would be possible to create a long term rivalry. In BareFoot each player needs to register an account at the BareFoot database. To establish a long term rivalry it would be necessary to store the figures of the relationships linked to an account. When the player starts the game on the next day, the rivalries will still be there. To assure that the rivalries are able to cool off after some time, the relationship values have to be stored with a time stamp of the last match. When the relationship values are loaded for the next game, the relationship values have to be decreased according to the passed time. Imagine you have played yesterday a Grand Prix and it was highly competitive. The competitor was the Captain and on the next day the heat is still present between the two characters.

Another improvement for the Rivalry AI would be to expand the concept that especially players, which are bad ranked in the race, profit from the Rivalry AI. One idea behind the Rivalry AI is to motivate the player. When a player is in a bad rank and has no chance to win the race, the Rivalry AI should give him a new goal: To beat the Rivalry AI (Vallerand and Losier 1999). The problem of the Rivalry AI is that it attacks excessively the player to build up aggressive behaviour. When the player is fighting for the lead position and has chances to win, the Rivalry AI will slow him down by attacking him. Therefore it is more difficult for the player to win the race, which isn't the goal of the Rivalry AI. Because of that the Rivalry AI should only start, when the player lost the chance to win the game. This would bring a motivation boost to new and inexperienced players and therefore could push the retention rate of the users.

A possible adaption for the Rivalry AI would be to create a Buddy AI. Especially for the genre fun racer this would be a new approach to replace the dusty rubber band concept. The concept of the rubber band to provide Dynamic Difficulty Adjustment (DDA) for the player, just alters the acceleration and maximum speed of the characters. The Buddy AI could help a player, who is far behind. Instead of blocking the character, the Buddy AI would offer the player a chance of drafting. When they catch up the AI starts to attack the opponents and push the

player forward in the racing field. This concept would trigger the cognitive emotions of cooperative situations, which wouldn't trigger aggressive behaviour, but friendly and cooperative behaviour. The Buddy AI could provide DDA and a spectacular method to catch up.

An improvement to the player modelling of the user's emotional state and to the smack talk between a player and a NPC would be to give the player the ability to start a simple dialogue with the opponent. For example when the race is over and the NPC starts to smack talk, the player could use the four buttons on his controller to response. Each button triggers an expression, which belongs to one of the following four categories: Insult, express antipathy, congratulate or cheer up. The same interaction is possible when the player gets stunned by an opponent. While he isn't able to move, the user can express his feelings. This would provide an additional and direct possibility to express the emotions of the player. Further the current mood and the relationship of the user can be better modelled. Another improvement additional to the verbal feedback could be avatar animation to certain events. In the user test was already implemented, that the avatar over the creature's head performed a jump and made a 360 degree rotation to attract attention. An expansion to the avatar movement could be an animated avatar, where the character points at the user and laughs to insult him or punches against the wall to express anger. This would attract more attention to certain events and could therefore improve the player modelling.

Concluding we state that the Rivalry AI works, but still needs some improvements. Starting with the player modelling problem of the player's relationships. Followed by creating a well-balanced set of parameters for the different socially interactive and emotional events. Solving these problems the Rivalry AI could become a finalized concept to personalize a game. In summary the Rivalry AI concept is an easy and efficient way to personalize a game. By using and expanding the µ-SIC System it is fast to implement, easy to understand and non-technical developers like game designers are able to create and maintain the AI created by this system. Although the concept of the Rivalry AI it isn't suitable to use for complex emotional simulations, it is promising concept to personalize casual games.

8. List of Figures

9. List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
DDA	Dynamic Difficulty Adjustment
FOV	Field of View
FSM	Finite State Machine
GAM	General Aggression Model
HFSM	Hierarchical finite state machines
KI	Künstliche Intelligenz
NPC	Non Player Character
PC	Player Character
QA	Quality Assurance
ROI	Region of Interest

10. Glossary

BareFoot	A fun racer in which the Rivalry AI is implemented
Fiero	The term Fiero is Italian and means personal triumph
Game Agent	Every fictional character in the virtual world, whether it's a player controlled character or the non player controlled character
Schadenfreude	Schadenfreude is a German word and expresses the joy over misfortune or failure of an opponent

11. Bibliography

Anderson, Craig A., and Brad J. Bushman. 2002. 'Human Aggression'. *Psychology* 53 (1): 27.

Anderson, Craig A., and Melissa Morrow. 1995. 'Competitive Aggression without Interaction: Effects of Competitive versus Cooperative Instructions on Aggressive Behavior in Video Games'. *Personality and Social Psychology Bulletin* 21 (10): 1020–30.

Anderson, Craig A., Akiko Shibuya, Nobuko Ihori, Edward L. Swing, Brad J. Bushman, Akira Sakamoto, Hannah R. Rothstein, and Muniba Saleem. 2010. 'Violent Video Game Effects on Aggression, Empathy, and Prosocial Behavior in Eastern and Western Countries: A Meta-Analytic Review.' *Psychological Bulletin* 136 (2): 151.

Bakkes, Sander, Chek Tien Tan, and Yusuf Pisan. 2012. 'Personalised Gaming: A Motivation and Overview of Literature'. In *Proceedings of the 8th Australasian Conference on Interactive Entertainment: Playing the System*, 4. ACM. http://dl.acm.org/citation.cfm?id=2336731.

Berkowitz, Leonard. 1962. 'Aggression: A Social Psychological Analysis.' http://psycnet.apa.org/psycinfo/1963-06518-000.

Berkowitz, Leonard. 1989. 'Frustration-Aggression Hypothesis: Examination and Reformulation.' *Psychological Bulletin* 106 (1): 59.

Biasillo, Gari. 2002a. 'Representing a Racetrack for the AI'. *AI Game Programming Wisdom. Charles River Media*, 439–43.

Biasillo, Gari. 2002b. 'Racing Al Logic'. *AI Game Programming Wisdom*, 444.

Castronova, Edward. 2002. 'On Virtual Economies'. http://papers.ssrn.com/sol3/papers.cfm?abstract_id=338500.

Chan, Elaine, and Peter Vorderer. 2006. 'Massively Multiplayer Online Games.' http://psycnet.apa.org/psycinfo/2006-05034-006.

Cheng, Ho-Lun, and Chek Tien Tan. 2007. 'Personality-Based Adaptation for Teamwork in Game Agents'.

Cook, Daniel. 2009. 'Lost Garden: Testosterone and Competitive Play'. http://www.lostgarden.com/2009/11/testosterone-and-competitive-play.html.

Darley, William K., and Jeen-Su Lim. 1992. 'The Effect of Consumers' Emotional Reactions on Behavioral Intention: The Moderating Role of Personal Relevance and Self-Monitoring'. *Psychology & Marketing* 9 (4): 329–46.

Eckman, Paul. 2003. 'Emotions Revealed'. *Recognizing Faces and Feelings to Improve*.

Elliott, Clark. 1994. 'Research Problems in the Use of as Allow Artificial Intelligence Model of Personality and Emotion'. http://www.aaai.org/Papers/AAAI/1994/AAAI94-002.pdf.

Elliott, Clark Davidson. 1992. 'The Affective Reasoner: A Process Model of Emotions in a Multi-Agent System'. http://dl.acm.org/citation.cfm?id=142741.

Eysenck, Hans Jurgen, and Stanley Rachman. 1965. *The Causes and Cures of Neurosis (Psychology Revivals): An Introduction to Modern Behaviour Therapy Based on Learning Theory and the Principles of Conditioning*. Routledge. http://books.google.at/books?hl=en&lr=&id=xnwuAgA-AQBAJ&oi=fnd&pg=PP1&dq=The+Causes+and+Cures+of+Neurosis&ots=qDx7XhM5v8&sig=t2M4Kx-kYbz5nItYGmPSkxrNss8.

Fairclough, Chris, Michael Fagan, Brian Mac Namee, and Pádraig Cunningham. 2001. *Research Directions for AI in Computer Games*. Trinity College Dublin, Department of Computer Science. http://www.tara.tcd.ie/xmlui/handle/2262/13098.

Frijda, Nico H., Andrew Ortony, Joep Sonnemans, and Gerald L. Clore. 1992. 'The Complexity of Intensity: Issues Concerning the Structure of Emotion Intensity.' http://psycnet.apa.org/psycinfo/1992-97396-003.

Fu, Dan, and Ryan Houlette. 2004. 'The Ultimate Guide to FSMs in Games'. *AI Game Programming Wisdom* 2 (2003): 283–302.

Hartung, Joachim, Bärbel Elpelt, and Karl-Heinz Klösener. 2005. *Statistik: Lehr-Und Handbuch Der Angewandten Statistik; Mit Zahlreichen, Vollständig Durchgerechneten Beispielen*. Oldenbourg Verlag. http://books.google.at/books?hl=en&lr=&id=DKcVnmCeskkC&oi=fnd&pg=PA1&dq=Statistik++Lehr-+und+Handbuch+der+angewandten+Statistik&ots=VnuGEIBkBs&sig=MwCOMic63eO0hPoS6um2Py4cH_w.

Houlette, Ryan, Daniel Fu, and David Ross. 2000. *Towards an AI Behavior Toolkit for Games*. Defense Technical Information Center. http://www.aaai.org/Papers/Symposia/Spring/2001/SS-01-02/SS01-02-011.pdf.

Izard, Carroll E. 2009. 'Emotion Theory and Research: Highlights, Unanswered Questions, and Emerging Issues'. *Annual Review of Psychology* 60: 1.

Lang, Peter J. 1995. 'The Emotion Probe: Studies of Motivation and Attention.' *American Psychologist* 50 (5): 372.

Lazzaro, Nicole. 2004. 'Why We Play Games: Four Keys to More Emotion without Story'. http://www.citeulike.org/group/596/article/436449.

Lebowitz, Michael. 1985. 'Story-Telling as Planning and Learning'. *Poetics* 14 (6): 483–502.

Lloyd, Peter, Andrew Mayes, A. S. R. Manstead, P. R. Meudell, and H. L. Wagner. 1984. *Introduction to Psychology: An Integrated Approach*. Fontana London.

Mac Namee, Brian, and Pádraig Cunningham. 2001. *A Proposal for an Agent Architecture for Proactive Persistent Non Player Characters*. Trinity College Dublin, Department of Computer Science. http://www.tara.tcd.ie/xmlui/handle/2262/13118.

MacNamee, Brian, and Padraig Cunningham. 2003. 'Creating Socially Interactive No-Player Characters: The M-Siv System'. *Int. J. Intell. Games & Simulation* 2 (1): 28–35.

Meehan, James Richard. 1976. *The Metanovel: Writing Stories by Computer*. DTIC Document. http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA031625.

Nelson, Janice D., Donna M. Gelfand, and Donald P. Hartmann. 1969. 'Children's Aggression Following Competition and Exposure to an Aggressive Model'. *Child Development*, 1085–97.

'North America Digital Games Market Report 2014'. 2014. *SuperData Research*. Accessed November 17. http://www.superdataresearch.com/market-data/north-america-digital-games-market-report/.

Nowak, Kristine L., and Christian Rauh. 2005. 'The Influence of the Avatar on Online Perceptions of Anthropomorphism, Androgyny, Credibility, Homophily, and Attraction'. *Journal of Computer-Mediated Communication* 11 (1): 153–78.

Ortony, A., G. L. Clore, and A Collins. 1988. 'The Cognitive Structure of Emotions'. *Cambrige University Press*.

Picard, Rosalind Wright. 1995. 'Affective Computing'. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.153.8488.

Reilly, W. Scott. 1996. *Believable Social and Emotional Agents*. DTIC Document. http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA310766.

Schroeder, Ralph. 2002. 'The Social Life of Avatars: Presence and Interaction in Shared Virtual Environments'. *Educational Technology & Society* 5: 4.

Schwab, Brian. 2004. *AI Game Engine Programming*. Cengage Learning. http://books.google.at/books?hl=en&lr=&id=UbMLAAAAQBAJ&oi=fnd&pg=PR7&dq=AI+game+engine+programming&ots=MK6nEIWJXu&sig=HRwf83W0x82tKUxCmXINEd2Bmyo.

Scott, Bob. 2002. 'The Illusion of Intelligence'. *AI Game Programming Wisdom* 1: 16–20.

Sherif, Muzafer, and Carolyn W. Sherif. 1953. 'Groups in Harmony and Tension; an Integration of Studies of Intergroup Relations.' http://psycnet.apa.org/psycinfo/1954-02446-000.

Stern, Andrew. 2002. 'Virtual Babyz: Believable Agents with Narrative Intelligence'. *Advances in Consciousness Research* 46: 215–28.

Trepte, Sabine, and Leonard Reinecke. 2010. 'Avatar Creation and Video Game Enjoyment: Effects of Life-Satisfaction, Game Competitiveness, and Identification with the Avatar.' *Journal of Media Psychology: Theories, Methods, and Applications* 22 (4): 171.

Trepte, Sabine, Leonard Reinecke, and Katharina-Maria Behr. 2009. 'Creating Virtual Alter Egos or Superheroines? Gamers' Strategies of Avatar Creation in Terms of Gender and Sex'. *International Journal of Gaming and Computer-Mediated Simulations (IJGCMS)* 1 (2): 52–76.

Vallerand, Robert J., and Gaétan F. Losier. 1999. 'An Integrative Analysis of Intrinsic and Extrinsic Motivation in Sport'. *Journal of Applied Sport Psychology* 11 (1): 142–69.

Vallerand, Robert J. Ph.D, Edward L. Ph.D. Deci, and Richard M. Ph.D. Ryan. 1987. 'Intrinsic Motivation in Sport'. *Exercise & Sport Sciences Reviews:*, no. Vol 15 (January): 389–426.

Williams, Russell B., and Caryl A. Clippinger. 2002. 'Aggression, Competition and Computer Games: Computer and Human Opponents'. *Computers in Human Behavior* 18 (5): 495–506.

Wish, Myron, Morton Deutsch, and Susan J. Kaplan. 1976. 'Perceived Dimensions of Interpersonal Relations.' *Journal of Personality and Social Psychology* 33 (4): 409.