FAKULTÄT FÜR INFORMATIK
Faculty of Informatics

"Gangsters and Pranksters: The Art of Thievery": A Multiplayer Stealth Game in the Context of the Generation of Dynamic Replay Videos

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Lukas Bydlinski
Matrikelnummer 0609406

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Dipl.-Ing. Dr. techn. Ivan Viola
Mitwirkung: Ing. Dr. techn. Peter Mindek

Wien, 24.8.2015

(Unterschrift Verfasser) (Signatur Betreuer)

FAKULTÄT FÜR INFORMATIK
Faculty of Informatics

“Gangsters and Pranksters: The Art of Thievery“: A Multiplayer Stealth Game in the Context of the Generation of Dynamic Replay Videos

BACHELOR'S THESIS

submitted in partial fulfilment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Lukas Bydlinski
Registration Number 0609406

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Dipl.-Ing. Dr. techn. Ivan Viola
Assistance: Ing. Dr. techn. Peter Mindek

Vienna, 24.08.2015

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Lukas Bydlinski
Passauergasse 14, 2340 Mödling

Hiermit erkläre ich, dass ich diese Arbeit selbstständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

_____ _____

(Ort, Datum) (Unterschrifft Verfasser)

Kurzfassung

Das Projekt "*Gangsters and Pranksters: The Art of Thievery*", an welchem ich in Kooperation mit dem Kollegen Alexander Bayer gearbeitet habe, ist auf zweierlei Zielsetzungen ausgerichtet: Einerseits auf die Entwicklung eines unterhaltsamen, kompetitiven Videospiels des „Stealth"-Genres für bis zu vier menschliche Spieler gleichzeitig, andererseits wurde es von der Planungsphase an bis zur Fertigstellung auf optimale Kompatibilität mit dem von Peter Mindek und seinen Kollegen entwickelten „Surveillance System" zur automatischen Generierung dynamischer Replay-Videos ausgelegt. Die vorliegende Arbeit konzentriert sich auf die Themen User-Interface, virtuelle Architektur, Leveldesign, Texturierung und Erstellung einer Event-Hierarchie.

Abstract

The projekt "*Gangsters and Pranksters: The Art of Thievery*", which I worked on in cooperation with colleague Alexander Bayer, has two goals: On the one hand, it shall work at its own as an entertaining multiplayer stealth game for up to four human players, on the other hand it was planned from scratch to be fully compatible with the „Surveillance System" developed by Peter Mindek and his colleagues for automatically generating dynamic replay videos. This thesis concentrates on the topics user interface, virtual architecture, level design, texturing and creation of an event hierarchy.

Contents

CHAPTER 1

Introduction

1.1 Motivation

My colleague Alexander Bayer and I already worked as a team on two video game projects during past courses at the TU Vienna – the Java ME-based 2D platformer "*The Tale of the Snail*" (see Figure 1.1) and a 3D shooter game, created with C++ and OpenGL, called "*The Tale of the Cat*" (see Figure 1.2) – and strived for the goal to work together in a gaming-based Bachelor project as well while contributing to science.

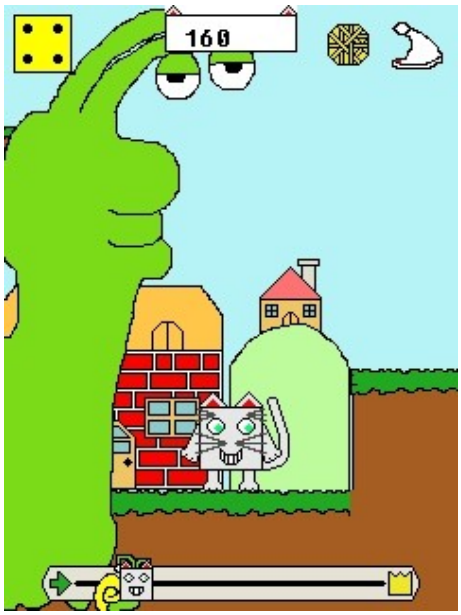


Figure 1.1 and 1.2: Screenshots of "The Tale of the Snail" (left) and "The Tale of the Cat" (right).

After researching suitable Bachelor project possibilities with the help of Professor Eduard Gröller, it was decided to undertake the task of creating a four player game specifically designed for being fully compatible with an existing surveillance system for competitive multiplayer games: This system, created by Mindek et al. and discussed in the paper "*Automatized Summarization of Multiplayer Games*" (2015) by the the developers (cited as [19] in the bibliography), serves the purpose of generating dynamic replay videos post mortem, thus using the accumulated information about events (e.g. „Player X uses item Y“) occuring during a multiplayer match of a compatible game (and about their importance) after the end of a match to synthesize a replay video that utilizes dynamically changing points of view to present the most relevant events at any given time (see chapter 3.0 for more detailed information).

1.2 Problem statement

Mindek et al.'s said surveillance system needed a game that supports it, so the goal we set ourselves was creating a competitive multiplayer game that should be, from scratch, designed to harmonize with the existing surveillance system (which is event-based and operates using the hierarchy of and

the relationships between events happening during a match in the game; see the chapters 1.1 and 3.0) for replay video generation.

1.3 Aim of the work

Our primary goal was the one directly resulting from the problem statement above: creating a multiplayer game that is compatible to the surveillance system. But of course, we also wanted to develop a game that is comfortable and fun to play in its own right, not suffering from fatal bugs or game-breaking glitches and that not simply feels like a clone of an existing game.

After some brainstorming, the idea of a competition of master thieves that try to steal valuable paintings, artworks and the (heavily guarded and often game-deciding) huge diamond while sneaking through the rooms of an art museum at night where security guards patrol came in mind: "*Gangsters and Pranksters: The Art of Thievery*" was born – a concept that uses well-known gameplay elements, but combines them into something new. The core gameplay, for example, resembles stealth games like the *Assassin's Creed* series, but with the big difference that sneaking is not used to knock enemies out or even kill them – on the contrary, neither the avatars of other human players nor the CPU-controlled watchmen can be harmed at all: The player characters are just thieves who steal, but do not hurt anyone – the most "aggressive" action they can do is throwing banana peels on the ground to cause others to slip. Comedic "extra weapons" like this or a bouncy ball that distracts guards are, on the other hand, inspired by classic comedy movies and *Super Mario Kart*.

Finally, the concept of putting up to four characters controlled by human players into a closed-off, arena-like structure, where some sort of conflict between them takes place, is borrowed from classic first-person shooter games like *GoldenEye 007*, but a twist is added not only by the mentioned fact that the contestants cannot defeat each other, but also that they are mainly "fighting the environment" and only secondarily "fighting" each other: The goal of the game is to collect as many works of art as possible to win by being the wealthiest thief after the match – a plan that does not automatically involve the other players. But of course, it is absolutely possible to annoy the other contestants and put them in gameplay-wise bad situations: for example by willingly attracting the attention of the guards when they are near another thief so that the latter is arrested.

So, because of up to four players competing against each other in splitscreen, different treasures with different values to steal, typical stealth moves, an arsenal of extra items, patrolling watchmen and moving security cameras, a lot is going on during a *Gangsters and Pranksters*-match, which means that a lot of events (that are often connected to each other) happen: And exactly this makes the game very suitable for the usage of the surveillance system.

1.4 Methodological approach

The process of prototyping was very important to us: We always followed the guideline to first create a playable prototype that is just based on simple graphics but already includes the most important features, before successively fleshing out the gameplay features, the game's world, controls and graphics. The following sub-chapter shows a more detailed view of these steps.

1.5 Structure of the work

- Step 1: Brainstorming possible gameplay approaches which should be fun to play especially

in multiplayer and are perfectly compatible with the existing surveillance system; also "thinking a bit outside the box" and finally creating the concept of a multiplayer stealth game instead of obvious splitscreen choices like a first-person shooter or a racing game (see chapter 1.3)

- Step 2: Studying tutorials and APIs of Unity (the engine we chose for our game after the brainstorming phase), C# and advanced Blender techniques before actually working on code and graphics.
- Step 3: Creating the first simple test levels with a "walking cube" as a player character and equally simple cubes for watchmen and collectible items – the gameplay-centered approach was always ensured.
- Step 4: Fleshing out the gameplay more and more by implementing advanced gameplay techniques (like the distinction between silent and noisy stealing methods and silent and noisy means of movement).
- Step 5: Creating a well-designed game world (the art museum mentioned before), characters that look like such (rather than "walking cubes") and the graphical user interface.
- Step 6: Inserting the item shop menu players can visit before a match actually starts and enabling the characters to use the items they bought there (using in-game currency that is provided before every match).
- Step 7: Managing the movement and behaviour of the watchmen (e.g. their reaction when they realize that something has been stolen et cetera) in a realistic way.
- Step 8: Writing the priority list of the possible occuring events and implementing the interface to the existing surveillance system.
- Step 9: Texturing all objects and characters visible to actually create the atmsophere of a museum invaded by thieves.
- Step 10: Placing all the NPCs and objects on their final positions, also setting proper monetary values for the artworks, paintings and the huge diamond the thieves are after, and setting waypoints that determine the patrol routes of the watchmen: After this final step, the game shall be playable in a fair way.

CHAPTER 2

State of the art / analysis of existing approaches

2.1 User interface and interactions between player character and environment

Being an arcade-style multiplayer game (see the short description in chapter 1.3), interactions between player characters and the environment play a very important role in *Gangsters and Pranksters*, but are mostly quite uncomplicated interactions between the subject and an object (e.g. a human-controlled thief stealing an object placed in the game world) that can be realized using changeable flags (see chapter 4.1): The implementation of true multi-agent interactions, as Serrano and Saugar discuss (see [24], p.889), was not necessary – if a player character is currently in the stealing process of a particular treasure, it was made sure that no other player is able to steal the same treasure at the same time (realized analogue to the "whoCuts"-principle mentioned in chapter 4.1); in this specific case, there was also no need to "consider interactions as first-class objects and to specify them with hierarchical state machines" ([3], p.51), as Blanch and Beaudouin-Lafon suggest. But more complicated considerations were needed when the nature of another important aspect of the game had to be decided: The user interface.

When working on video games as an interactive medium in general, a fitting realization of the user interface is central: E.g. Kay describes the problem of a badly designed user interface that "[repells] end users instead of drawing them closer to the hearth" ([13], p.123) because of hardly interpretable interface elements with a culinary metaphor: "[The] combination of ingredients did'nt gel. It was like trying to bake a pie from random ingredients in a kitchen: baloney instead of apples, ground-up Cheerios instead of flour, etc." ([13], p.124) – while he writes about a specific desktop personal computer he co-developed (see [13], p.123), this conclusion may serve as a paraphrase for general challenges in designing user interfaces that should neither simplify the processed issue too much nor be overly complex and confusing.

Specifically looking at video games and viewing a given, random sample of them under this thematic complex, it quickly comes to mind that it may be possible to roughly divide digital games' user interfaces in two types: Let's just call them, in the style of Mitchell and Shneiderman or Van den Berg and Coninx respectively, "*static user interface[s]*" ([20], p.34) and "*context-sensitive* user interfaces" ([28], p.87).

Figures 2.1(a), 2.1(b) and 2.1(c): Examples for a static (left) and context-sensitive user interface (right). Small picture: A "Nintendo 64" controller, the input device for the game shown right.

Static and context-sensitive user interfaces

Figure 2.1(a) shows Konami's classic platformer *Castlevania* as an example for a static user interface which doesn't change throughout the course of the game: It means that when the player character is controlled by the player using the NES controller, nearly every usable button or button combination always has the same function. "Left" and "right" on the game pad is for moving, the "A" button is for jumping, "B" for attacking with the main weapon, "down" is for ducking and "up" uses, in combination with "B", the collected sub weapon (in Figure 2.1(a) the axe, as shown as a symbol in the red box in the graphical user interface): With the only exception of the usement of stairways – standing near them, the avatar goes up by pressing "up" and down by pressing "down" – as a single element that is implemented by a more context-sensitive approach (see below), the hero's actions may be limited, but are always and at every given time available on the player's fingertips.

Figure 2.1(b) shows *The Legend of Zelda: Ocarina of Time*, Nintendo's famous action-adventure game that, released in 1998, may have influenced the growing popularity of context-sensitive user interfaces in later and modern video games: Here – assuming that the player is already familiar with the button layout and appearance of the used controller (see Figure 2.1(c) for illustration) – just a glance at the graphical user interface visible on the screenshot already gives an overview of the concept. The specifically coloured and placed HUD elements on the upper right side of the screen represent the blue "B" button, the red "A" button and the yellow "C left", "C right" and "C down" buttons that are similarly placed on the Nintendo 64 controller the player uses to control the player character.

Looking at their functions, no one of them is intrinsically tied to a specific action: The "B" button may normally use the hero's currently equipped main weapon – but that can be, contrary to *Castlevania*, switched in a sub-screen. The three yellow buttons are even more flexible and act as slots to up to three items in the hero's possession, and finally, button "A" ist the most flexible one of them all – depending on the surroundings and the avatar's movement, its function steadily changes: If he has already drawn his sword, the text "Put Away", displayed on the blue "A" icon (see Figure 2.1(b)) means that the "A" button can be used to put away the weapon. If the player character is running, the icon text changes and tells the player that he or she can use the same button to make a sommersault, if the avatar is standing next to a door, it can be opened with "A" and so on. And even the analog stick, normally used for walking and running, can be described as context-sensitive: There is no jump button – when the hero runs towards an abyss, he automatically jumps over it (or at least tries to).

In comparison, *Castlevania* and *Ocarina of Time* are two completely different games that use two totally different concepts – but both games use their own concept well and achieve being a well-playable end product: So, it cannot be said that the context-sensitive user interface concept is automatically a better choice than the static one or the other way around – it is all about suitability. *Castlevania* with its simple gameplay core, but harsh difficulty and emphasis on jumping would suffer greatly from an unnecessary complex user interface in the style of *Ocarina of Time* that even excludes a jump button; on the other hand, *Ocarina of Time* with its many moves and sub items would be a mess to play with just a few buttons with static functionality.

Generally speaking, an arcade-style action game, for example, that shall be easy to pick up but hard to actually master and where all actions performable by the player character are frequently needed, may better rely on a static user interface because it is important there to quickly memorize which button does what to use them as quickly when the avatar gets into dangerous situations: Here, Mitchell and Shneiderman seeing the process of "the user [becoming] familiar with a fixed interaction style" as an "argument for static user interfaces" is well applicable (see [20], p.34). Also, their remark "[A] static interface makes users skills portable across systems, assuming that the user has mastered the fixed interaction style." (ibid.) can act as a description for the purpose of existing conventions in the field of video games: Many games of the platformer genre running on consoles

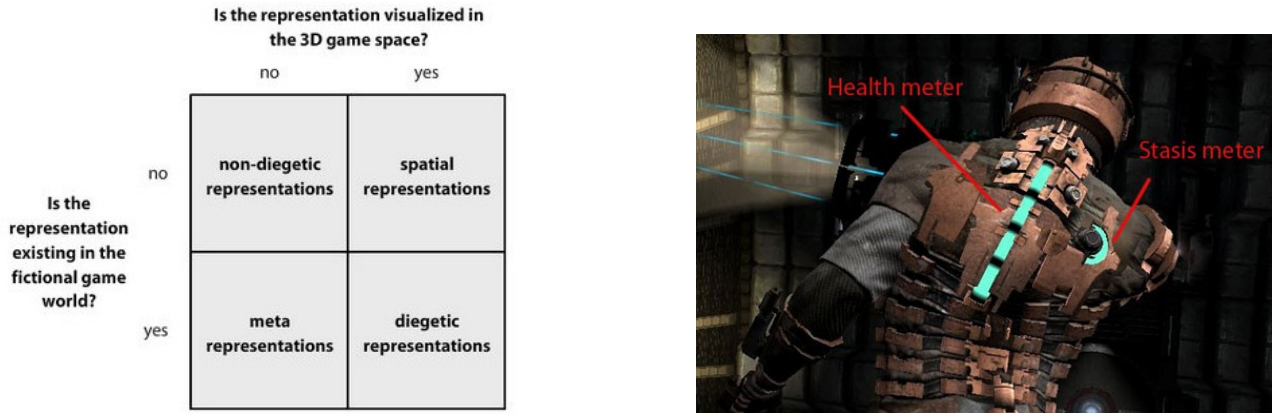
like NES or Nintendo 64 use the respective controller's "A"-button for letting the player character jump and "B" for running – this measure makes is easier for users who already played other games of the same genre to adapt to the control schemes of new ones quickly; or, in Mitchell's and Shneiderman's words, it "makes users skills portable across systems" (ibid).

On the other hand, if a product shall have a more adventure-ish feel with plenty of different situations (from communication to fights) the hero has to deal with, an at least partly context-sensitive user interface may be a better choice: When Van den Bergh and Coninx argue that "integrating context is an important issue because of the varying constraints they pose for user [interaction]" ([28], p.87), it meaningfully applys to the field of video games as well. For example, it can be considered redundant if a designated "climbing" button is used in a game that mainly takes place on flat plains without climbing opportunities – in this case, a context-sensitive action button like the one in *Ocarina of Time* might be an improvement.

Diegetic and non-diegetic user interfaces

Aside from this very central topic regarding the user interface, another related field exists that may seem like a mere cosmetic issue at first but is discussed in scientific literature as well: the actual optical design of the graphical UI and its relationship to the events in the virtual game world. E.g., Andrews statement "UI has everything to do with feeling badass!" ([1], p.4]), despite its vulgarity, illustrates that he strongly argues against seeing the field of UI as cosmetical. Using terminology from Fagerholt's Master of Science Thesis[1] (see [9]), he differs between four different UI representations:

> "***Diegetic:*** Interface that is included in the game world -- i.e., it can be seen and heard by the game characters. Example: the holographic interface in *Dead Space*.
> ***Non-diegetic:*** Interface that is rendered outside the game world, only visible and audible to the players in the real world. Example: most classic heads-up display (HUD) elements.
> ***Spatial:*** UI elements presented in the game's 3D space with or without being an entity of the actual game world (diegetic or non-diegetic). The character outlines in *Left 4 Dead* are an example of non-diegetic spatial UI.
> ***Meta:*** Representations can exist in the game world, but aren't necessarily visualized spatially for the player; these are **meta representations**. The most apparent example is effects rendered on the screen, such as blood spatter on the camera to indicate damage." ([1], p.1)



Figures 2.2(a)/2.2(b): Representation schema (left) and example for diegetic representation (right).

1 According to [1], p.1: "Terminology from Fagerholt, Lorentzon (2009) »Beyond the HUD - User Interfaces for Increased Player Immersion in FPS Games«. Master of Science Thesis, Chalmers University of Technology"

Figure 2.2(a) explains the terminology discussed above in a concise schema, and Figure 2.2(b) shows the armor of the player character of *Dead Space* (which Andrews already mentioned above) as an example for a diegetic interface: On the back of that armor, inter alia the hero's health meter is to be seen. While in Figure 2.1(a) and 2.1(b), the health meters (the red bars in *Castlevania*'s case, the red hearts in *Ocarina of Time*) are clearly artificial and are only to be seen to the players outside the game's virtual world, in Figure 2.2(b), the gameplay-wise important health meter is also integrated in an object in the game world (the avatar's armor), even in a plausible way (as a display of the armor's life support system).

Throughout his essay, Andrews analyzes a couple of different games and their UIs and remarks that while diegetic representations can have a positive effect on a game's immersion and reduce cluttered HUDs as common in non-diegetic representations, but also notes that it is especially distracting if an overall diegetic UI contains single non-diegetic elements (that may be important gameplay-wise) and that players sometimes actually prefer non-diegetic UIs with their wealth of information, despite being possibly cluttered and clearly not originating from the game world (see [1], p.2-3).

So in the end, Andrews reasons that "inclusion of non-diegetic interface elements seems almost required to reach the level of playability required in today's competitive market" ([1], p.4] and concludes "that there is a fundamental rule for all games that must always be met regardless of the looks and functions of any given interface: The rule of functionality preservation and translation must be met." ([1], ibid.): Simply put, the user inferface may be used to increase immersion and authenticity of the game world, but not if it means that this harms the gameplay experience.

Llanos and Jørgensen come to a similar (regarding diegetic user interfaces slightly more negative) conclusion after their research and analysis of this topic and suggest that playabilty, which often means permanent access to all the gameplay-wise important information, comes first and creating a user interface that blends right in with the virtual game world, but actually handicaps the player is a bad decision (see [15], p.10-11). Their conclusion matches matches very well with Andrews' (above cited) statement that (while he also emphasizes the positive aspects of diegetic UIs) non-diegetic UI elements are more or less necessary in modern gaming: "A minimal UI that seems natural to the game environment is not the best way to present information that is critical, or that needs to be gauged on a continuous basis." ([15], p.11)

2.2 Virtual architecture: Modelling objects and environment

Generally, when modelling 3D objects, it is always important to keep in mind which shading is supposed to be used on the final model: The standard "flat shading", which causes all pixels of the same polygon of a 3D model to take on the same colour, establishes a very edged look – so, this principle was only used in *Gangsters and Pranksters* when such an appearance was actually desired, e.g. when surfaces were supposed to have an explicitly artificial look. In most cases, it was resorted to "smooth shading" techniques like Gouraud shading where normals are computed for each vertex on the grid of polygons approximating the model's surface to shade the vertices, and then the shading information of each polygon's corner points is used to determine the colour of every separate point inside the corresponding polygon using linear interpolations (see [11], p.626): This way, less complicated (and therefore less resource-intensive) 3D objects can be used without looking extremely edged.

There are also propositions that search for other approaches of generating 3D objects apart from the time-consuming manual modelling process. For example, Sato et al. suggest an automatic method for that endeavour that doesn't involve manual modelling at all: The user just has to observe a real life object that shall act as a template for the virtual model and obtain two aspects of information – the object's shape and reflectance properties. So, multiple range images of the object

have to be captured and then merged together to reconstruct a surface shape. This obtained shape is then combined with a sequence of colour images of the object to estimate the parameters of a reflection model: That is done by separating the diffuse and specular reflection components from the colour image sequence and then estimating the reflectance parameters of each reflection component separately – this way, it is possible to reconstruct the reflectance properties of real life objects whose surfaces both feature specularity and diffusely reflected lights realistically! Having obtained both shape and reflectance properties, object images with realistic shading effects under arbitrary illumination conditions can be synthesized: A very sophisticated appraoch, but not suitable for a project like *Gangsters and Pranksters* which doesn't strive for a realistic look. (see [23], p.379)

Before now discussing the more specific term "virtual architecture" after the more general coverage of different 3D modelling approaches, it is important to determine what it actually represents. For example, Grosz sketched the ambiguity of the relationship between architecture and virtuality in a dualistic way: On the one hand, there is a usage of virtuality connected to the incorporation of technologies like security systems in buildings, the space of virtuality being understood as a containable, separate field; on the other hand, there is another concept as an "entirely new way of seeing, inhabiting, and designing [space]" that "involves reconceptualizing the real and the relations of embeddedness, the nesting or interimplication (perhaps another name for difference) of the virtual and real within each [other]", Grosz arguments (see [12], p.87-88).

Maher et al. (2001) define the term "virtual architecture" more clearly by distinguishing it from two other concepts from the field of archtitecture: By "[digital] architecture", they mean "the use [of] digital representations in the development of architectural designs", "[physical] architecture" stands for "the result of architectural design as a physical building" and eventually, virtual architecture is defined as "the result of architectural design that serves its purpose as a digital representation" (see [16], p.1).

A more detailed outline of the characteristics of virtual architecture has been portrayed by Maher et al. in another essay published one year earlier, calling it "an electronic representation of architectural [design]" and attributing two purposes to it, the first being the simulation of physical architecture for means like visualisation of architectural designs and the second being a functional virtual place: "The second purpose of virtual architecture involves the design and creation of virtual places in terms of its functional organization and electronic representation. Architects design buildings to provide places for people to live, work, play, and learn." ([17], p.1)

Obviously, we are currently interested in "the second purpose of virtual architecture" as mentioned above: *Gangsters and Pranksters*' game world, a large, but single-story museum, shall not serve as a digital blueprint for an actual building from the real world, but shall just act as an arena-like structure where the player characters try to sneak past guards, steal valuable works of art and try to give the other human-controlled avatars a hard time. But does that mean that considerations originating from the architectural field may be simply ignored? Maher et al. (2000) suggest that given the virtuality, the significance of such considerations about the geometric description of the space may be less important in this case, especially compared to the central topic of functionality, but do not lose their significance completely: "A consistent description of the frame of reference and topology of the space can help in the orientation and navigation within the place, whether we refer to a room, group of rooms or other part of the place. It also contributes to setting the ambience and indicates what the place and different parts of that place feel like." (see [17], p.1-2)

So, similarly to the UI and HUD issues discussed in chapter 2.1 with its tensions between boosted immersion on the one side and comfortable gameplay on the other, "virtual architects" have to master a balancing act as well: On the one hand, the users (in our case the players) shall be able to use the whole set of implemented functions to be performed in the virtual structure (in our case the museum serving as an arena) in a comfortable way. But simultaneously, a convincing optical

and architectural design that lets connotations with real buildings arise in the users' minds is not just to be considered as "eye candy" but adds both authenticity and, because of that, even more convenience regarding the functionality, like more clearness for the user whose avatar (in our case the thieves) moves through the virtual structure: In *Gangsters and Pranksters*' case, for example, a subtle sort of route guidance system is integrated in the appearance of the museum (see chapter 3.2) and the integration of a inaccessible garden area around the museum's walls may not affect the gameplay directly, but if a player character comes across a window through which said garden can be seen, the user knows that their avatar's current location is at the edge of the map.

Which all these factors in mind, a practical first step in the process of creating a virtual building is starting with 2D floor plans as in real life architecture: For example, So et al. discuss the three necessary steps for creating a 3D structure based on a 2D floor plan – wall extrusion ("the process of converting the 2D drawing of the wall structure into a 3D wall [model]"), object mapping ("the process of placing 3D objects and features as specified by the floor [plan]") and the ceiling and floor reconstructions (that "require a method of outlining the perimeter of the 2D floor plan in order to account for convex regions as well as holes in the original floor [plan]") – and suggest a reconstruction process that performs said steps in a semi-automated way (see [25], p.17-19) that influenced the modelling process of *Gangsters and Pranksters*' museum (see chapter 4.2).

2.3 Level design

Before discussing its specific sub-field of level design, it is important to outline the overall topic of game design, like Salen and Zimmerman do when they discuss their term of "meaningful play" (that denotes the important game design goal of "creating experiences that have meaning and are meaningful") and argue that "[playing] a game means making choices and taking actions" while "[all] of this activity [occuring] within a game-system designed to support meaningful kinds of [choice-making]" and "[every] action taken [resulting] in a change affecting the overall system of the [game]", so that "the board, the pieces, and even the rules of Chess can't alone constitute meaningful [play]" because " [meaningful] play emerges from the interaction between players and the system of the game, as well as from the context in which the game is [played]" (see [22], p.33): So while the overall experience resulting from the combination of all game elements present may be the most important factor, the level design is one very important piece of this puzzle as well, metaphorically comparable to the chess board in a game of chess.

Furthermore, factors like the level design and the gameplay core are not to be imagined as isolated entities, but heavily correspond with each other. Feil and Scattergood describe the "gameplay" term as following:

> "Gameplay is a catchall word for whatever the player does with your game that's fun. In *Unreal Tournament*, the gameplay is running around and wasting as many competitors as possible. The gameplay in *Warcraft 3* is controlling troops and defeating enemies. There are as many flavors of gameplay as there are games. When you're creating your game, you'll have to identify what your gameplay is and make it as fun as possible." ([10], p.9)

But the authors also emphasize the fact that game designers should never forget that "fun" is a very subjective term and that it is not advisable to compulsively try reinventing the wheel if there are already practicable and established game and genre conventions generally accepted by the players – unless a very good idea for improvement is present:

> "Remember, you're trying to make a game that many people will enjoy, not just one or two. […] You should also know your genre well enough to know what sorts of things it could do better. Although some of the mechanics may be set in stone, others might be more pliable. If you can find and improve the things that need improving, or change the things that won't

alienate the player, you're on the way to making a great game." ([10], p.3-5)

It may be easy to understand the maxim "As a future level designer, you'll want to make your levels fun." ([10], p.2), but much more complicated to actually achieve this goal – for example, regarding another important issue discussed by Feil and Scattergood:

> "For level designers, a lot of the game systems will be in place by the time you get to your toolset. Infliction of damage should be balanced with the characters' resistance to damage. The physics of the game, and how they affect the player, should be there as well. However, when you're starting from scratch, you have to think about these things carefully." ([10], p.9)

The example of the balance between damage inflicting enemies or obstacles and the hero's resistance to damage is just one factor of the whole issue of the multiple relations between the player and the level he moves in: In our case of *Gangsters and Pranksters*, the patrolling routes of the watchmen should make it challenging to sneak past them, but not impossible – and this, on the other hand, is connected to the speed the avatar moves. Also, it has to be made sure that there are enough hiding spots: This corresponds to the placement of the showcase pillars behind which the player character can duck and hide to avoid being spottet by the guards – but this only works if they are placed with respect to the environment and the characters moving in it. Because of this, big numbers of manual adaptions throughout the game development and testing process were still necessary, even though the museum was already modelled based on a manually sketched floor plan to minimize level design problems arising because of randomness: The endeavour of avoiding such problems was the primary cause for deciding against creating the game world automatically with e.g. Sorenson's and Pasquier's generative system for the automatic creation of video game levels which uses the FI-2Pop genetic algorithm and a genetic encoding technique (that was specifically developed to match level designer's needs) for making it possible to express high-level design goals in a top-down manner (see [26], p.131).

Strongly linked with the creation of the world itself where the game takes place is another important aspect of level design: the movement of characters placed inside it. Since they are not supposed to be, for example, able to walk through solid walls, the properties of the virtual environment have to be defined exactly: When e.g. Champagnard discusses this field, he divides said environemt into the two main components "structure" and "detail"; the first being handled by the physics simulation while the latter being for graphics rendering (see [4], p.60). Or as explained in more detail:

> "[The] **structure** is part of the environment that can physically affect movement. Naturally this includes the floor, walls, doors; chairs, tables, and other furniture; trees, roads, and bridges; and so on. This list is not exhaustive, and should include all the elements in the world that are mostly [static – those] the players cannot trivially push aside. […] [As] for the **detail**, it consists of the »cosmetic« part of the environment: the things game characters cannot collide with (or if so, [insignificantly) – books], kitchen utensils, or bits of food; grass, shrubs, and small ledges; among many others." ([4], p.60)

While this movement-wise limitations apply both to human-controlled and CPU-controlled characters in the game, there are important differences in their concepts of movement as well, as Champagnard elaborates:

> "[**Human**] **players** have to assimilate the environment based on imperfect visual information. The detail of the environment plays an important part visually. [...][**AI**]

> **characters** usually get a simplified version of this world before the game starts (offline), in both an imprecise and incomplete fashion (for instance, waypoints as guides). Alternatively, the environment can be perceived and interpreted online (as humans would). […] After space has been figured out, it's possible to determine which parts are free space and which can be considered solid. This implicitly defines all movement: what is possible and what isn't. Only then can intelligent movement be considered." ([4], p.62)

The waypoint principle mentioned briefly in the citation above is also discussed by Lidén (Valve Software) who proposes a sophisticated principle to increase the tactical ability of NPCs in first person shooter and action adventure games by still using waypoints, but pre-processing their relationships to each other in order to dynamically generate combat tactics for these non-player characters: "By pre-calculating and storing tactical information about the relationship between waypoints in a bit string class, NPCs can quickly find valuable tactical positions and exploit their environment." (see [14], p.211) – still, it is important to adapt the AI and movement complexity to the respective game and the events that occur in it. For example, in *Gangsters and Pranksters*' case, the watchmen act more tactical and aggressive when the spot or hear an intruder or the alarm goes off, but their regular patrolling routes are intentionally uncomplicated: Being an arcade-style multiplayer game, it is important for new players that it is possible to watch the watchmen's movement to quickly learn about hiding possibilites and how to stay out of their sight (see also chapter 4.3).

Finally, let us come back to Maher et al. (2000), because one of the topics they discuss does not only apply to the field of virtual architecture but to level design as well:

> "The geometry of a virtual place is subtly different to the geometry of its equivalent physical place. The walls of a room are important in defining the boundaries of the room, however, they need not be solid in a virtual room. In a physical room, solid walls provide security, visual privacy, and a barrier for sound transmission. In a virtual room, the concept of boundary is relevant for the wall since its visual presence indicates when you are in the room or outside the room. However, the other functions of a wall can be a programmed behaviour of the wall and not dependent on its geometric definition. For example, the security of a room can be programmed to only allow those people with a key or an invitation to enter and regardless of the geometry of the wall, a person without permission cannot go through the wall." ([17], p.9)

The concept of walls in actually modelled virtual rooms was very important to us while designing *Gangsters and Pranksters*' game world: It is all about the field of view of patrolling watchmen and moving security cameras and naturally, analogue to Champagnard's citations from before, it is important that neither of them can see through walls.
But the "room" as a security metaphor that Maher et al. (2000) mentioned above, too, plays an important part in our project as well: To improve the fairness of the gameplay, every player character has a "safe room", hidden behind a curtain with a painted brick wall (dyed in the colour of the respective players colour) – the room where their vault resides – which the CPU-controlled watchmen cannot enter. We implemented it like this for fairness' sake; scenario-wise, it can be comically explained that the (apparently quite incompetent) guards mistake the painted curtains for actual solid walls – but in the end, it only matters from an analytical perspective that the artificial watchmen, in contrary to the human-controlled thieves, do not possess the imagined "key" for the safe room!

2.4 Texturing

As Ebert et al. describe the core principle of texturing: "Texturing is a method of varying the surface properties from point to point in order to give the appearance of surface detail that is not actually present in the geometry of the surface." ([7], p.7) – the most basic procedure of texture mapping consisting in establishing a correspondence between the 3D surface of a parametric patch and a 2D image by denoting every point on the patch's surface as „patch space“ in the form (u, v) while $0 \leq u \leq 1$ and $0 \leq v \leq 1$. Assuming said 2D image is e.g. a JPEG image with 1024 pixels as width and 512 pixels as height, the point (u, v) in patch space is then shaded using the color found at the 2D point (u * 1024, v * 512) in the JPEG image: Uncomplicated, but prone to aliasing artifacts. (see [7], p.8)

Since there are several methods for generating textures, there is e.g. a big conceptual difference between extracting textures from pictures or photographs and generating them procedurally, as the authors discuss:

> "Procedural techniques are code segments or algorithms that specify some characteristic of a computer-generated model or effect. For example, a procedural texture for a marble surface does not use a scanned-in image to define the color values. Instead, it uses algorithms and mathematical functions to determine the color." ([7], p.1)

One approach in the field of procedural textures uses, for example, the so-called „reaction-diffusion“ principle Turk worked to synthesize textures in a manner that directly matches the geometry of a given surface; the principle being based on a biochemical process (see [27], p.289):

> "Reaction-diffusion is a process in which two or more chemicals diffuse at unequal rates over a surface and react with one another to form stable patterns such as spots and stripes. [...] We extend the range of textures that have previously been generated by using a cascade of multiple reaction-diffusion systems in which one system lays down an initial pattern and then one or more later systems refine the pattern. Examples of patterns generated by such a cascade process include the clusters of spots on leopards [...] In addition, […] a method by which reaction-diffusion textures are created to match the geometry of an arbitrary polyhedral surface [is introduced]. This is accomplished by creating a mesh over a given surface and then simulating the reaction-diffusion process directly on this mesh. This avoids the often difficult task of assigning texture coordinates to a complex surface." ([27], p.289)

Both directions of texture generation have their own advantages: While procedural technicals have the benefits of convenient generation and usage, textures extracted from photographs may provide, if used correctly, a nearly photorealistic look in the truest sense of the word.

But on the other hand, it seems impossible to find a "raw" texture in our real world because of casted shadows, non-symmetrical edges and possible distortion because of the camera's viewpoint while taking the picture act as "natural noise":

> "Recent work has focused on the problem of generating a smooth texture coordinate mapping over a complex surface, such that the mapping introduces a minimal amount of distortion in the size and shape of 2D textures applied with it. Ideally, a mapping preserves distances, areas, and angles, so that the texture is undistorted. However, in practice none of these properties is easily preserved when the textured surface is complex in shape and topology." ([7], p.10)

Eisenacher et al. tackle said distortion problem with another approach: Instead of trying to take

photos for texture synthesis under "ideal" circumstances (that may even be considered as a mere theoretic construct), they worked on the endeavour of synthesizing arbitrarily large high-quality textures from any surface from within an arbitrary photograph, removing arbitrary distortion during the process that is based on a generic per-pixel synthesis algorithm and an interface that enables the user to locally describe the geometry supporting the textures by combining rational Bézier patches (see [8], p.419).
A much less elaborate, but simpler and quicker principle for (seemingly) removing distortion is the usage of GIMP's so-called "Perspective tool" – if it is accepted that it's name is in fact a fraud: As the program's documentation describes, said tool doesn't perform a true perspective transformation at all because it doesn't follow perspective rules! Only distortion is executed by arbitrarily moving the four corners of the picture which causes all the image's pixels to be interpolated according to the new values of the transformation matrix which were determined by the new placement of the corners.[2]
But are photorealistic optics always a goal for game designers to strive for? Masuch and Röber suggest no:

> "Non-photorealistic rendering techniques (NPR) are beneficial in many ways: They can support in storytelling, be expressive, giving the game a certain artistic look and feel. The most important factors of great games are gameplay and the factor of immersion. The quality of immersion directly depends on consistency in graphical presentations and behavior of game world objects. According to Disney, realism in depiction is not the real question; it is believability[3][.]"([18], p.6)

So, according to them, it is all about consistency: As artificial as the world of e.g. a Disney cartoon may seem and be – what the recipients, of course, are aware of –, immersion can still take place. But if a sudden change of optical style happens – for example, if actual human actors are seen on the screen when the audience expects cartoon characters –, irritation and a loss of immersion can be the result.[4]

2.5 Creation of the event hierarchy for usage with the surveillance system

> "[Automated] video surveillance addresses real-time observation of people and vehicles within a busy environment, leading to a description of their actions and interactions. The technical issues include moving object detection and tracking, object classification, human motion analysis, and activity understanding, touching on many of the core topics of computer vision, pattern analysis, and aritificial [sic!] intelligence. Video surveillance has spawned large research projects in the United States, Europe, and Japan, and has been the topic of several international conferences and workshops in recent years. [There] are immediate needs for automated surveillance systems in commercial, law enforcement, and military applications." ([5], p.745)

As this quote from Collins et al. suggests, the usage of surveillance systems is often associated with other fields rather than with entertainment products like games; and as statements from Dikmen et al. – "Event detection in video is an emerging application area. Literature on this subject is advancing fast and existing test datasets are quickly being rendered too easy." ([6], p.9) – or

2 see GIMP documentation: http://docs.gimp.org/en/gimp-tool-perspective.html (Accessed on August 23rd, 2015)
3 Source of the Disney statement (according to Masuch and Rober): "Frank Thomas and Olli Johnston, The Illusion of Life, Hyperion 1981"
4 Of course, this applies not to movies that play with this otherwise often irritating technique to tell their story (like Disney's *Who Framed Roger Rabbit*, where humans and cartoon characters coexist story-wise).

Benmokhtar and Laptev – "Automatic video surveillance holds a great potential for security applications. The large variability of video data with respect to view points, lighting, clothing of people as well as occlusions currently makes this task to a highly challenging research problem." ([2], p.1) – show, these "real life" areas of application are dealing with challenges quite different from the ones we faced in the process of developing *Gangsters and Pranksters* and making it compatible with the virtual surveillance system. Our task is not to extract specific detail information from complex recordings, but it is all about events we made possible to occur in the virtual world we created.

As shortly hinted in chapter 1.2, Mindek et al.'s surveillance system (that will also be discussed in the following chapter 3.0) is inter alia based on the usage of a hierarchy (like a sorted list) of the possible event types that can occur in a compatible game: The system extracts all the events happening during the game and their respective causality – given a first person shooter as an example, the event "player A shoots" can lead to the event "player B dies"; in this case, player A has killed player B and a causal connection between said two events was established! Mindek et al. differentiate between two different connection types between events, both of them being visualized in a resulting "event graph": "causal links", which connect causally dependent events in the graph, and "player links", connecting events with the same participants taking part in it because of their semantic relationship. (see [19], p.96)

Based on the information found in the event hierarchy list, the most important events are selected and used for the creation of a dynamic, post-mortem generated replay video (see [19], p.94); more information about the concept of the surveillance system is found in the following chapter 3.0. So, when thinking about the creation of an event-hierarchy for use with said surveillance system with all the previously discussed aspects in mind, the following clippings of a game-theoretical model sketched by Osborne and Rubinstein seem applicable:

- "A set A of *actions* from which the decision-maker makes a choice." ([21], p.4)
- "A set C of possible *consequences* of these actions." ([21], ibid.)
- "A *consequence function g: A → C* that associates a consequence with each action." ([21], ibid.)

Just as Mindek et al. discussed, it is all about events happening that often influence each other because e.g. event V conditions event W or event Y is a result of event X! This principle is important to be kept at the back of one's mind while working on the creation of an event hierarchy (see also chapter 3.5 and 4.5).

CHAPTER 3

Methodology and suggested solution

3.0 Mindek et al.'s surveillance system: A short high-level description

In short, Mindek et al.'s surveillance system was created for the purpose of automatically generating dynamic replay videos of competitive matches fought out in compatible competitive multiplayer video games. The principle developed for achieving this goal rests on three pillars:

- **The event graph:** As already discussed in chapter 2.5, events (like "Player X uses item Y" or "Player X attacks Player Y") occur during multiplayer matches and information about these events and their causality/relationships (e.g. "Player X loses a life" as a consequence of "Player X attacks Player Y") is stored in an event graph. Every event is captured by a camera.
- **"flock of cameras":** The term "flock of cameras" describes a set of all cameras relevant to the gameplay, each camera having dynamically changing parameters and a specified, limited lifespan. Within said lifespan, an assigned continuous importance function supplys the camera with information about the importance of the events it has captured in time.
- **"ManyCams":** Because every event is captured by a camera, there is a big number of views captured by the flock of cameras. Now, the "ManyCams" system selects the most interesting ones based on a known importance hierarchy of possible event types. These views that were extracted by ManyCams are then merged and smooth transitions are inserted: This way, a story that visually narrates the gameplay is created!

While the event graph and the flock of cameras are constructed and operating during the gameplay, ManyCams uses a post-mortem principle and generates the replay video after the game has ended. (see [19], p.93-94)[5]

3.1 User interface and interactions between player character and environment

Very early in the planning phase, it quickly became clear that Gangsters and Pranksters should use a mainly context-sensitive user interface: The player characters (thieves) are able to do a variety of moves, and many of them can only be performed in specific contexts.

5 This reference refers to the whole chapter 3.0.

Figure 3.1: A screenshot of the red player's starting point in "Gangsters and Pranksters".

So, there is a single item button, but the currently equipped item can be dynamically changed with two buttons for scrolling through the item list (in the example above, a banana peel to make the guards or other players slip and fall down is currently equipped); and we based the context-sensitive "Action" button on the "A" button as used in Ocarina of Time (while mostly icons are used instead of written info): If a player is standing near a painting or an artwork, they can steal it by pressing that button. If they are standing next to a spot where a (now stolen) painting or an artwork was (recognizable by a single nail in the wall or an empty showcase), they can craft a fake painting or artwork and put it there to deceive the watchmen (who cannot tell the difference between original and fake). And there is the example shown above: When a player character stands near their personal vault (the one in their own colour), they can store their stolen goods in it so that they are safe and the watchmen cannot take them away anymore, even if the avatar is apprehended by them!

Figure 3.2: The red player tries to cut open a glass showcase and the interface changes.

But the context-sensitivity of our interface doesn't end here: To explain this it has to be noted that, to add a bit of realistic elements to the otherwise comical setting, we chose to make the stealing process not too simple. In other words, to actually steal a painting the player does not only have to shortly push the action button, but they also have to hold it down until the stealing animation has

ended and the painting icon appears in the inventory on the lower edge of the screen because the thief tries to cautiously take the painting off the wall without damaging it or making noise. When the player lets go of the button before that – for example, if a watchman is approaching and they have to get away –, the painting remains hanging on the wall.

The principle is analogue regarding the stealing of artworks – but here we added an additional twist: The artworks are more valuable then paintings, and so, they are also harder to get! Specifically, they are displayed in closed, transparent showcases made from security glass which the thief has to cut open – which the player also does by holding down the action button for a given time (during which the thief's glass cutter is visible on screen). After that, a circular hole is seen in the showcase and the thief may steal the artwork inside – once again by pressing and holding down the same button once again.

The now discussed twist we added is inspired by a move called "knockout smash" from *Batman: Arkham Asylum*: In this game (which mixes elements from the action-adventure, stealth and brawler genres), title character Batman (who never kills his enemies) can sneak past gangsters to make an ambush and render them unconscious silently, which takes some time. But if the player suddenly has to hurry – for example, because the hero's attack was detected by another, armed thug –, they can press another button that performs said knockout smash, a very quick attack that knocks the enemy out but makes a lot of noise by doing that. So, Batman can try to get away fast, but maybe other gangsters are now alarmed because of the noise as well.

In *Gangsters and Pranksters*, while there are no physical battles at all, players can perform a similar action during the silent, but slow glass-cutting procedure: As seen in Figure 3.2, a hammer symbol appears in the item window. That means on the one hand that the regular items cannot be used while using the glass cutter (because our thief has his hands full at the moment), but on the other hand that another context-sensitive move is now ready to use: the hammer[6]!

So, if the item button is pushed while the action button is held, the avatar deals a hammer blow to the glass showcase that damages it significantly more than the glass cutter and may smash it open very quickly – but of course, this move makes a lot of noise! This way, the players can choose between multiple different, strategic options.

After all, the essence of our considerations was primarily that our game should be comfortable to play and use all necessary buttons, but at the same time not more than it needs: We didn't want to use lots of buttons among which some almost never come to use, but implement an intuitive concept which only enables specific actions when they are meaningful to perform. Another example for this would be the "drop diamond" icon that appears in the item window while the avatar is carrying the huge diamond (an action that utilizes all the thief's physical strength): It is just logical that while doing this he cannot use an item; instead, the item button is occupied with another, meaningful action – dropping the diamond to the ground (for example, to get away)!

Because of all these possible actions, many of which can only be performed at certain circumstances, and the splitscreen multiplayer mode, which scales down the display windows, it was clear to us that we would use a non-diegetic interface: A diegetic one would just hide too many important facts for the player and would even be more problematic in splitscreen mode, where graphical interface elements embedded into the game world would appear drastically smaller. But small diegetic elements were added when they were meaningful and as practical as "artificial" HUDs: So, the "turbo shoes" and "ninja sandals" are visible on the avatar's feet after they are used as an item, and as long as they can be seen there, they are in effect – a gameplay-wise important info, this time communicated with diegetic means.

6 We originally planned to integrate more than one playable character, and every one of them should have used a different hammer type (see the Appendix); in the submitted product, the only player character is El Matador (in different colours), who comically uses a dipper to smash the showcases open. But to avoid confusion between the symbol and the object, the more general term "hammer" is still used in this paper.

3.2 Virtual architecture: Modelling objects and environment

Based on the concluding assumptions of Maher et al. (2000), we tried to stick to the following methological guidelines during the game designing process:

- "The virtual room can be designed with an analogy to the physical room, introducing new possibilities for function and geometry of cyberspace." ([17], p.9) – The museum is both optically and regarding the composition and furniture meant to look like one; the halls are meant to display works of art and the corridors link them with each other like in real museums. But at the same time, since we are talking about a game that takes place in cyberspace, the layout was also heavily influnced by the desire for making it well playable by, for example, arranging the furniture in a way that creates good hiding spots for the player characters (see chapter 3.3).
- "Virtual architecture is comprised of objects that have properties in an object-oriented programming sense." ([17], ibid.) – The different properties of different objects and the ways to interact with them (see also chapter 3.1) are central in *Gangsters and Pranksters*: Works of art can be stolen, security mechanisms disabled and so on; only meaningful actions are possible.
- "The definition of the objects in virtual architecture follow [sic!] from the intended function of the place." ([17], ibid.) – Expanding it to "[...] from the intended function of the place to the current time of day in the virtual world": The events of our game take place at a heavily guarded art museum at night! So, both the surveillance cameras and the guards "function" differently than they would in broad daylight (where they also may be present): If they spot an intruder, he is apprehended at once!
- "The geometry of the virtual room provides the visual cues for its use and creates ambience for the people in the room." ([17], ibid.) – We implemented an unobtrusive but clear guidance system in the game's setting: The museum is made of four quarters (each of which contains a starting point for one of the up to four players) with deliberately the same room layout (in favor of gameplay-wise balance), but differently (but not lurid) coloured walls – player 1's avatar starts (and respawns after their apprehension) in the red area, player 2's in the blue segment, player 3's in the green part of the museum and player 4's in the yellow zone. Also, the nature of the floor imparts knowledge to the players: A rug marks a hall where paintings and artworks are displayed but also watchmen are patrolling, a wooden checkerboard pattern stands for corridors that lie outside the normal patrolling route of the watchmen but may be secured by security cameras and contain no works of art. Slabs on the floor signalize cross-over areas between two colour-coded areas and, finally, slabs that get smaller and smaller from one room to the next show that the player is on their way to the center of the museum where the extremely valuable huge diamond is kept. On a side note, a garden and a garden wall were modelled und textured, even though the player is never able to get there: This detail was just added so the museum's outer walls could have transparent windows through which the garden can be seen – both a visaul detail and the important information that the room the avatar is currently inside is located at the outer edge of the museum.

3.3 Level design

After the architectural process, as discussed in the chapters 2.2., 3.2 and 4.2, the structure of the game world itself was already finished – which meant actually "designing the level" more or less consisted of placing the objects, characters and their patrolling routes in a meaningful and

gameplay-wise convincing way in the museum quarter (and then copying the final results into the other three quarters). This was planned with the simplest means possible: with pen and paper!

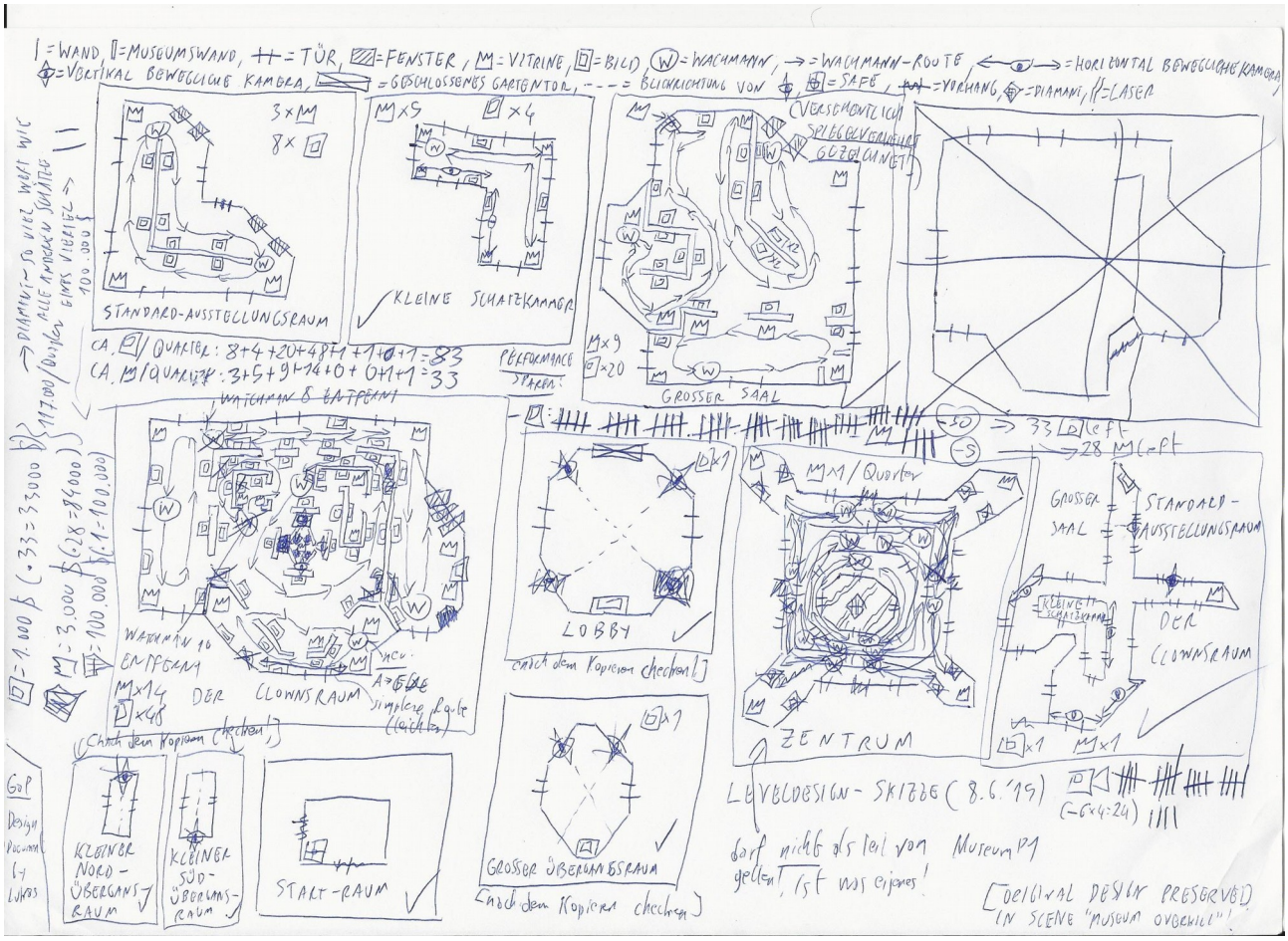


Figure 3.3: Scan of my level design document of a museum quarter.

As seen above, said document was far from true to scale (because of accidently changing from one museum quarter to the next, the scribbles are even partly mirror-inverted) and was revised several times because when tested in the game, it turned out that e.g. there were too many watchmen in a room that made the task of stealing the works of art there impossible or that the placement of a showcase pillar had to be corrected because patrolling watchmen could spot the "hiding" player before. So, the level design was improved again and again until we created the final version.

Sadly, the last "improvement" was necessary because of performance issues: There were much too many objects present in the game world that prevented the game from running smoothly. So, many paintings and some showcases had to be removed – but I made sure that no important hiding spot was taken away and tried to remove these paintings in a well-balanced way so that the museum does not seem "empty" as a whole.

3.4 Texturing

We tried different approaches regarding the visuals of the main character: One of them was using shots of Alex' head from every possible direction to use these photographs as raw material for generating the final texture.

Figure 3.4: All around-shot of Alex' face for a texturing attempt.

But soon we realized that our character model just was much too simple for such a detailed texture: Mapping an image of Alex' actual face on a head modelled with just a few polygons just seemed odd, while the motionless facial features – especially the eyes – further reinforced the "uncanny valley"-feel.

Figure 3.5: A snapshot of the unsuccessful undertaking of using Alex' photos as a texture.

So, because our project's priority was the gameplay (and the usage of the surveillance system) and designing a really realistic character model would take a lot of time we needed otherwise, we finally concluded to texture the player character in a more stylized way using a comic-like approach.

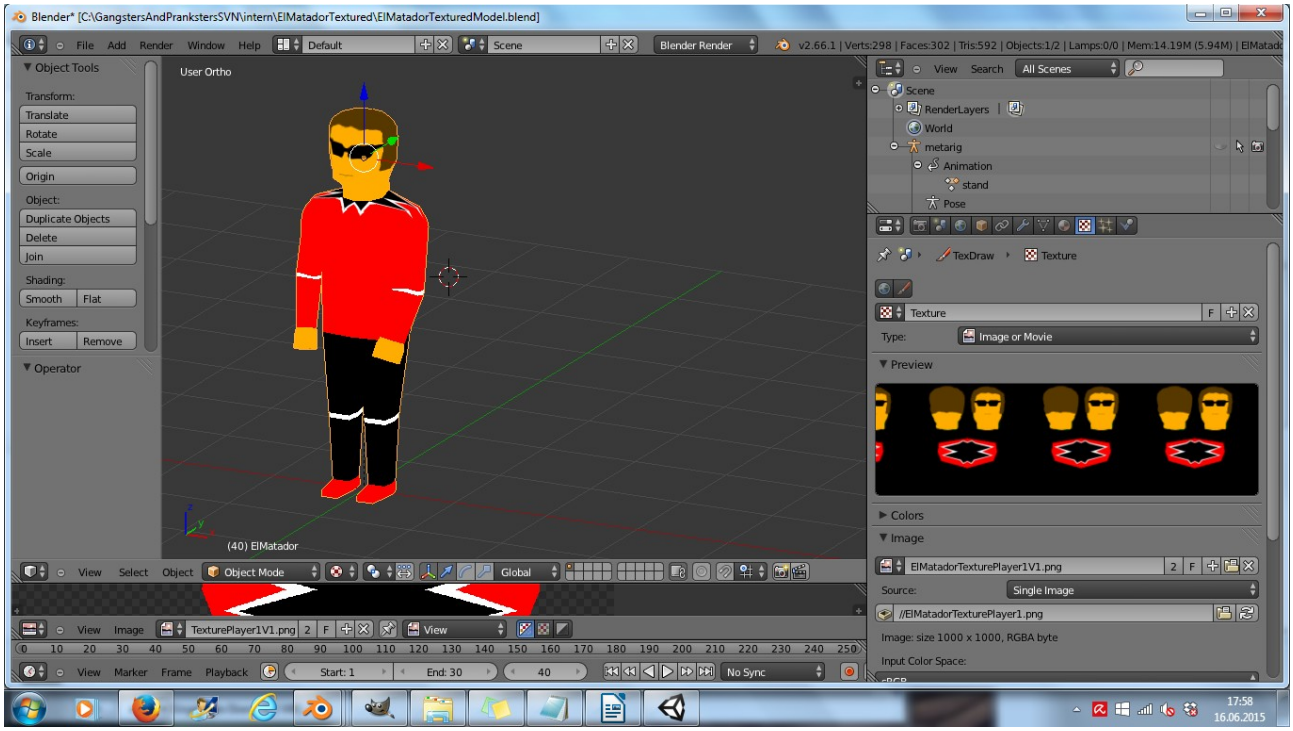


Figure 3.6: The actually used textured model of the player character "El Matador".

While the watchman's texture was made in a similar way (by creating it from scratch using the image manipulation program GIMP), we didn't give up on textures based on real photos completely: On the contrary, using them on the museum's model and certain objects – like in the case of the texture of the extremely amateurish (but for some reason, regarding the guards, convincing) fake of a displayed artwork – seemed quite expressive to us!

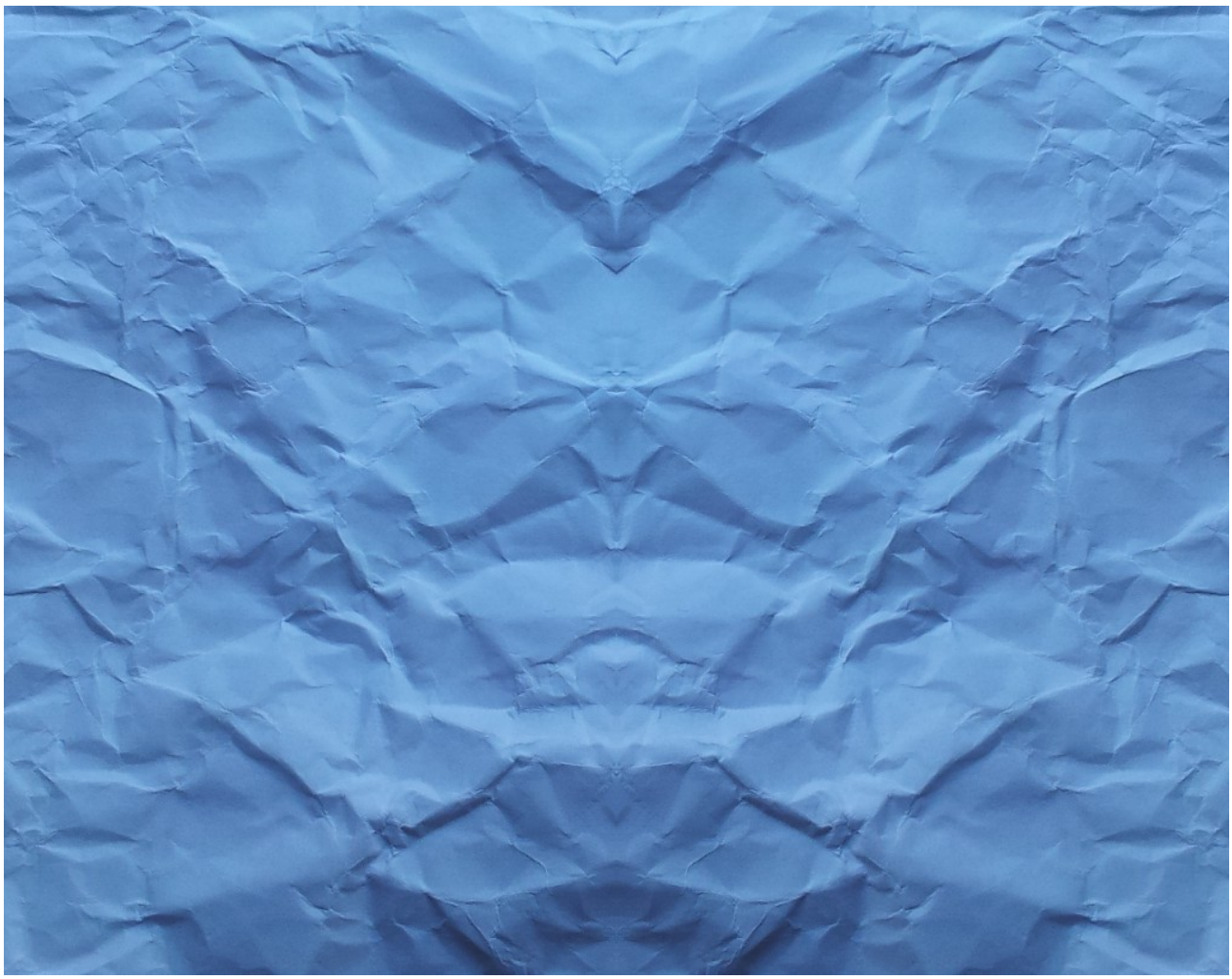


Figure 3.7: The texture of the faked artwork, based on a photograph edited with GIMP.

Of course, we didn't strive for a photorealistic look in the end: When I extracted image parts from photographs, I edited them using GIMP in different ways for removing distortion, making the (once natural) image symmetrical using the mirroring tool (both makes the texturing process in Blender much easier), varying factors like hue and saturation (the red, blue, green and yellow walls are all based on a white wallpaper) or playing with the dimensions (the wall and floor textures in the safe room are actually taken from very small tiles of wood). Because all of these measures, an overly realistic appearance is reduced and both the completely artificial components and the ones based on real photographs are made compatible to each other, avoiding the consistency problems discussed in chapter 2.4.

3.5. Creation of the event hierarchy for usage with the surveillance system

The creation of the event hierarchy was a methodologically straightforward, but nonetheless challenging task: At first, I tried to write an outline of all the events that could happen in our game world. Then, I grouped them in a variety of different classes with different priorites: The "class 1 events" are the most important, the "class 29 events" the least important ones, and the more "extraordinary" and gameplay-wise important events have a higher priority than common events that occur all the time – for example, the event of a character losing the diamond (class 2), which of course means that the same character must have acquired the diamond before, is ranked much higher (which means a smaller class number) than e.g. when a player brings his loot to his safe area,

an event that will occur often (class24).

Of course, because of some gameplay elements being different over the course of the development process compared to our initial vision and others had to be cut, the event hierarchy had to be revised again and again.

CHAPTER 4

Implementation and used technologies

4.1 User interface and interactions between player character and environment

The C# class "PlayerCharacterController.cs", coded in Unity's "MonoDevelop" editor, is central to the user interface: Because the UI changes very often and fast, the code has to make sure that only the actions can be performed at a time that are allowed at that moment (checks are performed every frame): For example, like the cutting and stealing processes that take some time to succeed (see chapter 3.1), we implemented (for a bit more "realistic" feeling and challenge) a certain "itemUsageDelay" (1 second as a float value) which denotes the time that passes between the moment the player presses the item button to use the currently equipped item and the moment the item-effect takes place in the game. This means that from the time the item is pressed our "deltaTime" (a float value that is linked to the actual framerate) is added to the "itemUsageDelayCounter" (float value, initialised with 0) every frame until itemUsageDelayCounter >= itemUsageDelay: Then, the player can actually profit from the item effect (and the counter is reset).

Also, of course, it has to be made sure that during this item preparation phase no other item from the avatar's possession can be used: This is done by "itemPreparationInProgress" which is set to true if the delay time discussed above has not passed yet (then, no other item can be used) and to false if the item usage procedure is already finished (then, another item can be used).

Another example for control mechanisms like this is the hammer that can only be used if the boolean flag "hammerInsteadOfItem" (in PlayerCharacterController.cs) is true, which means that the player is currently using the glass cutter (only then the hammer is usable, see chapter 3.1). Also very important and checked via Showcase.cs: A player can only cut a showcase if no other player is currently cutting the same showcase (in the int variable "whoCuts", the code number of the player who is currently cutting it is stored – 1 for player 1, 2 for player 2 etc. – or 0 if nobody is cutting it right now) and if the showcase is still securily closed (if the int varibale "showcaseState" has the value "Tags.unharmedShowcase") and not cut open ("Tags.cutOpenShowcase") or even smashed ("Tags.smashedShowcase"). And if e.g. player 1 starts cutting a showcase, its „whoCuts“-variable gets the value 1 and in the next iteration after the next frame, the player is actually damaging the showcase they are cutting and so on, until the showcase is finally cut open (which is similarly realized code-wise like the itemUsageDelay from above).

4.2 Virtual architecture: Modelling objects and environment

The museum and all objects inside were all modelled using Blender, relying on experience with the program during the designing process of *The Tale of the Cat* and the comfortable wall and room building principle illustrated in tutorial [T1]: Before even contructing a single plane, I started with the construction of a floor plan only consisting of vertices and edges!

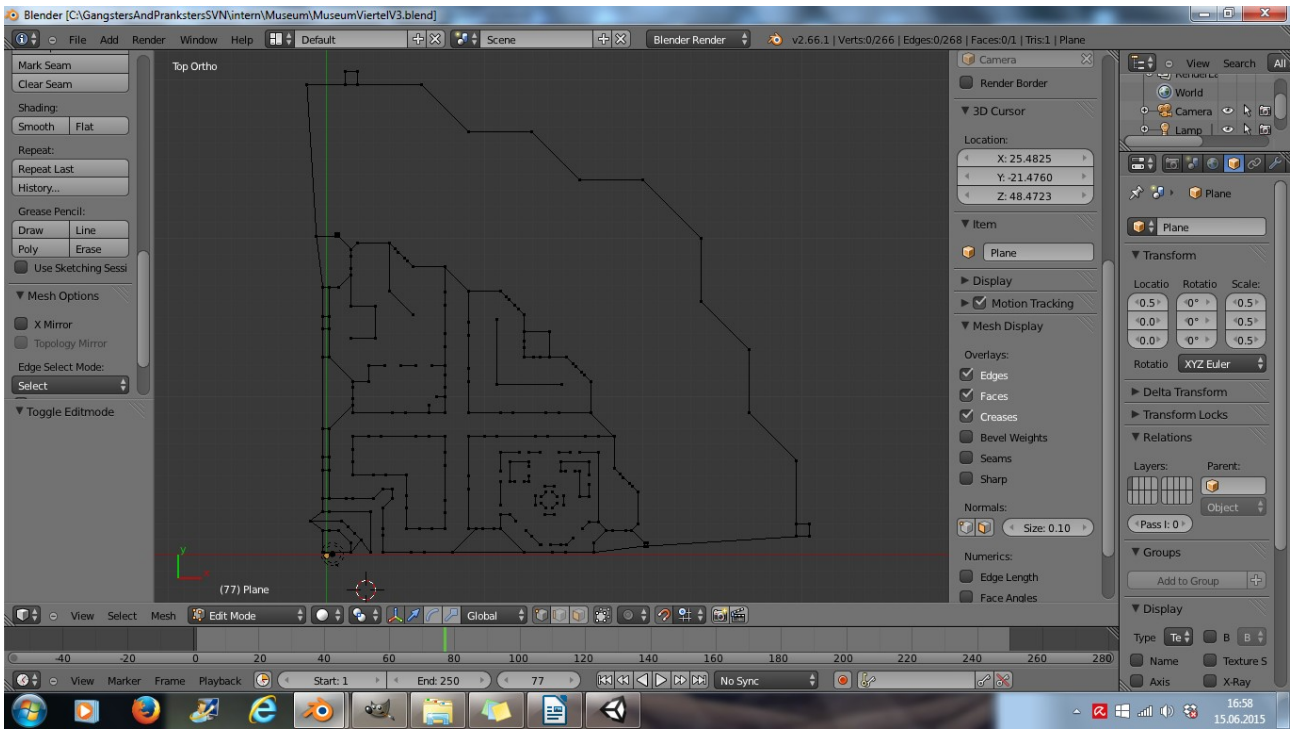


Figure 4.1: Finished floorplan of one of the museum's quarters, modelled using Blender.

After I was satisfied with the overall layout of the map, the next step was to extrude and solidify the existing edges that should become the walls, build the floor, insert the doors and so on until the modelling phase was finished and the structure could be textured. Of course, many experiments were performed until this: How high should the walls be? What about shape and size of the doors? How big should the difference be between size and thickness of the regular walls (that are thought to be permanent parts of the building) and the accordion partitions (which are often used in real museums as temporary spots to display paintings)?

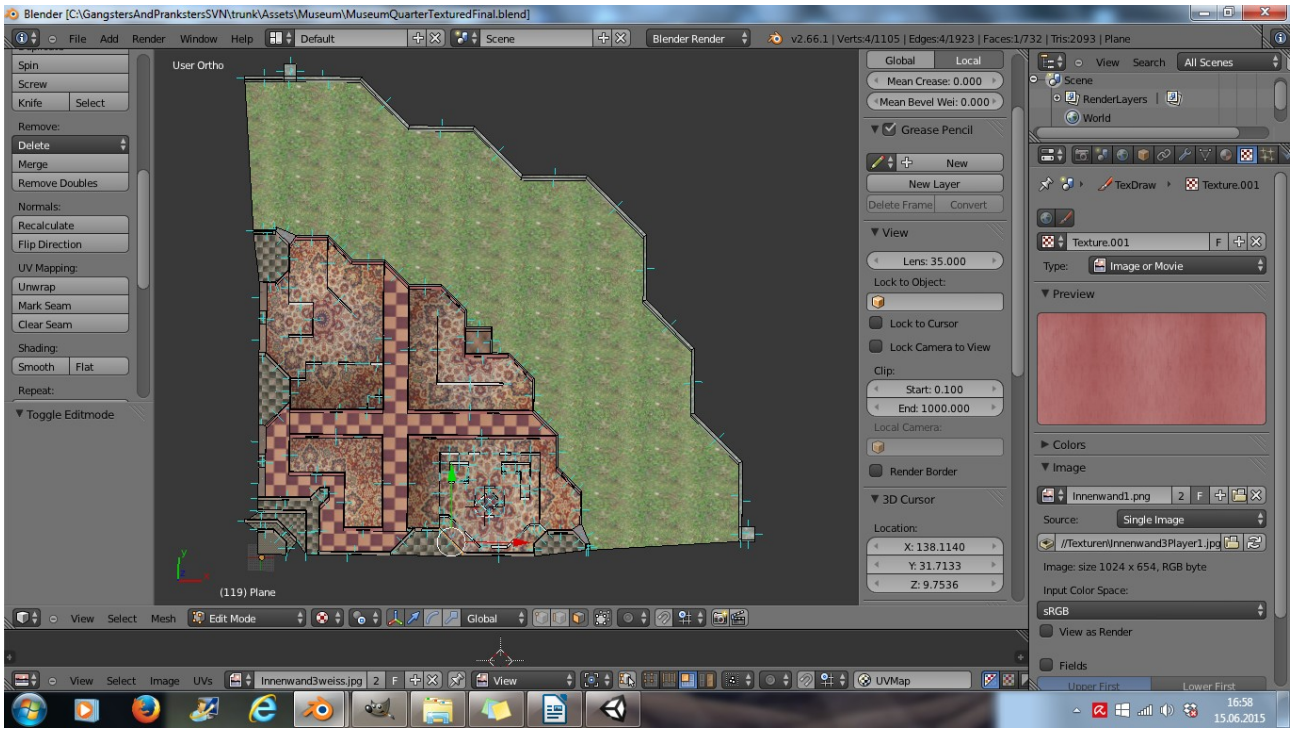


Figure 4.2: Bird's eye view of the finished museum quarters as a 3D model, modelled using Blender.

Finally, after the factors location, rotation and scale were applied to the object in Blender, four instances of this finished museum quarter (either one with an applied unique wall texture in red, blue, green or yellow) had to be imported into the graphical Unity editor to assemble it to a whole museum to essentially finish the architectural process. A (more or less cosmetic) museum roof as a separate object was added later.

4.3 Level design

The tasks discussed in chapter 3.3 were performed comfortably by using the graphical Unity editor: Here, it is very simple to set objects into the game world, duplicate them and change their transformation matrix. Also, as stated before, it was possible to set all the objects of one museum quarter "MuseumP1" (it was named that way because player 1's safe house / starting point is located there) as children of MuseumP1, so this quarter had just to be duplicated three times to create four quarters with the same layout because the children were copied in this process as well. Then, these four quarters had to be arranged in the right way (using transformations) to form a completed, "stitched together" game world and finally, every quarter got its own coloured texture (red for MuseumP1, blue for MuseumP2, green for MuseumP3 and yellow for MuseumP4).

Regarding the creation of patrolling routes for the watchmen, Unity offers a convenient "waypoint system" that allows to create waypoints in the scene than can be assigned to any watchman: For example, if "Watchman1" uses the three waypoints "Watchman1WayPointA", "Watchman1WayPointB" and "Watchman1WayPointC" in the order "A → B → C → B" ("Size=4" in the "PatrolWayPoints"-inspector because of the four entries in the order), it means he walks from A to B, then from B to C, then from C to B and then (because after all the elements denoted in the order ran through, it starts again at the first one) to A again and so on: An "L"-shaped patrolling route. As another example, "A → B → C → D" could stand for a rectagular patrolling route where the watchman never makes a turn of 180 degrees (like he did in the example before).

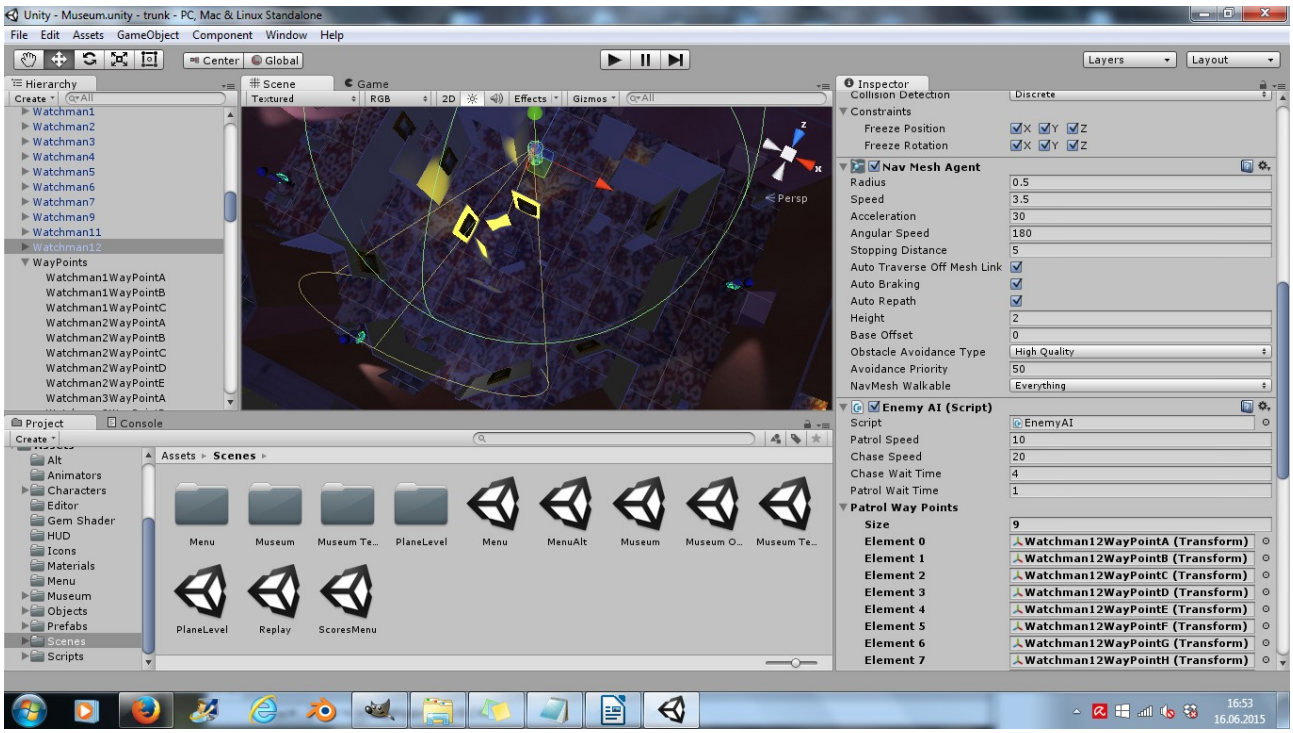


Figure 4.3: Placing objects like watchmen and their waypoints in the graphical Unity editor.

While these watchmen and watchman routes could be easily duplicated for other museum quarters

as explained above, the very center of the museum was considered an independent segment and the watchmen patrolling there were knowingly not classified as children of MuseumP1 or such because it is important that they seamlessly patrol from one quarter of the central area to the other.

4.4 Texturing

As explained in chapter 3.4, the programs GIMP and Blender played an important role in the texturing process: Sometimes parts of photographs, edited and cut using GIMP, where used as textures. Other times, I started with generating a texture layout in Blender by using the "Unwrap" function and, mostly, the "Cube Projection" (in cases of simple structures) or "Project From View" principles (in cases of more complex structures that require manual arrangement), exporting it as an image file and using it is a template while working on the texture in GIMP.

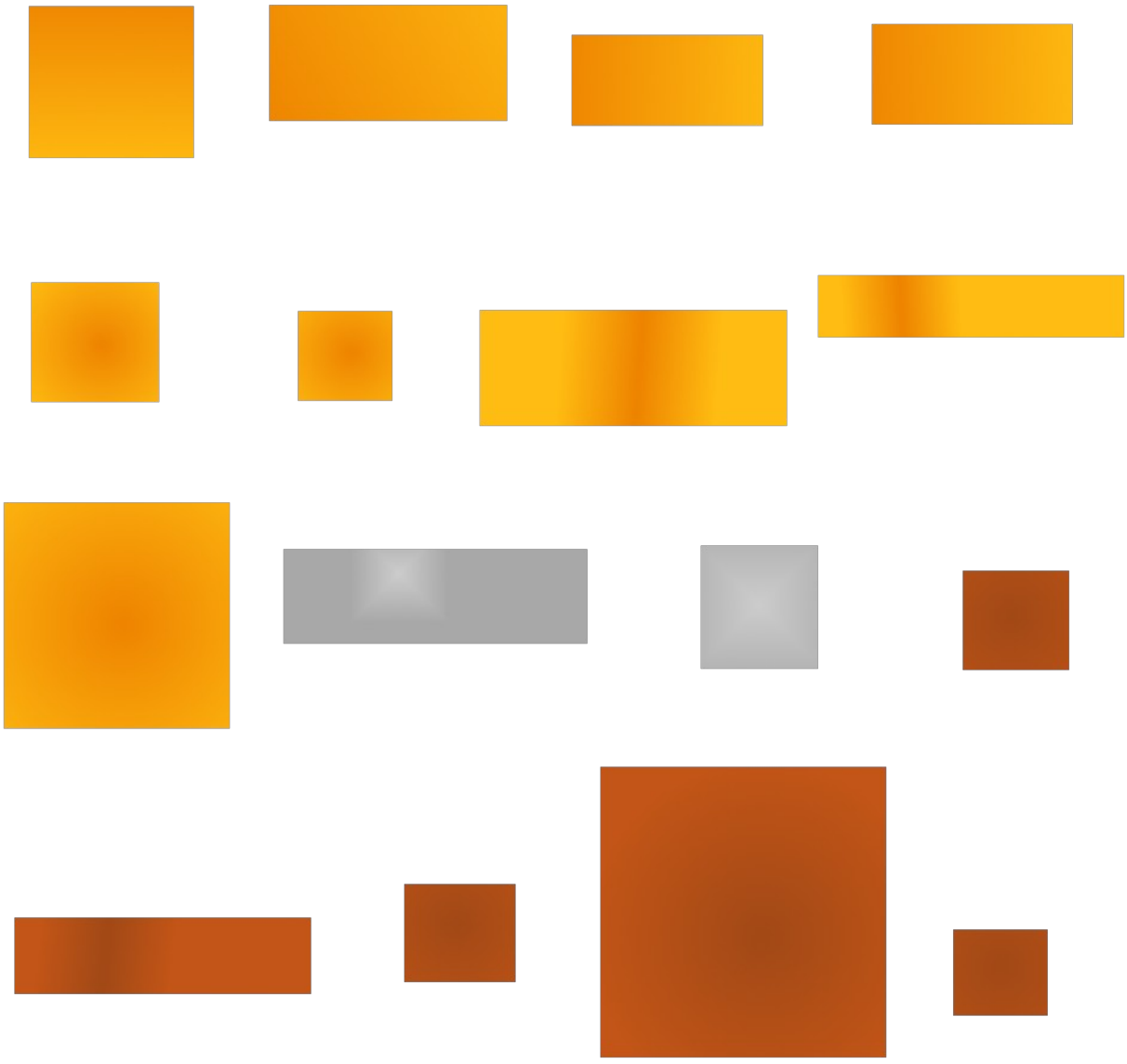


Figure 4.4: The texture map of El Matador's glass cutter.

As seen above, I wanted to avoid simple, one-coloured areas (aside from the very stylized player character and watchman) and used GIMP's colour gradient tool to achieve an optical style that seems a bit more vivid.

4.5 Creation of the event hierarchy for usage with the surveillance system

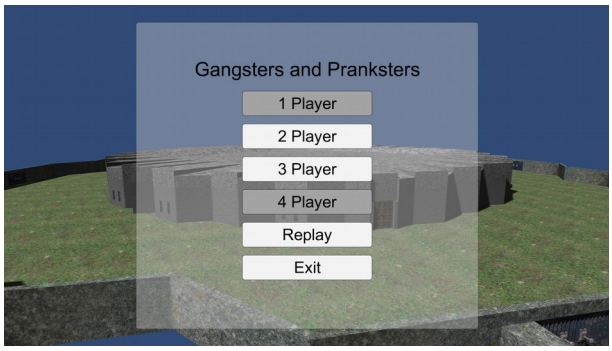
After the final event hierarchy had been created, it just had to be put down in writing as a .txt-document called "EventHierarchy.txt" and now can be found in our submission's "Replay"-folder so the game logic is able to resort to it.

CHAPTER 5

Results

5.1 Demonstration of a Gangsters and Pranksters match

This section gives a quick overview of a process of a Gangsters and Pranksters match.



Figures 5.1(a) and 5.1(b): The main menu (left) and the item shop screen (right).

Gangsters and Pranksters can be played by up to four players (see Figure 5.1(a)): Player 1 may use the PC's keyboard or an Xbox 360 controllers; the players 2-4 have to use Xbox 360 controllers. Before a match actually starts, any player may spend up to 500$ of in-game currency to fill up their inventory with useful items in the item shop menu (see Figure 5.1(b)).

Figure 5.2: A four player match.

In the case of multiplayer matches, the screen is split (see Figure 5.2) and every player character is given a specific colour for being easily able to tell them apart: Player 1 ist red, player 2 blue, player

3 green and player 4 yellow. This colour code also applys to their respective HUD elements (see the treasure inventory on the lower central corner and the item and action displays on the upper right corner of each split screen segment).

Figures 5.3(a) and 5.3(b): A player stealing (left) and faking (right) a painting.

The goal of the game is to acquire the highest score of all the human-controlled competitors: This is executed by stealing works of art. Figure 5.3(a) shows a player during the stealing process of a painting: To successfully steal said painting, the player has to press and hold the action button for a specific time when standing near to it (the possibility of doing so is denoted by a money sack symbol in the action button HUD). When a painting has been stolen, a fake can be fabricated by the same input when standing at the place where it was formerly located (illustrated with a single nail sticking out of the wall where the artwork was hanging before) – in Figure 5.3(b), both the previously stolen painting can be seen in the inventory (it takes two slots) and the fake painting is already visible hanging on the wall. By faking, the patrolling guards can be detained from being alarmed because they mistake the fake for the original. Both actions follow the rule that the action button may not be released prematurely: If the player does this, the stealing process is aborted and the stealing timer is reset, which means when they try another attempt, the neccessary stealing time (= the time the action button has to be held down) is not reduced.

Figures 5.4(a) and 5.4(b): A player cutting (left) and smashing (right) a showcase.

Another option to increase one's score is by stealing other artworks being displayed in security glass showcases: These works of art are more valuable than the paintings and only take one slot in the inventory, but are harder to get because the showcase has to be opened first before the stealing process can take place. The more time-consuming, but safer approach here is to silently cut it open by using a glass cutter as shown in Figure 5.4(a): This action is performed analogue to stealing paintings, but the damage done to the showcase is preserved (see also chapter 4.1); if the player cuts the showcase a bit, interrupts the process and then comes back later, the time needed for cutting it doesn't reset. The faster, but more dangerous method (because it alarms nearby watchmen) is to press the item button while being in the cutting process (denoted with a hammer symbol in the item window as seen in Figure 5.4(a)), which causes the player character to loudly smash the security

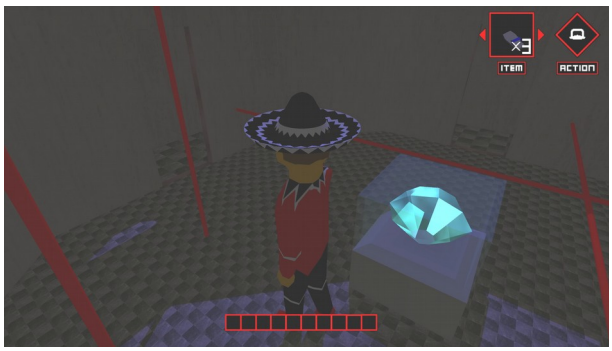
glass with a hammer and quickly steal it – the result is seen in Figure 5.4(b). There, the symbolic crafting tools in the action button display also denote that, analogue to the painting principle, a faked version of the stolen artwork can be put back in the showcase after it has been stolen; but in this specific case (of a smashed showcase), the watchmen cannot be fooled.

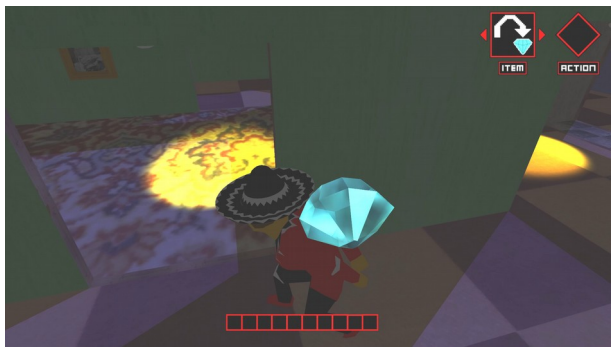
Figures 5.5(a) and 5.5(b): A player putting their loot in their personal vault (left) and being apprehended by a watchman (right).

But for the paintings' and showcase artworks' values being added to a player's score, they have to be put in one's personal vault found in one's personal safe area marked with secret doors in the respective player's colour (see Figure 5.5.(a)): This is done by pressing and holding the action button near the (also colour-coded) vault (that only the player with the same colour can interact with) until the message "THE LOOT YOU CARRIED IS SAFE NOW!" appears. This also clears the artwork inventory, which is important because when all 10 slots are taken by loot, it is not possible to steal additonal treasures. But if a player character is apprehended by a watchman (see Figure 5.5(b); there is also a faked papier mâché artwork to be seen in the cut open showcase), they lose all treasures carried in their inventory.

Figures 5.6(a) and 5.6(b): A player using the turbo shoes item (left) and sneaking past a vertically moving security camera (right).

To make it easier for the players to steal as many artworks as possible, the items purchased in the shop at the beginning of the match can be used: Some, like banana peels other characters slip on, can be used to directly cause trouble for competing players or watchmen. Others, like the turbo shoes (visible on the player's feet in Figure 5.6(a)), boost specific status values of its user: In this example, their running speed increases, while e.g. the usage of the ninja sandals item enables the player to run without making noises that alarm nearby guards as when they are running without using it (normally, only moving at walking speed and crouching are silent means of movement). In Figure 5.6(b), the player crouches to sneak past a moving camera (vertically and horizontally moving ones are implemented): If they get into the camera's field of view that is illustrated by an orange searchlight, the light turns red and the alarm goes off.





Figures 5.7(a) and 5.7(b): A player reaching the central diamond chamber (left) and carrying the huge diamond back to his safe area (right).

There are two events that cause a match to end, one of them being one of the players making their way to the central diamond chamber (that is heavily secured by an especially thick showcase security glass that takes more time to cut and alarm-triggering moving laser beams) as seen in Figure 5.7(a), stealing the huge diamond displayed there (what causes the red alarm coming into effect after a short countdown, provoking all patrolling watchmen to run to the diamond chamber then) and carrying the heavy gem (whose weight detains the carrier from running, see Figure 5.7(b)) back to their safe area: In this case, the game immediately ends and said player earns a very large score bonus that very well may be the key to winning the game.

Figure 5.8: The end screen of a match.

But if a player tries to steal the diamond, but fails and the watchmen are then able to take the gem back to its showcase and seal it off, the watchmen's "backup countdown" starts and when it has ended, every player is automatically apprehended and loses all the artworks in their inventory: Here, the winner is determined by the score of each player (without anyone possessing the diamond), as usual based on the value of the artworks brought to one's personal vault.

5.2 Output data and integration with the surveillance system

After a *Gangsters and Pranksters* match has ended, replay information about it is exported in the text documents "eventData.txt" and "replayData.txt" (paths follow). If viewpoints and timestamps are supplied manually, it is already possible to generate replays; the game is not yet integrated with the surveillance system in its current state (when it is integrated, these information is supplied by the surveillance system itself). To achieve such integration, the surveillance system has to be supplied with all the information found in "eventData.txt", "replyData.txt" and "EventHierarchy.txt" (the contents of the last one being inalterable and just denoting the importance hierarchy of the event types). Detailed information about the notations used in these text documents follows.

Event data
The generated list of events happening during the match is stored in "Binary\Replay\eventData.txt", using the following notation:

Event_ID; Start_Timestamp; End_Timestamp; Event_Type_Text

Event_ID is an integer, *Start_Timestamp* and *End_Timestamp* are float values and *Event_Type_Text* is a String. *Start_Timestamp* and *End_Timestamp* denote when the event starts and ends and if the special case *Start_Timestamp == End_Timestamp* occurs, it means that it is an event without a duration.

See "Binary\Replay\EventHierarchy.txt" for the texts denoting the events that are used here. If a relationship with another event exists, the Event_Type_Text is extended by "Because of event [Event_ID]."

Replay data
A list of all game objects with information about their (over time changing) locations is stored in "Binary\Replay\replayData.txt". The exported information here is much more complex than in the case of the event data; every row in the document represents a timestamp plus the data of all objects at this moment and the notation looks like this (when there are n objects):

Timestamp Object_Data_1 Object_Data_2 … Object_Data_n

Nesting follows because this notation is only the highest level of information; there are three different notations for *Object_Data*, depending on the object said data is referring to: It could be a player character, CPU-controlled watchman or not animated object:

Object_Data version 1 (Player): ID Player_Name Player_Number Position_Information Animation_Information
Object_Data version 2 (Watchman): ID Watchman_Name Position_Information Animation_Information
Object_Data version 3 (Not_Animated_Object): ID Object_Name Position_Information

Position_Information uses the same notation in all three cases:

Position_Information: x_Coordinate y_Coordiante z_Coordinate Quaternion_x Quaternion_y Quaternion_z Quaternion_w

For *Animation_Information*, there are two different notation types depending on the object:

Animation_Information (Player version): Speed Crouch_Speed Animation_Layer_Number Animation_Layer_Information
Animation_Information (Watchman version): Speed Animation_Layer_Number Animation_Layer_Information

The length of the *Animation_Layer_Information* depends on the *Animation_Layer_Number* n):

Animation_Layer_Information: Animation_Layer_1 Animation_Layer_2 … Animation_Layer_n

An important annotation regarding the following information about *Animation_Layer* follows: "*Transition_Boolean*" denotes a transition between two animations states. If it is "true", four values follow, if "false", only two values follow.

Animation_Layer (if Transition_Boolean==true): Transition_Boolean Current_Animation_State_Hash Normalized_Time_of_the_Current_Animation_State Next_Animation_State_Hash Normalized_Time_of_the_Transition
Animation_Layer (if Transition_Boolean==false): Transition_Boolean Current_Animation_State_Hash Normalized_Time_of_the_Current_Animation_State

Player_Name, *Watchman_Name* and *Object_Name* are Strings, *Transition_Boolean* is a bool data type, *Current_Animation_State_Hash* and *Next_Animation_State_Hash* are integer values and everything else is denoted as float.

Event hierarchy
In "Binary\Replay\EventHierarchy.txt", the texts used for the above discussed event data are found in form of a linear list: The smaller the index value, the more important is the event type!

Finalising the replay
After the surveillance system's work based on the supplied information in the three said text documents was finished, a list of viewports and time data is returned that denotes which viewports at which time are needed by the surveillance system. This list has to be stored in "Binary\Replay\viewportList.txt" and has to be notated in the following form (all values being the float type):

Timestamp x_Coordinate y_Coordiante z_Coordinate Quaternion_x Quaternion_y Quaternion_z Quaternion_w

Based on this list of specific viewports at specific times, *Gangsters and Pranksters* then returns screenshots (.png format) that are safed consecutively numbered in the "Binary\Replay\" folder as "image1.png", "image2.png", … , "imagen.png" (in the case there are n needed screenshots). Finally, the surveillance system has to be supplied with these images to merge them to generate the video replay of the match!

CHAPTER 6

Critical reflection

6.1 Comparison with related work

As stated in the introduction, *Gangsters and Pranksters* borrows elements from a variety of games but strives to become more than a mere mixture of well-known elements. Of course, our game can by no means compete with the complexity of "serious" stealth games like *Metal Gear Solid* and doesn't offer the variety of fleshed-out arena fighting games like GoldenEye 007, but turned out to be an uncomplicated, decent little game that is especially fun with some human opponents – and exactly that is what, in addition to the compatibility to the virtual surveillance system, we tried to achieve.

6.2 Discussion of open issues

Like in most development processes there was, of course, a variety of features that eventually had to be cut – especially tragic if they were already nearly finished (see the appendix): For example, we designed our PlayerCharacterController in a way we could easily experiment with different character stats like stealing speed, walking and running speed etc. because we planned to integrate three additional player characters – an old gentleman thief, the stereotype burglar Ede and a thief that actually is a realistic proportioned cat (which was already fully modelled and partly animated by Alex), each of them with their different special item (of course, El Matador, the last remaining player character, was planned to have one of them, too) for a single usage during a match. Sadly, all these elements had to be cut due to the expenditure of time – like the planned alarm-secured glass doors in the central rooms or cosmetic details like different motives for the paintings and different artwork models.

Speaking of cosmetic elements: While we tried to minimize them, there still are a couple of cosmetic issues in the final product (that luckily do not harm the gameplay experience). E.g. the effect of the walls becoming transparent when the current camera angle demands it (really important gameplay-wise for easier navigation) may not be as "pretty" as we planned it, some vertically moving surveillance cameras optically seem (according to their searchlights) to see "through walls" (but actually, that doesn't happen gameplay-wise) and sometimes, when watchmen run toward a thief, the latter seems to "bounce" off them, sometimes through walls (but this makes no difference gameplay-wise because after the thief got close to the player character, the latter is apprehended in any case and "respawns" in his safe area).

CHAPTER 7

Summary and future work

Summing things up, working on *Gangsters and Pranksters* was a complex and very time-consuming work, but especially a really enriching experience. While, of course, many things did not turn out as imagined in the planning phase and a variety of elements had to be changed or removed, our gameplay-centered approach seemed to be a valid direction: We invested much time in the architecture and design of the game world and the gameplay core, so while some planned additional amendments had to be removed, the essence of the game has been preserved in the final product and we hope to have contributed a little bit to the scientific field of game surveillance systems.

As for future work, we are definitely thinking about improving our game on our own to add all the features we originally wanted to include and eventually publish it.

APPENDIX A

Work-sharing

This Bachelor thesis was a co-production of Alexander Bayer and Lukas Bydlinski – a detailed list of the work-sharing follows.

Scripts [by Alex and Lukas]:

(Please check the preliminary information and the "[Lukas]", "[/Lukas]", "[Alex]" and "[/Alex]" tags found in the C# classes' comments for very detailed information regarding the work-sharing.)

ArtworkController.cs
GameControllerScript.cs
PaintingController.cs
PlayerCharacterController.cs
Showcase.cs
Tags.cs

Scripts [by Alex]:

BagController.cs
BallController.cs
BananaController.cs
DiamondController.cs
CurtainController.cs
EnemyAI.cs
EnemySight.cs
Event.cs
EventHandler.cs
EventHierarchy.cs
GameSetup.cs
HashIDs.cs
Item.cs
LaserRay.cs
LastPlayerSighting.cs
LoadReplay.cs
Loot.cs
MenuController.cs
NailController.cs
PlayerMovement.cs
ReplaySystem.cs
RotateUI.cs
RotationOfCamera.cs
ScoreScreen.cs
ScoreSetup.cs

SecurityCamera.cs
ThirdPersonCamera.cs
UniqueID.cs

Character models and animations [by Alex]:

ElMatador.fbx
Watchman.fbx

Modelled game world [by Lukas]:

MuseumQuarterTexturedFinal.blend

Modelled objects [by Alex]

CautionGlass.blend
EnemyMarker2.blend
Nail.blend
OutOfOrder.blend
SecurityCamera.blend
SecurityCameraFOV.blend

Modelled objects [by Lukas]

Artwork.blend
BananaPeelModel.blend
BouncyBallModel.blend
CurtainModel.blend
CurtainRailModel.blend
Diamond.blend
ElMatadorHammerModel.blend
ElMatadorHat.blend
GlassCutterModel.blend
MuseumDoorModel.blend
MuseumGardenDoorModel.blend
MuseumRoofModel.blend
MuseumWindowFrameModel.blend
MuseumWindowGlassModel.blend
NinjaSandalsModel.blend
Painting.blend
SafeHandleModel.blend
SafeModel.blend
ShowcaseGlass.blend
ShowcaseGlassCutOpen.blend
ShowcaseGlassSmashed.blend
ShowcasePillar.blend

SuperGlassCutterModel.blend
TurboShoesModel.blend
WatchmanCapModel.blend

Textures [by Alex]:

CautionGlassTexture.png
EnemyMarkerPlayer1Texture.png
EnemyMarkerPlayer2Texture.png
EnemyMarkerPlayer3Texture.png
EnemyMarkerPlayer4Texture.png
OutOfOrderTexture.png

Textures [by Lukas]:

ArtworkTexture.jpg
ArtworkTextureFakeVersion.jpg
BananaPeelTexture.png
BouncyBallTexture.png
Bretter2.jpg
CanvasPaintingWinterImpressionsTexture.png
CanvasPaintingWinterImpressionsTextureFakeVersion.png
CurtainRailTexture.png
CurtainTexturePlayer1.png
CurtainTexturePlayer2.png
CurtainTexturePlayer3.png
CurtainTexturePlayer4.png
ElMatadorHammerTexture.png
ElMatadorHatTexture.png
ElMatadorTexturePlayer1.png
ElMatadorTexturePlayer2.png
ElMatadorTexturePlayer3.png
ElMatadorTexturePlayer4.png
ElMatadorGlassCutterTexture.png
Hausmauer1.jpg
Innenwand3Player1.jpg
Innenwand3Player2.jpg
Innenwand3Player3.jpg
Innenwand3Player4.jpg
Innenwand3weiss.jpg
MuseumDoorCaseTexture.jpg
MuseumDoorKnob.jpg
MuseumDoorTexture.jpg
MuseumGardenDoorTexture.jpg
MuseumRoofTexture.jpg
Museumswand3.jpg
NinjaSandalsTexture.png
SafeHandleTexturePlayer1.png

SafeHandleTexturePlayer2.png
SafeHandleTexturePlayer3.png
SafeHandleTexturePlayer4.png
SafeTexturePlayer1.png
SafeTexturePlayer2.png
SafeTexturePlayer3.png
SafeTexturePlayer4.png
Schachbrett4.jpg
ShowcasePillarTexture.png
Steinmauer.jpg
SuperGlassCutterTexture.png
Teppich.jpg
TurboShoesTexture.png
Uebergangsfliesen3.jpg
WatchmanCapTexture.png
WatchmanTexture.png
Wiesenausschnitt.jpg
Zwischenfliesen5.jpg

HUD graphics [by Alex]:

HUDInteractCutting.png
HUDInteractDiamond.png
HUDInteractSafe.png
HUDInteractSteal.png
HUDInventoryArtwork.png
HUDInventoryPainting.png
HUDInventoryPlayer1.png
HUDInventoryPlayer2.png
HUDInventoryPlayer3.png
HUDInventoryPlayer4.png

HUD graphics [by Lukas]:

HUDFirstDigit0.png
HUDFirstDigit1.png
HUDFirstDigit2.png
HUDFirstDigit3.png
HUDFirstDigit4.png
HUDFirstDigit5.png
HUDFirstDigit6.png
HUDFirstDigit7.png
HUDFirstDigit8.png
HUDFirstDigit9.png
HUDInteractFake.png
HUDInteractPlayer1.png
HUDInteractPlayer2.png
HUDInteractPlayer3.png

HUDInteractPlayer4.png
HUDItemDropDiamond.png
HUDItemHammer.png
HUDItemNoItem.png
HUDItemPlayer1.png
HUDItemPlayer2.png
HUDItemPlayer3.png
HUDItemPlayer4.png
HUDSecondDigit0.png
HUDSecondDigit1.png
HUDSecondDigit2.png
HUDSecondDigit3.png
HUDSecondDigit4.png
HUDSecondDigit5.png
HUDSecondDigit6.png
HUDSecondDigit7.png
HUDSecondDigit8.png
HUDSecondDigit9.png
InventoryFullMessagePlayer1.png
InventoryFullMessagePlayer2.png
InventoryFullMessagePlayer3.png
InventoryFullMessagePlayer4.png
LootIsSafeMessagePlayer1.png
LootIsSafeMessagePlayer2.png
LootIsSafeMessagePlayer3.png
LootIsSafeMessagePlayer4.png

Operations using the graphical Unity editor [by Alex]:

- The assembly of the four museum quarters to form the completed game world
- The integration of all the HUD graphics listed above but HUDItemHammer.png, HUDFirstDigit0-9.png and HUDSecondDigit0-9.png into the game scene

Operations using the graphical Unity editor [by Lukas]:

- Placing objects and watchmen in the scene and setting waypoints for the latter [Lukas]
- The integration of the following HUD graphics into the game scene: HUDItemHammer.png, HUDFirstDigit0-9.png and HUDSecondDigit0-9.png

Menu graphics [by Alex]:

ArrowLeft.png
ArrowRight.png
Player1Character.png
Player2Character.png
Player3Character.png
Player4Character.png

Event hierarchy list [by Lukas]:

EventHierarchy.txt

APPENDIX B

Unused content

A list of elements we worked on – many of them being finished or nearly finished – that did not make it into the final product follows (see our submission's folder "unused").

"Real character" experiment [by Alex and Lukas]:

AlexAsElMatadorVersion1.blend
AlexAsElMatadorVersion2.blend
ElMatadorFaces.jpg

Unused character model and animations [by Alex]:

BurglarCatRiggedPartlyAnimated.blend

Unused HUD elements [by Alex]:

EnemyMarker.blend
EnemyMarkerPlayer1TextureOriginal.png
EnemyMarkerPlayer2TextureOriginal.png
EnemyMarkerPlayer3TextureOriginal.png
EnemyMarkerPlayer4TextureOriginal.png
HUDCrosshairs.png

Unused object and texture [by Lukas]:

(It was planned to move this metal plate during the process of the diamond being sealed off.)

ProtectivePlateModel.blend
ProtectivePlateTexture.jpg

Modelled special items for the planned but unused characters [by Lukas]:

EdeHandCannonModel.blend
EdeHandModel.blend
GentlemanUmbrellaModel.blend
GentlemanUmbrellaRotorModel.blend
ToyMouseKeyModel.blend
ToyMouseModel.blend

Textures for these special items [by Lukas]:

EdeHandCannonTexture.png
EdeHandTexture.png
GentlemanUmbrellaTexture.png
GentlemanUmbrellaRotorTexture.png
ToyMouseKeyTexture.png
ToyMouseTexture.png

Hammer models for the unused characters [by Lukas]:

BurglarCatHammerModel.blend
EdeHammerModel.blend
GentlemanHammerModel.blend

Textures for these hammers [by Lukas]:

BurglarCatHammerTexture.png
EdeHammerTexture.png
GentlemanHammerTexture.png

GlassCutterModel.blend-textures for the unused characters [by Lukas]:

(No glass cutter was planned for the burglar cat because he was supposed to just use his claws.)

EdeGlassCutterTexture.png
GentlemanGlassCutterTexture.png

Unused alternative painting texture [photo taken and edited by Lukas]:

OilPaintingRiverBridge.png

Bibliography

[1] Andrews, Marcus: "Game UI Discoveries: What Players Want" (2010). Available online: http://www.gamasutra.com/view/feature/132674/game_ui_discoveries_what_players_.php (Accessed on June 10th, 2015)

[2] Benmokhtar, Rachid and Ivan Laptev: "INRIA-WILLOW at TRECVid 2010: Surveillance Event Detection" (2010). Available online: http://www-nlpir.nist.gov/projects/tvpubs/tv10.papers/inria-willow.pdf (Accessed on June 10th, 2015)

[3] Blanch, Renaud and Michel Beaudouin-Lafon: "Programming rich interactions using the hierarchical state machine toolkit". In "Proceedings of the working conference on Advanced visual interfaces (AVI '06)". p.51-58. ACM: New York, NY 2006. Available online: **http://dl.acm.org/citation.cfm?id=1133275** (Accessed on August 17th, 2015)

[4] Champandard, Alex J.: "AI Game Development: Synthetic Creatures with Learning and Reactive Behaviors". New Riders Publishing: USA 2003/04.

[5] Collins, Robert T., Alan J. Lipton and Takeo Kanade: "Introduction to the Special Section on Video Surveillance". In: IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 8, August 2000. p.745-746. Available online: http://www-preview.ri.cmu.edu/pub_files/pub2/collins_robert_2000_2/collins_robert_2000_2.pdf (Accessed on Jone 10th, 2015)

[6] Dikmen, Mert, Huazhong Ning, Dennis J. Lin, Liangliang Cao, Vuong Le, Shen-Fu Tsai, Kai-Hsiang Lin, Zhen Li, Jianchao Yang and Thomas S. Huang / Lv, Fengjun, Wei Xu, Ming Yang, Kai Yu, Zhao Zhao, Guangyu Zhu and Yihong Gong: "Surveillance Event Detection". Department of Electrical and Computer Engineering University of Illinois Urbana, Coordinated Sciences Laboratory & Beckman Institute for Advanced Sciences / NEC Laboratories America, Inc.: Urbana, IL / Cupertino, CA 2009
Available online: http://www-nlpir.nist.gov/projects/tvpubs/tv8.papers/ifp-uiuc-nec.pdf (Accessed on June 10th, 2015)

[7] Ebert, David S., F. Kenton Musgrave, Darwyn Peachey, Ken Perlin and Steven Worley: "Texturing & Modeling: A Procedural Approach". Third Edition. Elsevier Science (USA), Morgan Kaufmann Publishers: San Francisco, CA 2003.

[8] Eisenacher, C., S. Lefebvre and M. Stamminger: "Texture Synthesis From Photographs". Article first published online (April 24th, 2008). Available online: http://onlinelibrary.wiley.com/doi/10.1111/j.1467-8659.2008.01139.x/abstract (Accessed on August 18th, 2015)

[9] Fagerholt, Erik and Magnus Lorentzon: "Beyond the HUD: User Interfaces for Increased Player Immersion in FPS Games". Master of Science Thesis. Chalmers University of Technology, Department of Computer Science and Engineering, Division of Interaction Design: Göteborg, Sweden, 2009. Available online: **http://publications.lib.chalmers.se/records/fulltext/111921.pdf** (Accessed on August 17th, 2015)

[10] Feil, John and Marc Scattergood: "Beginning Game Level Design". Thomson Course

Technology PTR: Boston, MA 2005. Available online: http://www.tsqtsg.cn:88/date/H/D0011731.pdf (Accessed on June 10th, 2015)

[11] Gouraud, Henri: "Continuous Shading of Curved Surfaces", In: "IEEE Transactions on Computers". Volume C-20, Issue 6, June 1971. p.623-629. Available online: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=1671906&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D1671906 (Accessed on August 17th, 2015)

[12] Grosz, Elizabeth: "Architecture from the Outside: Essays on Virtual and Real Space". Massachusetts Institute of Technology: Cambridge, MA 2001. Available online: http://projectlamar.com/media/groszarchitecture.pdf (Accessed on June 10th, 2015)

[13] Kay, Alan: "User Interface: A Personal View" (1989). In: Packer, Randall and Ken Jordan [Editors]: "Multimedia: From Wagner to Virtual Reality". Expanded Edition. p.121-131. W. W. Norton & Company: New York / London 2001-2002. Available online: http://proteus.fau.edu/practicum/texts/kay.pdf (Accessed on June 10th, 2015)

[14] Lidén, Lars and Valve Software: "Strategic and Tactical Reasoning with Waypoints“. In: „AI Game Programming Wisdom". p.211-220. Charles River Media: Hingham, MA 2002. Available online: **http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.132.5049** (Accessed on August 18th, 2015)

[15] Llanos, Stein C. and Kristine Jørgensen: "Do Players Prefer Integrated User Interfaces? A Qualitative Study of Game UI Design Issues". Proceedings of DiGRA 2011 Conference: Think Design Play. Available online: http://www.digra.org/wp-content/uploads/digital-library/11313.34398.pdf (Accessed on June 10th, 2015)

[16] Maher, Mary Lou, Ning Gu and Fei Li: "Visualisation and Object Design in Virtual Architecture". Key Centre of Design Computing, Faculty of Architecture, University of Sidney: Sidney 2001. Available online: http://www.researchgate.net/profile/Mary_Maher/publication/30872591_Visualisation_and_object_design_in_virtual_architecture/links/54c25cf90cf256ed5a8ccfe6.pdf (Accessed on June 10th, 2015)

[17] Maher, Mary Lou, Simeon Simoff, Ning Gu and Kok Hong Lau: "Designing Virtual Architecture". Key Centre of Design Computing, University of Sidney: Sidney 2000. Available online: http://maryloumaher.net/Pubs/2000pdf/caadria2000.pdf (Accessed on June 10th, 2015)

[18] Masuch, Maic and Niklas Röber: "Game Graphics Beyond Realism: Then, Now, and Tomorrow".
Otto-von-Guericke University Magdeburg, Games and Graphics Research Group, Institute for Simulation and Computer Graphics: Magdeburg (n.d.). Available online: http://www.digra.org/wp-content/uploads/digital-library/05150.48223.pdf (Accessed on June 10th, 2015)

[19] Mindek, Peter, Ladislav Čmolík, Ivan Viola, Eduard Gröller and Stefan Bruckner: "Automatized Summarization of Multiplayer Games". In: "Spring Conference on Computer Graphics 2015" (April 2015). p.93-100. Available online: https://www.cg.tuwien.ac.at/research/publications/2015/mindek-2015-mc/ (Accessed on August 18th, 2015)

[20] Mitchell, Jeffrey and Ben Shneiderman: "Dynamic versus static menus: an exploratory comparison". In: "SIGCHI Bull. 20, 4 (April 1989)". p.33-37. ACM: New York, NY 1989. Available online: http://dl.acm.org/citation.cfm?id=67247 (Accessed on August 17th, 2015)

[21] Osborne, Martin J. and Ariel Rubinstein: "A Course in Game Theory". Massachusetts Institute of Technology / The Maple-Vail Book Manufactoring Group: Cambridge, MA 2004 / Binghamton, New York 1994.

[22] Salen, Katie and Eric Zimmerman: "Rules of Play: Game Design Fundamentals". Massachusetts Institute of Technology: Cambridge, MA 2004.

[23] Sato, Yoichi, Mark D. Wheeler and Katsushi Ikeuchi: "Object shape and reflectance modeling from observation". In: "Proceedings of the 24th annual conference on Computer graphics and interactive techniques (SIGGRAPH '97)". p.379-387. ACM Press/Addison-Wesley Publishing Co.: New York, NY 1997. Available online: http://dl.acm.org/citation.cfm?id=258885 (Accessed on August 17th, 2015)

[24] Serrano, Juan M. and Sergio Saugar: "Operational semantics of multiagent interactions". In: "Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems (AAMAS '07)". ACM: New York, NY 2007. Article 130, 8 pages. Available online: http://dl.acm.org/citation.cfm?id=1329285 (Accessed on August 17th, 2015)

[25] So, Clifford, George Baciu and Hanqiu Sun: "Reconstruction of 3D virtual buildings from 2D architectural floor plans". In: "Proceedings of the ACM symposium on Virtual reality software and technology (VRST '98)". p.17-23. ACM: New York, NY, 1998. Available online: http://dl.acm.org/citation.cfm?id=293704 (Accessed on August 17th, 2015)

[26] Sorenson, Nathan and Philippe Pasquier: "Towards a Generic Framework for Automated Video Game Level Creation". In: Di Chio, Cecilia, Stefano Cagnoni, Carlos Cotta, Marc Ebner, Anikó Ekárt, Anna I. Esparcia-Alcazar, Chi-Keong Goh, Juan J. Merelo, Ferrante Neri, Mike Preuß, Julian Togelius, Georgios N. Yannakakis [Editors]: "Applications of Evolutionary Computation". Lecture Notes in Computer Science, Vol. 6024. p. 131-140. Springer-Verlag: Berlin Heidelberg 2010. Available online: http://link.springer.com/chapter/10.1007/978-3-642-12239-2_14#page-1 (Accessed on August 17th, 2015)

[27] Turk, Greg: "Generating textures on arbitrary surfaces using reaction-diffusion". In: "ACM SIGGRAPH Computer Graphics". Volume 25, Issue 4 (July 1991). p.289-298. ACM: New York, NY 1991. Available online: http://dl.acm.org/citation.cfm?id=122749 (Accessed on August 18th, 2015)

[28] Van den Bergh, Jan and Karin Coninx: "Towards modeling context-sensitive interactive applications: the context-sensitive user interface profile (CUP)". In: "Proceedings of the 2005 ACM symposium on Software visualization (SoftVis '05)". p. 87-94. ACM: New York, NY 2005. Available online: http://dl.acm.org/citation.cfm?id=1056030 (Accessed on August 17th, 2015)

Figures:

Figure 1.1: http://multimedialu.googlecode.com/svn/halloffame/WS2011/k02.jpg (Accessed on June 11th, 2015)

Figure 1.2:
http://www.cg.tuwien.ac.at/courses/CG23/HallOfFame/2013/screenshots/taleofthecat.png
(Accessed on June 11th, 2015)

Figure 2.1(a): http://www.oldschooljunkie.com/wp-content/uploads/2011/03/Castlevania-Boss-Battle.png
(Accessed on June 13th, 2015)

Figure 2.1(b): http://upload.wikimedia.org/wikipedia/en/1/14/ZELDA_OCARINA_OF_TIME.jpg
(Accessed on June 13th, 2015)

Figure 2.1(c): http://static.giantbomb.com/uploads/original/0/7465/968375-n64_controller.jpg
(Accessed on August 22nd, 2015)

Figure 2.2(a): www.gamasutra.com/db_area/images/feature/4286/terminology.gif
(Accessed on June 13th, 2015)

Figure 2.2(b): http://www.gamasutra.com/db_area/images/feature/4286/deadspace1.jpg
(Accessed on June 13th, 2015)

Figure 3.1: Extracted from the Unity editor of our own project (brightness was increased by 20% in the LibreOffice paper).

Figure 3.2: Extracted from the Unity editor of our own project (brightness was increased by 20% in the LibreOffice paper).

Figure 3.3: Scan of my own design document.

Figure 3.4: Photographs made by myself.

Figure 3.5: Extracted from the Blender editor of our own project.

Figure 3.6: Extracted from the Blender editor of our own project.

Figure 3.7: Photograph taken and edited by me.

Figure 4.1: Extracted from the Blender editor of our own project.

Figure 4.2: Extracted from the Blender editor of our own project.

Figure 4.3: Extracted from the Unity editor of our own project.

Figure 4.4: Image created by me using GIMP.

Figure 5.1: Extracted from a self-made video of our game.

Figure 5.2: Extracted from a self-made video of our game.

Figure 5.3(a): Extracted from a self-made video of our game (brightness was increased by 20% in

the LibreOffice paper).

Figure 5.3(b): Extracted from a self-made video of our game (brightness was increased by 20% in the LibreOffice paper).

Figure 5.4(a): Extracted from a self-made video of our game (brightness was increased by 20% in the LibreOffice paper).

Figure 5.4(b): Extracted from a self-made video of our game (brightness was increased by 20% in the LibreOffice paper).

Figure 5.5(a): Extracted from a self-made video of our game (brightness was increased by 20% in the LibreOffice paper).

Figure 5.5(b): Extracted from a self-made video of our game (brightness was increased by 20% in the LibreOffice paper).

Figure 5.6(a): Extracted from a self-made video of our game (brightness was increased by 20% in the LibreOffice paper).

Figure 5.6(b): Extracted from a self-made video of our game (brightness was increased by 20% in the LibreOffice paper).

Figure 5.7(a): Extracted from a self-made video of our game (brightness was increased by 20% in the LibreOffice paper).

Figure 5.7(b): Extracted from a self-made video of our game (brightness was increased by 20% in the LibreOffice paper).

Figure 5.8: Extracted from a self-made video of our game.

Used technologies:

"Gem Shader: Version 1.04 (October 19th, 2010)“ by Unity Technologies. Available online: https://www.assetstore.unity3d.com/en/#!/content/3 (Accessed on June 16th, 2015)

Video games designed by us:

The Tale of the Cat: http://www.cg.tuwien.ac.at/courses/CG23/HallOfFame/2013/games/taleofthecat.zip (Accessed on June 11th, 2015)

The Tale of the Snail: http://multimedialu.googlecode.com/svn/halloffame/WS2011/k02.html (Accessed on June 11th, 2015)

Referenced video games:

"Assassin's Creed". Ubisoft, PlayStation 3: Europe 2007.

"Batman: Arkham Asylum". Eidos/Warner Bros./Rocksteady Studios, PlayStation 3: Europe 2009.

"Dead Space". Electronic Arts/EA Redwood Shores, PlayStation 3: Europe 2008.

"GoldenEye 007". Nintendo/Rare, Nintendo 64: Europe 1997.

"Metal Gear Solid". Konami, PlayStation: Europe 1999.

"Super Mario Kart". Nintendo, Super Nintendo Entertainment System: Europe 1993.

"The Legend of Zelda: Ocarina of Time". Nintendo, Nintendo 64: Europe 1998.

Referenced movies:

"Who Framed Roger Rabbit?" Robert Zemeckis, Buena Vista/Disney/Touchstone: USA 1998.

Used tutorials:

[T1] "Blender 2.59 Modeling Tutorial, Modeling Walls & Rooms Quickly" by Youtube user "weeliano", uploaded August 30th, 2011. Available online: https://www.youtube.com/watch?v=5QXD_twk4gw (Accessed on June 10th, 2015)

[T2] "Project: Stealth (Unity 4X)" @ unity3d.com. Last changed on June 2nd, 2015. Available online: https://unity3d.com/learn/tutorials/projects/stealth (Accessed on June 10th, 2015)