The Design and Development of Computer Games

Markus Gross, Robert W. Sumner, Nils Thürey

ETH Zurich, Disney Research Zurich

The design of modern computer games is as much an art as painting, sculpting, music, or writing. Albeit relatively young and still evolving, game design is highly complex and requires a broad spectrum of artistic and technical skills. The following contribution reviews the process of designing and developing modern computer games. We will acquaint the reader with the most important fundamentals of game design, walk through the various stages of the design process and highlight the specifics of each stage. We will analyze and elucidate the domain-independent principles underlying game design, such as iteration, evolution, consistency, and others. Our findings are illustrated by three case studies on computer games developed within the scope of the ETH Game Programming Laboratory class.

1. Introduction

Over the past 50 years, video games evolved from simplistic 2D line drawings into highly complex software systems with rich media content, creative artwork, and complex game flow. Today, such video games share a 30 billion dollar market and their socio-economic impact is significant. Individual AAA game projects operate with budgets of tens of millions of dollars, and their interdisciplinary game design teams comprise both artists and engineers. Video games have also been the single most important force to drive the hardware development of personal computing, and contemporary game consoles, such as Sony's Playstation 3 or Microsoft's Xbox 360, rival the compute power of supercomputers from days gone bye.

The first computer game, called *Spacewar*, was developed in 1962 at MIT (Graetz 1981, pp. 56-67) and subsequently commercialized as an arcade game in 1971. A further milestone in early game development was *Pong*, a very simple version of tennis developed in 1972. Pong was later released for home consoles and delighted a generation of video game enthusiasts. The continual increase in graphical complexity was evident in *Space Invaders*, a 1978 arcade game with colored spaceships displayed on raster screens. One of the most popular video games of the early eighties

was *Pacman*, a game that appealed equally to both male and female demographics. The early eighties also spawned the first version of Nintendo's *Donkey Kong*, the beginning of a highly successful sequence of jump-and-run games. In the late seventies and early eighties, the first cartridge-based home console systems from Atari (2600 and 5200) and Nintendo (NES) came out, along with the blockbuster games *Super Mario* Bros. and *Zelda*. A major milestone of the late eighties was Nintendo's Game Boy, as well as the hugely popular title *Tetris*. In the late eighties and early nineties, game hardware moved to 16-Bit architectures. Dramatically increased compute and graphics powers provided the platform for a series of graphically, artistically, and technically more complex game designs. Prominent examples from this era comprise the *Mario* and *Zelda* sequels, *Sonic the Hedgehog*, and others. The game titles from the early nineties clearly evidence the increased challenge in designing compelling video games with rich assets and graphics content. The mid nineties brought another quantum leap with Sony's Playstation and Nintendo's N64, the first consoles with 3D graphics acceleration. This time also marks the transition from 2D visuals to full three-dimensional content. The power of three-dimensionality was harnessed by many game genres, such as sports, adventure, action, shooting, and puzzle. Around 2000, additional game hardware was released, including the Playstation 2, Game Cube, Xbox and others. In parallel, personal computers were empowered by sophisticated 3D graphics accelerators. The release cycles of the latest generation game consoles, specifically Playstation 3 and Xbox 360, evidence the dramatically increased complexity and cost of game hardware design. Expressive 3D content, very rich media assets, computationally expensive physics, and online functionality characterize the broad spectrum of titles and genres available at present. The addition of online functionality alone has created its own genre of massive multiplayer online games, such as *World of Warcraft*.

The design of early video games focused mostly on technical and implementation issues, since gameplay was simple and artwork very limited. Compute resources were the limiting factor. Since then, the design of video games has evolved into a sophisticated art of its own, and it is being taught at art schools and universities all over the world. Game design poses great challenges as it encompasses artistic as well as technical elements, which, in a compelling design, cannot be separated from one another.

The art of game design is relatively young and still evolving. As such, the literature on game design is limited, but continuously growing. For instance, S. Rabin's book (Rabin 2005) provides a holistic view of the topic, including some of the more formal and abstract concepts, such as fun and flow. For more practical purposes, T. Fullerton et al. give a workshop-like recipe for the design of computer games (Fullerton 2004).

The goal of this chapter is to survey and discuss the basic principles and core components of modern video game design, development, and implementation. We will describe the process and its iterative nature, focus on the different stages of game development, and abstract generic design principles. While the paper focuses on generic principles applicable to all game genres, our experience draws upon a capstone class taught at the Computer Science Department at ETH Zürich over a period of 3 years (gamelab 2009). Our goals and experiences from this class are summarized in a recent paper (Sumner 2008), and many examples presented in this chapter are taken from the class projects.

The organization of this chapter follows the main stages of game design. After giving an overview of the process in section 2, we summarize the most important formal and technical elements of game design in section 3. Section 4 focuses on the various aspects of fun and motivation. We discuss conceptualization as the first major stage of game design in section 5, and section 6 addresses preproduction and rapid prototyping. The actual production stage is explained in section 7, while section 8 focuses on evaluation, quality assurance and game testing. Section 9 provides examples from our game development projects of previous years.

2. Overview and Stages of Game Design

The notion of design varies significantly among different disciplines within art, science, and engineering, and the ideas inherent in architectural design differ significantly from software design, genetic design, or fashion design. At present, there is no commonly accepted definition of "design," and neither are there commonly agreed-upon rules or axioms to leverage a systematic approach to design.

The term Game Design, is relatively recent and emphasizes the creative process underlying the conception and implementation of a new computer game. Such creative elements comprise the definition of the game's gameplay, as well as the creation of game assets, the underlying story, the artwork, the media content, and the software. By contrast, game development is focused on implementation, execution, and quality control, and is, as such, centered on engineering.

Game design is often conceived as an *iterative* process (Bates 2004), such as illustrated in figure 1. The core loop of iterative design entails the following steps. Ideas are generated, formalized, tested, evaluated, revised, and refined until no further improvements are possible. Then, the design transitions to the next stage where the process is repeated. The paradigm of iterative design thus describes the entire process from the initial idea to the playable game. The emphasis of *iterative design*

lies in frequent testing and early prototyping. It is noteworthy that the same paradigm is utilized in many other design fields as well. Most notably, the creation of computer-animated feature films employs a similar process, where artists and engineers collaborate on a major creative software project.

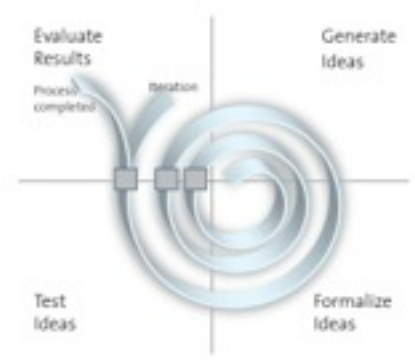

Fig. 1. : The concept of iterative design in its most generic form.

Iterative game design allows one to gain an early and deep understanding of the game's gameplay and foundations, the central elements of a successful design. Iteration, as opposed to a waterfall model of software development, is much more cost effective and helps uncover design flaws early on. As the iterative design spirals down from the first concepts to the final game, initially vague and unstructured ideas coalesce and solidify.

The progressive refinement of iterative game design is often broken up into four major phases, as depicted in figure 2 (Fullerton 2004, p. 197).

1. In the concept phase, brainstorming is used to generate ideas for the game's gameplay and other central concepts. During brainstorming, the team tries to come up with as many ideas as possible. Such ideas can be structured using concept cards, whiteboards, inspiration boards or charts. The goal is to narrow ideas down to the few most promising ones, and to write short summaries for each. In professional game development studios, concept art is often pinned up on the walls to inspire designers and developers. Another important paradigm of the concept phase is the creation of a physical prototype using paper, pen, and cardboard. This prototype serves for iterative playtesting and can be evaluated, revised, played, and tested until the designer is satisfied. Pictures and drawings symbolize main characters, assets, and artwork. First, concept documents are written and

concept art is created. The project management develops project plans and budgets, and contracts with publishers are finalized. This stage concludes with initial presentations (screenings) of the game, in which the team pitches the core elements of the game, the game's structure, the formal elements (see section 3), and the project plan in front of a peer audience. The paradigm of peer presentation is also used in animated feature films, where peers review the progression of the story on a regular basis during internal storyboard screenings.

2. In the pre-production phase, a first playable game prototype is developed. This software prototype should model the core gameplay, but it need not contain extensive artwork, media, or graphics. Frequently, software tools are employed for rapid prototyping. Again, the gameplay is tested, evaluated, revised and refined using the paradigm of iterative design. As an option, external game testers can be included. This process also helps to determine the required technology game assets. The design document constitutes a refined and extended version of the concept document and outlines every aspect of the game. It provides detailed descriptions of the flow, the artwork, and the game's formal elements. At the project management level, plans are refined and resources must be allocated. This planning helps avoiding a major redesign at a later, and hence, more expensive stage.

3. The production phase is focused on the development of all game assets, the design of levels, the development of required software technology, and the production of alpha-code. At this stage, structured design clearly prevails over flexibility. All team members verify the correctness of the design document. Project management challenges increase, software production must be coordinated, and the workload is distributed among team members. Individual strengths and weaknesses have to be considered. At the end of this phase, the first fully-functional version of the game, the alpha version, should be available. Iterative design is consistently applied to avoid major redesigns at this stage.

4. In the final, quality assurance (QA) phase, the alpha code is utilized for extensive game testing to discover flaws and glitches, and to fine-tune the game's difficulty level. The gameplay at this stage must be solid and the major focus is placed on usability and playtesting. The quality-assurance (QA) phase often involves external game testers. More recently, some professional game development teams have designed very systematic, almost scientific, approaches to game testing, including detailed statistical evaluations and visualizations (Halo 3).

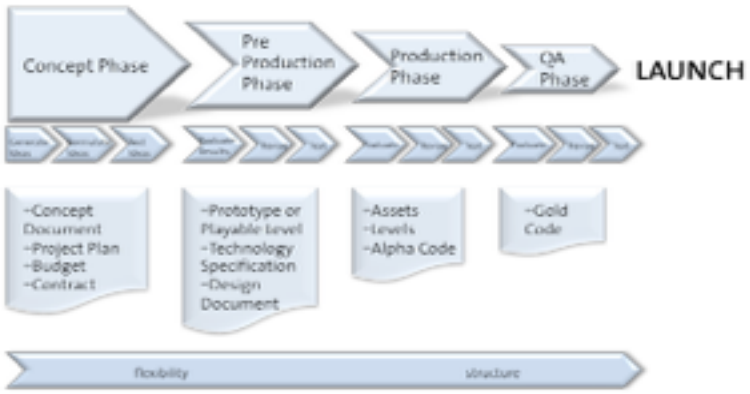

Fig. 2. : The 4 major phases of game design. As the design progresses, structure dominates flexibility (Adapted from Fullerton 2004, p. 197).

In summary, we can identify the following domain-independent design paradigms:

Iteration as opposed to waterfall

Peer Review and regular screenings

Rapid Prototyping at all stages

Evolution of a design document (story board)

Quality Assurance by systematic testing and evaluation

The following sections will discuss each of the phases of game design in greater detail and exemplify the above paradigms. As preparation, we will first acquaint the reader with some fundamental considerations about the nature of computer games.

3. Formal and Technical Elements of Computer Games

3.1.Game Type and Genre

At the most basic level, a computer game is characterized by the category into which it falls – the game *genre*. Such genres include action, adventure, shooters, jump-and-run, sports, racing, fighting, role-playing, simulations, strategy, puzzle, music, dance, artificial life, and others. These genres define the basic principles from which the player derives and retains his or her motivation to play. Motivation can be generated from hunting and killing, collecting items, direct competition, general reward for success, being a hero, having metaphysical abilities or superpowers, problem solving, learning control skills, socializing, and other similar principles. A more careful analysis of the aforementioned principles reveals that they fundamentally relate to the following four psychological abstractions:

- The archaic notion of hunters and gatherers
- Darwin‘s principle and direct competition as the survival of the strongest
- The desire to be superior and extraordinary
- The desire to manipulate and control other beings

3.2.Formal Elements

At the next level of refinement, a game is defined through its formal elements. The formal game elements refer to components that form the structure of a video game. Such components entail players, objective, procedures, rules, resources, conflicts, boundaries, outcome, goals, and others. We will discuss a few of the most significant components here.

One of the most important formal game elements is the number of *players*, their roles, and their interaction patterns. Player interactions can be single player versus game, multiplayer competition, multiplayer cooperative, team competition, etc. Some important player interaction patterns are illustrated in figure 3.

Fig. 3. Illustration of game player interaction patterns (adapted from Fullerton 2004, p. 46).

Of similar importance is the game's *objective*. The objective defines what the player is trying to accomplish. It is hence central for retaining the player's motivation and can include capturing, chasing, racing, rescuing, exploring, searching, and others. The objective is bounded by the rules of the game; it must be challenging, but achievable. Many modern designs utilize artificial intelligence (AI) to adjust difficulty levels to ensure the player always has an achievable goal. The objective further sets the tone of the game.

The game *procedures* define the methods of play and actions to achieve the objective. Such procedures involve actions and states, such as starting an action, progressing an action, or resolving an action. Procedures also have to consider system limitations, such as resolution or latency.

The game *rules* constitute the allowable player actions. Rules can define concepts (chess or poker), restrict actions (soccer), or determine actions (puzzles). The two major design paradigms for the game's rule base are:

Consistency and **Logical Correctness**

The *resources*, often called game assets, are used to achieve goals and objectives, but also cater to the above mentioned principles of collect-

ing and gathering. The subconscious aspect of such assets requires careful consideration during design as it significantly influences the player's motivation. Examples include lives, units, money, health, objects, terrain, and time.

Most similar to creative writing, the notion of *conflict* plays a central role in the conception of a computer game. Such conflict arises when a player is trying to achieve the goals within the game's constraints and boundaries. Conflict is essential to create a challenging game and to uphold motivation. Examples include obstacles, opponents, or dilemmas.

Finally, there is the game's *outcome*. Outcome describes the measurable achievement in the game. In most games, the outcome relates to a "winning" situation, however, some games have different outcomes. An interesting class of exceptions is massive mulitplayer online games (MMOGs), such as *World of Warcraft*, where the concept of a single outcome is dissolved and replaced by different kinds of rewards, partly related to the player's social status within the online community. Such games have no clearly defined outcome, continue forever, and thus have high potential for addictive behavior.

3.3. Technical Elements

In addition to formal game elements, there is a variety of technical elements that define a computer game. Modern professional games address a wide spectrum of problems related to information technology and computer science, and the employed technology has a significant impact on the production cost. As a design principle, we postulate:

TFF: Technology follows function (gameplay)

A detailed discussion of all relevant technical elements for video games is beyond the scope of this paper, and we limit our considerations to a summary of the most relevant ones. We refer the interested reader to technical textbooks on game development, such as (Irish 2005, Rabin 2005, Novak 2007).

First and foremost, there is *software engineering*. Modern games demand a variety of sophisticated methods and algorithms, making the typical code base of professional game projects rather complex. The code complexity and the involved cost create the need for maximum reusability between different game projects. Such requirements are best addressed by advanced software engineering methods with proper class design, hierarchy, and the definition of interfaces. To the extent possible, standardized APIs and libraries are utilized. Microsoft's XNA (Miles 2008) is a good example of a low-level game API. Higher-level APIs are available for

physics, such as Nvidia's PhysX (Nvidia PhysX). The design principle underlying game software can be summarized as:

Simplicity: As simple as possible, as complex as required

A second important technical game element relates to 3D graphics. Modern 3D graphics requires complex geometry and appearance specification, including the scene structure (scene graph) and organization, import and export of 3D models, their positioning and transformation, advanced pixel shading and lighting, texture mapping specifics, animation, anitaliasing, and video replay.

The way the player *interfaces* with the game comprises another important factor. User interaction must be properly specified early on. Among the variety of controllers, we can chose between conventional ones, such as mouse, joystick, buttons, dials, keyboard, cameras, and microphone, or design customized controllers to leverage gameplay. Harmonix's *Guitar Hero* evidences the great benefit of a proprietary controller design. Special attention must be devoted to the control of viewing and cameras as well as the associated degrees of freedom. In general, camera control in computer games is a highly non-trivial problem and the subject of ongoing research (Oskam 2009).

The game's *artwork* and assets constitute the central elements that determine the game's appeal and aesthetics. As such, the creation of the artwork can easily consume a substantial portion of the production budget. 3D models comprise scenes, landscapes, characters, and objects. The proper choice of modeling software is highly important. In addition to 3D models, the design of texture maps makes up a relevant part of the artwork. Texture maps are utilized for a variety of different effects, including general texture maps, shaders, lighting effects, surface properties, and data for more general computations. Most often, the creative use of real-world digital photos extends the repository of synthetically generated textures.

Another salient element is *sound* and *audio*. Sound includes both general background or theme music as well as auditory feedback during gameplay. Sound contributes substantially to the feel of the game and draws upon the longstanding experience in cinematography of enhancing emotional response using sound elements. It is crucial to clarify the required sound elements early on, determine why they are needed, and how they contribute to the gameplay. The theme score of *Halo* exemplifies the sophistication of musical compositions as part of modern games.

Game *physics* is gaining increasing importance for the realistic simulation of rigid bodies, collisions, friction, explosions, crashes, deformations, cloth, and other effects. One of the challenges of game physics, as opposed to scientific computations or special effects, is realtime perfor-

mance in combination with unconditional stability and robustness. As such, most algorithms for game physics abstract from the actual physical laws and strive instead for visual plausibility. In addition, careful design of data structures is central for performance optimization. Since the design of these physical algorithms (Eberly 2003) is highly non-trivial, game development teams often resort to physics libraries, such as Nvidia's PhysX or Intel's Havoc.

Lastly, one of the most important technical game elements is the game's artificial intelligence (AI), or the game's ability to adapt to the player and to compute believable behavior of its characters. Game AI encompasses path planning, modeling of agent behavior, modeling of the player, global control of the game's state, and other processes. It is essential to achieve adaptivity in a video game. Most commonly employed algorithms include A* path planning, agent models, state machines, search and prune, and, more recently, statistical learning (Buckland 2008).

4. Understanding Fun

When asked why someone plays a video game, "fun" will be at the top of the list of reasons. Games are fun. That's why we play them. Thus, the concept of "fun" is crucial in game design, and understanding "fun" is critical in developing a successful game. While "fun" may seem like a nebulous concept, the theory of "Natural Funativity" (Falstein 2004) connects this concept to our evolutionary roots.

Natural Funativity suggests that games and other playful pastimes are thought to be fun and enjoyable not by chance but by the virtue of evolution. Nearly all games (video and otherwise) involve some form of challenge, such as running fast, throwing a ball or spear, or solving a puzzle. The reason most people describe these activities as fun is because, in our deep ancestral origins, those who happened, by the randomness of genetics, to enjoy such activities in their playtime were better prepared to handle life threatening situations that borrowed similar skills. Those who enjoyed running with their friends were more likely to outrun the tiger; those who enjoyed throwing balls as a pastime were able to survive as hunters; those who enjoyed solving puzzles were more likely to outsmart their enemies. Thus, the things we consider fun are not accidental, but rather were specifically favored in evolution as the pastimes that increased chances of survival. The theory extends beyond physical and mental challenges to social aspects of the human experience. Our desire to form groups, interact with colleagues, chat with friends, tell stories, and flirt with others are rooted in our tribal hunter-and-gatherer ancestors, since those who preferred human bonding over going-it-alone were more likely to survive.

With the theory of Natural Funativity in hand, the concept of “fun” in games is much less elusive. First-person shooters speak to our origins as hunters. Role-playing-games contain the element of exploration, which we consider fun since, in evolution, those who enjoyed exploring were more likely to find better shelter, better food, and other items necessary for survival. Games like Tetris and many other puzzle games help us hone our logic, perception, problem solving, and pattern recognition skills---all important in the history of our survival. On the social side, the cinematic nature of games appeals to our roots in storytelling. Games that focus on interacting with friends and flirting speak to our tribal roots in their purest form. Finally, it comes as no surprise that many of the most successful games combine several aspects of fun---physical, mental, and social---to resonate more strongly with our hard-wired concept of fun.

5. Conceptualization

Although the technical sophistication of modern video games is enormous, with hundreds of thousands of lines of source code comprising countless algorithms and mathematical computations, game *design*---which ultimately makes the player experience so rewarding---is as much an art as painting, writing, sculpting, or storytelling. In fact, game design draws heavily from all of these artistic endeavors, as well as many more.

As is the case with any highly creative process, there is no formula, recipe, or other step-by-step procedure to replicate the creativity necessary for game design. With no surefire approach, many people, especially beginners, find the prospect of designing a game from scratch extremely intimidating. Conceptualization, or, in short, the initial creative spark that leads to the core game idea, causes particular angst.

While creativity itself resists formal explication, the creative *process* can be formalized to help harness the natural creativity present in everybody. Here, we describe formalizations employed by both seasoned game designers and neophytes alike to aid in the conceptualization process when the core game idea is developed.

Write it down!

Although the first rule may sound mundane, it is extremely important: write things down! Ideas come all of the time---day, night, walking, running, driving, showering, eating, listening to music, etc. You may even wake up in the morning with an idea in your head from the dream you were having. If you do not write it down immediately, this ephemeral thought may disappear. Thus, the first formalization to assist in game

design is training yourself to write down each and every idea that you have.

Brainstorming

Brainstorming is a formal way to stimulate creativity and the free flow of ideas. There are a multitude of brainstorming techniques ranging from list creation, in which you list everything you can think of about a particular topic, to randomization, in which you select words at random from the dictionary or from a stack of pre-arranged cards and consider making a game related to the words, to research, in which you learn as much as possible about a particular topic, to mind mapping, where you expand ideas in a radial fashion to represent semantic connections between topics (Fullerton 2004). The goal and focus of brainstorming is always to generate lots of ideas. When working in a team, it is important to maintain a positive attitude and avoid criticism so that ideas flow unfiltered. Although brainstorming offers no guarantee (Gabler 2005), it has the side effect that new ideas may continue to come when unexpected. Have pencil and paper ready, and write them down!

Organization and Refinement

When a few brainstorming sessions are added to the normal flow of ideas, you may find that your list becomes unwieldy. At this stage, search for a way to organize and categorize your ideas. As you do so, look for patterns or structure that might hint at a profitable game direction. When going over your idea pool, select the most promising ones and expand upon them with a new, more directed brainstorming session. Restricting the potential landscape of ideas can often stimulate creativity and lead to clever or new approaches to a topic. As this process is iterated, continue to highlight and expand the best ideas while considering how they could map to the formal and technical elements of game design described in Section 3.

6. Prototyping

A game prototype represents the first working version of the formal system describing the game. At the prototyping stage, only a rough approximation of the artwork, sound, and features of the game need to be considered. This allows a designer to focus on the fundamental game mechanics, instead of worrying about production-related issues. The goals of prototyping are twofold: define the gameplay in its purest form, and learn whether the core mechanics hold the player's interest. In addition, a prototype can help to balance the rules and discover emergent behavior. The complex interactions of gameplay elements typically make it

difficult to uncover play patterns without a concrete realization in the form of a prototype.

As a first step, the core gameplay can be summarized by describing the game's core concept in one or two concise sentences that define the single action a player repeats most often while striving to achieve the game's goal. Over time, the meaning and consequences in the game might change, but the core gameplay will remain the same (Fullerton 2004).

Fig. 5: A paper prototype for a strategy game is shown on the left. Each unit is represented by a piece of cardboard. The two images to the right depict the game's units in the final software version.

A prototype can be realized in a variety of ways. One popular option is to create a physical prototype using paper, cardboard, and a set of counters since this works very well for testing game mechanics, rules, and procedures. Figure 5 shows an example, where a team from the game design class at ETH outlines how they balance the units of their strategy game using a paper version. However, it can be difficult to test action-oriented games in this fashion. A second prototyping option is to create a video-based animatic or a storyboard. Such a visual prototype can capture the user experience and communicate the game's ideas to others. One drawback of this form of prototyping is that videos can be difficult to produce. A third option is a software prototype. Tools for rapid software development, such as *Flash*/*Shockwave*, *Visual Basic*, or level editors of existing games, are well suited for this task. Figure 6 shows a game prototype created within only a few days using the *Director* software. It served as the basis for the creation of the game *Battle Balls*, and already contained the game's core mechanics that were later included in the console version.

Once the core game-play has been explored and refined on the prototype, the design process can focus on other areas of the game, such as extending the basic feature set, the game's controls, or its interface. At later stages of the development process, prototypes for specific elements of the game are created, again focusing on the underlying me-

chanics of the new element. Such prototypes can be used to demonstrate novel control schemes or new visual effects.

Fig. 6: On the left, an early software prototype of the game *Battle Balls* is shown. It is already fully playable, and contains all major game-play elements of the final version, shown on the right hand side.

7. Playtesting

Testing a game is itself a continuous, iterative process that overlaps all stages of game design. The goal of playtesting is to gain insights into how players experience the game, identify strengths and weaknesses of the design, and learn how to make the game more complete, balanced, and fun to play. Moreover, the results of a playtesting session can provide the necessary evidence to abandon unsuccessful parts of a game design. Since making significant changes to a game's design is more difficult at the later stages of the production cycle, it is important to start testing very early on in the design process.

Although a designer constantly tests his or her design as a game is created, the emotional attachment to one's own ideas makes it is crucial to have external testers review the game. Friends and family are good candidates to give feedback during early stages of the development. In later stages, unrelated testers can give more objective feedback and offer a fresh, unbiased viewpoint. Usually, after a selection process, the testers are invited to a testing session individually or in groups. During these sessions, it is vital to neutrally observe the testers. An explanation of the rules or the grand vision of the game might bias a tester's opinion and prevent him or her from giving useful feedback.

During the testing sessions, it is important to gather as much information as possible. The testers can be observed while they are playing the game, which might give insights about where players typically get stuck, where they become frustrated, or which obstacles and enemies are too

easy to overcome. After a testing session, a tester can be asked to answer questions about formal aspects of the game. Typical questions are whether the objective of the game was clear at all times, whether a winning strategy became apparent, and whether the tester found any loopholes in the rules. Finally, each testing session is followed by an analysis stage to determine which conclusions should be drawn from the feedback of the testers, and which parts of the game design might have to be changed as a consequence.

A very elaborate form of playtesting is performed by *Bungie Studios*, creators of the *Halo* franchise on the Xbox (halo3 2009). In Bungies's own research facility, testers are invited to play a current version of the game. While they are playing, a database captures all aspects of their performance: locations of deaths, weapons, vehicles, extras, the player's progress over time, as well as a full video of the playing session. For Halo 3, more than 3000 hours of game-play from about 600 testers were analyzed. This analysis made it possible to fully balance the multi-player game, and minimize the frustration during the single-player campaign. With this large amount of gathered data, state of the art data mining techniques can be used to extract information. One possibility is to create "heat-maps" that visualize which areas are occupied most often in multi-player matches, or adding color-coded time stamps to the map of a level. The latter can be used to quickly identify regions where a player performs an undesired amount of backtracking, and, thus, lost focus on the game's objective.

8. Case Studies

To illustrate the design principles outlined in the previous sections, we present and discuss three game projects, selected from the game design class at ETH Zurich, that exhibit different design goals and challenges. *Titor's Equilibrium* is a two-player game based on the physical interactions of objects in an abstract futuristic setting, *Sea-Blast* focuses on multiplayer submarine battles, and *Toon Dimension* features cooperative game play in a stylized comic world. Videos of all games developed during the class are available on the course website (gamelab 2009).

Titor's Equilibrium was developed by three students of the game class in 2007 and takes place in an abstract futuristic setting inspired by films such as *Tron*. The game is designed for two simultaneous players, where each player controls a fragile ghost that can survive only for a limited amount of time in mid-air. To survive for a longer time, and attack the opponent, each player can enter geometric objects that are spread throughout the game's levels. Depending on the type of object, the player has different abilities of attack and defense. Winning is achieved by destroying the other player's object and preventing him or her from en-

tering another one. The different attack styles are balanced in a rock-paper-scissors fashion.

Fig. 7: Concept art and several screenshots at different development states of the game project "Titor's Equilibrium." The painting on the left is an early sketch showing the game elements and prescribing the game's visual style. The two middle images show technology tests for rendering the game objects and smoke trails. The right image shows the first playable version with preliminary graphics.

The visual design of the game contains abstract geometric forms, and sets the different game elements apart by form and color, as shown in figure 7. This abstract style was deliberately chosen in response to the limited amount of time available for development during the class. Complex shading techniques give the game elements an interesting look. The computation of accurate physical interactions between the objects presented an additional difficulty during the realization of this game. The students chose to implement this part of the game engine themselves, which comprised a large portion of the overall development process. An example of a player interacting with the environment is shown in figure 8.

Fig. 8: A typical scene from the final game: the green player attacks the orange one in the foreground by dashing forward through several stacked boxes.

Sea-Blast was realized during the game class of 2008 (see figure 9). The game can be played by up to four players, either fighting against each other or against computer controlled enemies. The setting of the game is an underwater battle between submarines. The game's basic concept is very simple: players control a small vehicle in the level, and can shoot at their opponents with different types of weapons. This simple initial design permitted careful extension of the game's foundation with balanced game-play elements. For example, vehicles with different characteristics can be selected at the start of each playing session, allowing for different winning strategies. To set their game apart from similar games, the students chose to implement a physical simulation to calculate the mo-

tion of the water filling each level. This results in complex interactions between the water flow and the players and weapons. The physical simulation is a strategic element that can be used to distract or damage the opponents, for example by creating a local vortex that draws objects to its center.

The *Sea-Blast* game highlights the importance of an overall balanced game design. While the underlying concept makes sure the game is intuitive and fun to play, the technological aspects, such as the fluid simulation, set it apart from other games. Great artwork and sound effects round off the game's experience. The importance of all of these elements also means that a team working on such a game should consist of talented people that are able to produce all elements of the game with the desired quality. In this case, the *Sea-Blast* team achieved a very polished final version, and the game is publicly available for purchase via the Xbox online marketplace.

Fig. 9: Three screen shots from the Sea-Blast game: from left to right, the title screen, a scene from a four player battle, and two players fighting against computer-controlled enemies.

As the name suggests, cartoons inspire *Toon-Dimension's* (2009) setting and visuals. An evil scientist has split the world's dimensions, and two players in different dimensions must cooperate to win the game. While most of the environment can be seen and modified by all players, certain game-play elements belong only to a single player's dimension, and can be modified exclusively by this player. This setting allows for interesting puzzles in which the players must collaborate to combine pieces from different dimensions.

The game's graphics and animations follow a comic style. Cel-shaded three-dimensional models give the game a painted look, and cartoons inspire the exaggerated motions of players and enemies. In figure 10, several screen-shots of the game illustrate its unique visual style. In contrast to the previous two games, *Toon-Dimension*'s game-play is primarily based on the cooperation of the players. As a result, a great deal of effort was required to design the multiplayer puzzles. The team focused on creating a limited number of high-quality levels.

As evident from these three games, multiplayer gamplay is a popular choice in our game classes. In a multi-player game, the actions of other players comprise a significant portion of the game's mechanics, which simplifies development and makes such a design well suited for small teams with a limited time budget.

Fig.10: The game *Toon-Dimension* features comic-style visuals and a complex cooperative gameplay.

9. Conclusion

Perhaps more than any other discipline, game design is both an art and a science, a technical endeavor and a form of artistic expression. The most cutting-edge games stress the technical sophistication of modern console hardware and incorporate advanced algorithms for graphics, physics, interaction, artificial intelligence, and data structures while, at the same time, present never-before-seen visual styles, explore new forms of story telling, and invent new ways for humans to interact with computer and with their friends, families, and colleagues. The technology drives the design by providing ever increasing computational power and the ability to express more sophisticated visual styles. The design, in turn, drives the technology by pushing the limits of modern day hardware and software in ways that scientists could never conceive.

In this chapter, we have described the game design process and how it draws upon both technical and artistic elements. The earliest games - Spacewar, Pong, Space Invaders - seem exceedingly simple from today's standards. Indeed, both the technological and artistic sophistication of video games has increased with an unabated speed since the first game was made. The iterative process of modern game design stresses an early understanding of a game's core mechanics and visual style so that design elements can be refined and tuned. Ideas are generated, prototypes built, game assets created, and gameplay tested in a spiral loop that zeros in on a highly polished final product. Formal elements such as players, procedures, rules, objectives, resources, conflicts, and outcome are

developed alongside the visual style, user-interface, and thematic score. Brainstorming and other formalizations of conceptualization can greatly help with this design process. Prototypes are used to test the design and validate that it lives up to our inherent standards of fun. Playtesting completes the iterative process with a formal method to obtain feedback and improve upon a game's design. The final output is a collection of artwork, a collection of 3D models and animated characters, of algorithms, of procedures, of stories and music and sounds - all unified and coordinated by a single game design to deliver a wondrous experience that can only be found in the world of gaming.

10. Acknowledgements

The authors would like to thank all students and development teams participating in the ETH Game Programming Laboratory classes of 2007, 2008, and 2009. For *Titor's Equilibrium*: Marino Alge, Gioacchino Noris, Alessandro Rigazzi ; for *Sea Blast*: Urs Dönni, Martin Seiler, Julian Tschannen; for *Toon-Dimension*: Peter Bucher, Christian Schulz, Nico Ranieri.

References

Bates B. (2004): "Game Design", Course Technology PTR; 2 edition, ISBN-10: 1592004938

Buckland M. (2004):" Programming Game AI by Example", Wordware Publishing, ISBN: 978-1556220784

Eberly D. H. (2003): "Game Physics", Morgan Kaufmann

Falstein N. (2004): "Natural Funativity", Gamasutra, http://www.gamasutra.com/features/20041110/falstein_01.shtml (retrieved July 2009)

Fullerton T., Swain C., and S. Hoffman S. (2004): "Game Design Workshop: Designing, Prototyping, and Playtesting Games", CMP Books

Gabler K., Gray K., Kucic M., Shodhan S. (2005): "How to Prototype a Game in Under 7 Days:

Tips and Tricks from 4 Grad Students Who Made Over 50 Games in 1 Semester", Gamasutra,

http://www.gamasutra.com/features/20051026/gabler_01.shtml (retrieved July 2009)

gamelab (2009), Game Programming Laboratory, http://graphics.ethz.ch/teaching/gamelab09/home.php (retrieved July 2009)

Graetz J.M. (1981): "The Origin of Spacewar", in Creative Computing, http://www.wheels.org/spacewar/creative/SpacewarOrigin.html (retrieved July 2009)

Halo 3 (2009): "How Microsoft Labs Invented a New Science of Play",http://www.wired.com/gaming/virtualworlds/magazine/15-09/ff_halo, WIRED Magazine, Issue 15.09 (retrieved July 2009)

Irish D. (2005): “The Game Producer's Handbook”, Course Technology PTR; 1 edition, ISBN-10: 1592006175

Miles R. (2008): “Microsoft® XNA™ Game Studio 2.0: Learn Programming Now!”, Microsoft Press, ISBN-10: 0735625220

Novak J. (2007):” Game Development Essentials: An Introduction”, Delmar Cengage Learning; 2 edition, ISBN-10: 1418042080

NVIDIA PhysX, http://www.nvidia.com/object/physx_new.html (retrieved July 2009)

Oskam Th., Sumner R. W., Thuerey N., Gross M. (2009): “Visibility Transition Planning for Dynamic Camera Control”, Proceedings of the SIGGRAPH/Eurographics Symposium on Computer Animation

Rabin S. (Ed.) (2005): “Introduction to Game Development”, Charles River Media, ISBN-10: 1584503777

Sumner R. W., Thuerey N., Gross M. (2008): “The ETH Game Programming Laboratory: A Capstone for Computer Science and Visual Computing”, Game Development in Computer Science Education (GDCSE); ACM