

Index

[Foreword 2](#_Toc318640438)

[Introduction 3](#_Toc318640439)

[1. The Idea for a Game. 4](#_Toc318640440)

[2. Creating a Production Plan 8](#_Toc318640441)

[3. Division of work 13](#_Toc318640442)

[4. Designing a Game 19](#_Toc318640443)

[5. Programming 23](#_Toc318640444)

[6. Testing the Game 24](#_Toc318640445)

[7. Marketing and Publishing 27](#_Toc318640446)

[8. Overall Conclusion 28](#_Toc318640447)

[Afterword 28](#_Toc318640448)

[Sources 29](#_Toc318640449)

# Foreword

“**Computer programming** (often shortened to **programming** or **coding**) is the process of [designing](http://en.wikipedia.org/wiki/Software_design), writing, [testing](http://en.wikipedia.org/wiki/Software_testing), [debugging](http://en.wikipedia.org/wiki/Debugging), and maintaining the [source code](http://en.wikipedia.org/wiki/Source_code) of [computer programs](http://en.wikipedia.org/wiki/Computer_program).”  
-Wikipedia page on Computer Programming

The beauty of programming lays in the different steps within programming. It is one of my favorite things because I can be creative, and still produce something interactive which is interesting and fun for users. The concept of trial and error is used much in the world of programming. To test and test the code you wrote, and fixing any obstacle before moving on to writing the next block of code. This is what truly interests me in programming; a problem will come up, and I have to solve it. When solved another problem will come up and require solving as well. This will continue until you have a product free of any problems, ready to be shown to out standers, which in their turn will encounter many new problem which you had overseen. A never-ending line of problems, all ready to be fixed.

Why I chose the subject of programming for my final paper, is because I always had a particular interest in instead of using something interactive, creating something interactive and shaping it to my will. I began when I was 14 years old and found out that you could connect a Wii-Remote with your PC through a Bluetooth connection and that you were able to write scripts that uses the input of this Wii-Remote. After learning the basics of programming like variables and ‘if-statements’ I went up a level to creating interactive flash-games. After trying this I finally found the perfect way to live up to my belongings with the XNA-Framework by Microsoft. This gave people the opportunity to create full-scale games for Xbox360 and PC’s.

Today I have been programming games for PC and Xbox for about two years, and I have gained much experience. The thing is that I never took the time to create a game of such a size that it would be able to interest people for a longer period. Now when having to choose a subject for my paper I found the process of creating a game a perfect option. I can create something fun for me and others, something to be proud of, and having fun while doing work for school which requires much time. The level of mathematics and physics used in programming makes this a suitable subject in my opinion.

The thing I want to get to know out of this paper is; what is the best way to make and publish a game? When I tried creating a game on a larger scale together with a friend of mine, I had to plan and think carefully about how to attack the problems and create the things I want. How do the professionals work on this, do they use humongous whiteboards in small offices with all kinds of scribbling on it, or do they just note down the ideas they come up with so they can use them later if they like to? I have a little experience creating a game on larger scale, but I want to expand my horizon to develop myself further into the world of programming.

In the big break this year I had some free time on my hands, and on rainy days I sat behind my computer and tried to work out some things I wanted to get done. Eventually I came up with a game where 2 players have to compete with each other to drive as fast as they can into the car of the other player. The concept sounds simple but the programming behind it took me 2 weeks. I have written an algorithm based on the Riemann-Som to check for collisions for example. This is what I like about programming, having a difficult problem, and solving it all by myself so the game works like a charm. I want to re-create this game from scrap, and bringing it up to a level where it is suitable for publishing on such things as Xbox Marketplace ready for sale.

Let’s start!

# Introduction

What topic we will cover in this paper is games, or particularity, the development of games. Many people play games on a regular basis, but to make these games playable, they must go through a difficult and long process. It depends on what you want to achieve by the game, but in general even the smaller games take months or even years with enormous groups of people. The developers don’t just improvise, but entire plans are made on how to tackle the creation of games. I will describe the process of a game from idea to publishing, as detailed as possible. I will develop a game by myself parallel with this paper, and will compare every step a professional developer takes, with the steps I take.

So how do they come up with an idea? Do they go to the gym or play football, and get ideas at random? Or do they force ideas by brainstorming with teams? This is what I will cover in Chapter 1: The Idea for a Game. But after this process, the developer only has the idea. Now the plan needs to be plotted, because the people working on the game need to know what they have to do. What resources do they use for this and what else comes with creating a production plan? This topic will be handled in Chapter 2: Creating a Production Plan.

The developer almost always consists of a group of people. But who does what and who takes charge? Is the work only divided to simple groups, or How does he actually design the game? Does one person do this, or do they hire professionals to design the games? This will be answered in Chapter 3: Division of Work. But now that the teams are in place, and the plan is in place, what needs to happen next? Well, the game has to be designed, visually as well as thematically. Story line and interface come in to play here as well. So how do they design a game? Chapter 4: Designing a Game will handle this subject.

Now we have designed the game, and the game needs to be made into something playable on a console or computer. Here the programming comes into play. But how do they program? Do they just go try something, and see what they will end up with, or will they create diagrams and big plans? Are they in favor of trial and error? This will be handled in Chapter 5: Programming the Game. The game of course has to be tested while programming, but after the development as well! So how do they hire people, and what do they test? When is the developer satisfied with the results, and are there general things to test? This will be discussed in Chapter 6: Testing the Game.

Last but not least comes the marketing and publishing. How do they reach the players, and what tactics do they have for this? Does the developer fully rely on friends telling friends how cool a game is, or do they have entire teams promoting the game? See Chapter 7: Marketing and Publishing for this!

Go ahead and read through the paper. I kept track of how I created my own game, which I described fully in ***Attachment 1***. Through the rest of this paper I describe briefly how I did a step, and make a conclusion.

# 1. The Idea for a Game.

How do professional developers get an idea?

First, the development company needs to get a main idea. Companies often set up teams of marketing professionals to look for holes in a market, and see what the people are interested in at the moment[[1]](#footnote-1)[[2]](#footnote-2). The shooters developed between the year 2000 and 2005 were mainly WWII games, but now there is a period of Modern Warfare games. By this I do not mean the Call of Duty series, but actual modern warfare as a setting. These teams create multiple ideas, and try to choose what might be the best option. Another thing they can do is provide the developer with multiple options, and let the developer choose.

The teams set to the task of thinking of ideas is not very simple. There are different things to consider when creating a game. Such things are time-intervals, genre, platform, goals and market. Or even more in-depth: story line, graphics, characters[[3]](#footnote-3).

Time-intervals is the way a player interacts with other players, non-players(computer controlled characters) or items. There are 3 different types of time-intervals. At first you have a turn-based game, most common in RPG’s. This is when the player is given an amount of time to make a decision while the other player has to wait it’s turn. After one player has chosen what to do and executed this, it is the other players turn to do something. Another time-interval is Real-time. There are no turns in this time-interval. Both players have to react as fast as possible because the other player is busy as well with planning and executing. This is most common in action-games. There also is a time interval type which is limited-turn based. It has the idea of a turn based interval, but the time given for choosing an action is limited. This way there still is pressure behind the player to think fast. This last time interval is actually a mixture of the first 2 intervals.

Genre sounds important, but is not as important as it sounds. Some genres are Puzzle, Serious games(solely educational), Action, Adventure and RPG.

*“I tend to de-emphasize genre in my designing and thinking. I feel that genre is a bit of a double-edged sword for designers. On one hand, genres give designers and publishers a common language for describing styles of play, they for a shorthand for understanding what market a game is intended for, what platform the game will be best suited to, and who should be developing a particular title. On the other hand genres tend to restrict the creative process and lead designers toward tried-and-true gameplay solutions. I encourage students to consider genre when thinking about their games from a business perspective, but not to allow it to stifle their imagination during the design process”*

*-Tracy Fullerton (Associate Professor, USC School of Cinematic Arts; Director, Game Innovation Lab)*

Game Development Essentials, 3rd edition, page 68

The platform is narrowly connected with genre. Would you rather play an 3D adventure game on your phone, or a simple puzzle game? When thinking about consoles, the action game genre is a better choice.

The last main thing to keep in mind is the market you will be developing the game for. Do you want children, young adults, teens or elderly people to play your games? Different age groups play on different platforms. Handhelds such as the game-boys are more for children and young teens, while consoles are more for late teens and young adults. Another thing connected to your market is again genre. Puzzle games are played by another market than FPS games (First-Person shooters).

Last but not least; the goal of your game. Normally when we are discussing games, the games are purely created for entertainment. But you have “Serious Games” as well. These games are created to educate people on certain skills, think of surgery or flying. A well-known type of serious game is a simulator, like Flight Simulator and America’s Army.

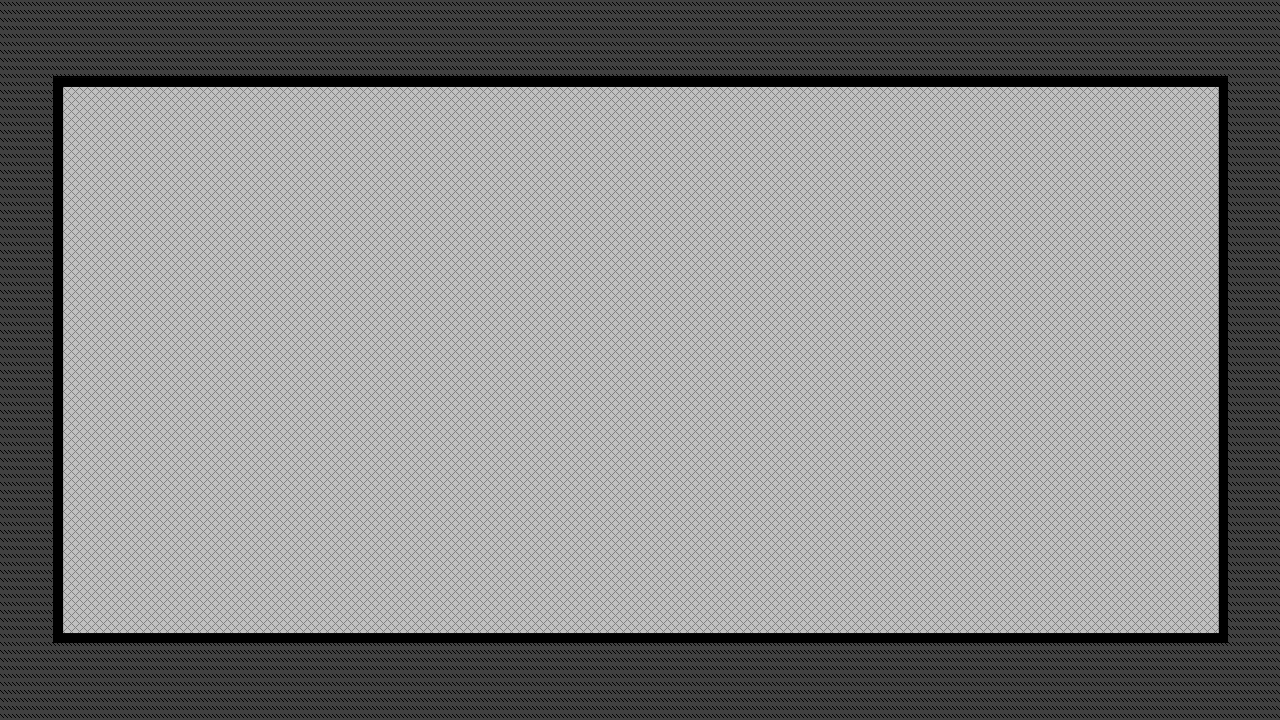
You can also go more in-depth such as story line, graphics and characters. Do you want the game to be focussed on the gameplay, such as is the case in modern online multiplayer games, where it is just about winning without any story, or do you want the game to be all around the story. Giving the player a feeling of watching an interactive movie. What will make your game more attractive? As well as the graphics. You have games of cool 3D graphics, but there are also puzzle games in a 2D theme of for example chalk board. Or more in the style of a cartoon, with thick black outlines of the sprite? Characters are important as well. In Real Time Strategy games you often control groups of soldiers for example. You will not make a personal connection to in-game characters. But in games with a main character and a good story line, the player can make a personal connection with the character, even though this can make the character entirely fictional. This is what professional game development companies keep in mind when creating a basic picture of the game.

How did I get the idea?

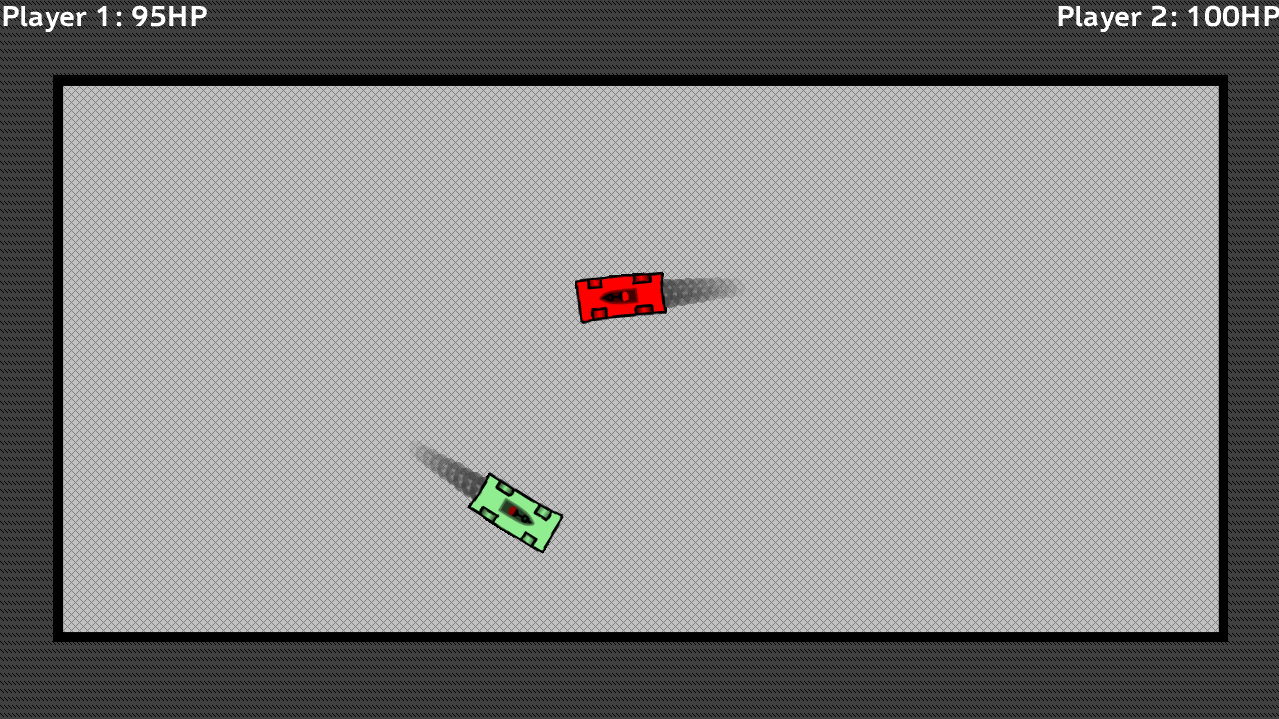
When creating a game, there first has to be an idea. For me, this was easy. Before I started this paper on the process of developing a game, I already made a game. As I said in my introduction, I want to redevelop the game to a greater scale, making it more interesting. The original game was created by accident. I tried to develop a game-engine for Physics. I failed, but something else was born. I made an algorithm in such a way, that the algorithm itself would create the movement speed on the x-axis and the y-axis with only given the direction(in degrees) and speed. I could change the direction whenever I liked, and so I created a sort of driving movement. After racing around a bit on an empty screen, I started to think about making some sort of arena. After some useless drawings I created an arena which looked metallic (see figure 2).

C:\Users\Beer\Documents\Visual Studio 2010\Projects\DrivingCar\DrivingCar\DrivingCarContent\Car.pngThe lighter part is where the car should drive in. The darker part I used for decoration and a place to keep track of the scores. After this I created a car.

*Figure 1 – The Original Car Image*

  
*Figure 2 – The Original Arena*

By using the framework I programmed in, I could shade the car with any colour I liked. This way I could use the same car over and over but still make a difference between them using different colours. I now wanted to create text (a string in programmer language) to show the score of each player in the screen. I also spawned 2 cars which could drive with a trail of smoke. This together made a simple-looking game altogether (See figure 3). After this I had to create a collision test-algorithm, discussed later in this paper, because the framework did not come with one used for rotated sprites(images).

I do not have any tactics to try and think of a game. I just watch a movie or am in school for example and suddenly something comes to mind. I remember it and as soon I can reach my PC I immediately start programming.

*Figure 3 – The Game In Progress*

Conclusion

There is a very clear difference between the way a professional developer handles this subject and how I do this. The developer does this very organized and with commercial purposes in mind. I myself get more the idea on impulse. The professional developer uses multiple parameters to come to a decision as I said before, keeping in mind its market. I make a game of what I would like to see, and not to make as much money as possible. The reason for this main difference I think is that I have no budget, and the professional developer does, and I do not want to create this game to generate as much income as possible, but to entertain myself and others.

# 2. Creating a Production Plan

How do professional developers create a production plan?

As I understand it there are many different phases in the cycle of the development of a game. You have the concept, than the pre-production process, than a prototype which will be a demo for selling the game to a publisher(discussed later in this paper. Than the actual production will start, alpha testing, beta testing, the gold phase, which is the manufacturing of the game, and at last the post-production process[[4]](#footnote-4)[[5]](#footnote-5). What I will cover in this chapter is how the developer decides what to do, in the overall development of a game.

Of course first the concept has to be envisioned, which I have discussed in the previous chapter. This is often a short step, and done with a limited amount of people. The next step is pre-processing, where the actual plan of approach comes to the table. Many things need to be planned ahead to speed up the process and be able to present the final project with only a prototype. The developer creates a document named “game design document (GGD)”[[6]](#footnote-6) which will be the production plan. This plan will change very often, since the development of games is dynamic. The first thing to cover in this document should be the game interface, what elements do you want to include in the interface, how much time is required to produce the interface, what are the costs and needs to produce the interface? The next thing to include is the Game World, including the information and elements on a level. Everything including art, gameplay, animation, background, characters and cinematics should be included in this part of the document.

If there are characters in this game, there should be planning for these as well. What weapons does the character have, what abilities does the character have, what can the player find in the game? One of the most important things included in this plan is the game-engine, which is not limited to the programming! It considers number of characters that can be present on screen at once, or number of animation per character. What number of colors can be used per texture map? Is there support for special controllers?

For further in-depth analysis on creating a plan you might want to read ***Attachment 2***, which is an example table of contents that came with my book “Game Development Essentials 3”.

Another very important choice to make not included in the GDD, is what tools and software to use when designing and programming a game.

How did I create a production plan?

So now I have to think about what I like to have in my game. There are some elements I like to see in my game. I wrote them down as bullet points:

* Time-Interval
  + Real-Time only
* Genre
  + Action
* Platform
  + Xbox 360
  + Windows-PC
* Goal
  + Entertainment
* Gameplay
  + The game is based on the multiplayer modus, arena style.
    - 4 Players Max
    - NPC’s Available (Non-Player Characters/Artificial Intelligence)
  + The players car is fully customizable.
  + The players car is bound to the players gamertag(Xbox Only)
  + The players car is bound to the players custom save-game (PC Only)
  + There is a currency system, to buy upgrades and customization items.
  + Upgrades:
    - Defensive Upgrades
    - Offensive Upgrades (Both range and direct)
    - Movement Upgrades
    - Looks Upgrades
  + The game has a levelling system, to unlock new arena’s

So I like to create this game as a Multiplayer-Game where you can battle up to 3 other players or computerized players. To create a feeling of achievement and to be able to set personal goals, thus making the game addictive, I will add a currency system where you can earn money for example based on amount of damage dealt, damage done to you and time. An upgrade system will be placed in the game as well, where the money gained can be used. The game will be in Real-Time interval, where both players play simultaneously in an attempt to defeat their contestant. To make the game more personal I like to give each player his own car, which he can bind with and what will create a feeling of ownership.

Programming & Graphics tools

So I got my Ideas in place, but now I have to actually create a game. I have to use a programme to program my game in, test my game in, and design my game in(Graphically). As I said before I already have experience in creating games, both programming them and designing them.

I always used, and will use, the windows-language C#. There are different levels of programming levels, first you have ASSEMBLY, which is the language used to talk directly to the hardware of the system. After that you have the 2nd programming language level such as C, a language created to make it easier for humans to program in. I use a 3rd level of language named C#. You can program with this language in Notepad, a standard tool in Windows, but in this you can only type, not test or look up what mistakes you have made. To make this job easier to do, Windows has developed a program to develop programmes and debug them. This program is named “*Microsoft Visual C# 2010 Express”*. This is a free tool released by Windows. *[[7]](#footnote-7)*

But with this tool it is still hard do develop game effectively, since there is no graphical engine, and no methods (groups of code) to draw graphics to the screen. Another team of Windows has created the “*XNA Framework*”, with their most recent version XNA 4.0.[[8]](#footnote-8) This gives multiple classes to draw basic things to the screen and connect online and with storage devices. This is very helpful, but very basic as well. There is no code to check collisions in an effective way for example, which I will develop myself later. This framework is used for graphical and storage use only, the actual ‘Game’ still has to be developed fully by the user.

To draw things to the players screen, the game needs graphical images first, which in turn need to be created first. The program I use for this is “*Paint.NET*”, a free program developed by a small company[[9]](#footnote-9). This tool is similar to the Windows program Paint, but with much more extra’s. I use a simple program like this since I can use it and create what I need. Because I myself am not a very creative person with skill to draw, I don’t need any sophisticated programmes like Photoshop.

Creating Graphics

First I’d like to complete most of the graphics first, and further graphics (such as offensive/defensive items and animations) later. I have to create multiple arena’s for different levels and I want about 10 arenas in my game. I have to draw a basic car as well which is coloured white, so I can overlay it with colours later. There also has to be a menu, and that is what I will create primary to the main game. The rest I will developed mixed with the process of programming.

Programming Steps

In this subsection I will globally sum up the steps I want to take to achieve certain goals. This section will not include every step, because I know that when I am busy programming, I will bump into things I had not yet foreseen.

Step 0:

I have to import my limited code already generated for previous projects.. This will save much time and nothing very important is included. I will include my “Sprite” class only, which is a group of code that I myself wrote that handles graphics, so that the graphics file is immediately ready to be drawn to the screen.

Step 1:

I have to program the screen-engine. This engine must only update and draw what is needed in the window of the player at a particular moment. I need different states in the form of: Main Menu, Load Menu, Save Menu, Playing, Playing.Pause(State in a state), Options Menu. This way I can assign different buttons and images to different screen, and the Screen manager can make the program only draw what is necessary. This way I will spare CPU. This Manager will become clearer later on.

Step 2:

I have to create the menu’s and buttons. No engine for playing is required yet. I want to make the menu to a state where it is almost finished, so I don’t have to edit this anymore. The button’s won’t get any actions assigned yet, except for the ones carrying the player to a different screen. The Screen Manager is already ready, so I can control this.

Step 3:

In this step I will make a controllable car. This car can rotate to the angle the driver is controlling it. I have to make the driving look natural. I will create a trial of smoke as well, just to make it look nice. Another graphical addition the car will have is a trail when it brakes.

Step 4:

I have to make the car customizable, through colouring. I will create an upgrade menu as well, but I will think later of the upgrades that it should contain, so I will leave this screen unfinished. The car class will need room for later code for upgrades as well.

Step 5:

I will make the game compatible for up to 4 players simultaneously, where when pressing the play button, each player should sign in manually with different controllers for setting the different players. There should also be an option to add NPC’s, in different difficulties. The AI department will not be created yet, first I want to get the game finished for real players only, this is due to the difficulty of reproducing human reactions. I should also create code where 4 players can drive in the arena at the same time.

Step 6:

I have to write collision code for 4 cars at the same time. This will proof a difficult challenge, mathematically and in the aspect of bugs. This will also include giving momentum to a car that gets hit, without changing the direction of the nose.

Step 7:

Here I will create code to determine the damage by the player that drove into another player. This depends on the hardness of the car’s outside and angle for example, by both of the players. I will have to create an algorithm which keeps in mind all the variables this depends on.

Step 8:

I have to create any option of respawning and repair I will put into the game, determined by global standards and personal upgrades.

Step 9:

Now it is time for the code of offensive items, with the attacks further in the development decided. Damage has to be set for each particular item, and effect per item. The upgrades have to be decided upon as well, which will be imported into the game.

Step 10:

Code for a finished game has to be created, as well a screen (popup?) for when each player earns his money. After this players will be asked for a rematch or a return to main menu. This will close the circle of never-ending gaming experience.

Step 11:

I will have to create the AI engine for computer-controlled cars, for this I have to do some personal research, but the book I am reading at the moment covers this subject. Difficulty levels have to be imported as well, but I think this is a matter of precision of the ‘virtual’ player by just one variable in percentages (such as: easy 0.3, medium 0.5, hard 0.7, near impossible 0.9).’

Step 12 & 13:

Create codes for saving personal statistics and scores and finalizing the product by removing bugs

Step 14:

After play testing remove the bugs found by the testers.

Step 15:

Publish

Conclusion

While the plans of my game look similar to the way a professional approaches it, the importance of my plan is far less than of a professional. This is because I am a single developer trying to develop a more simple game, while a professional development company consists of multiple people. All of these people developing the game need to be updated and fully informed about what to do next and what to involve in the game. Because I am alone it is handy to write out my plans, but they do not have to be exact, while this should be the case with a developer.

The GDD[[10]](#footnote-10) is very important in a games process, and people claim this is a more difficult part than the actual designing of a game. The GDD needs to be on a server which can be accessed by all employees so they can extract information at every moment necessary.

# 3. Division of work

When a game is developed much work has to be done. A developer hires professionals which are experts in a certain area, to develop only a particular part. So how is the work divided? What is the main structure of a development team? What are the roles people play and what job do they have?

Producers[[11]](#footnote-11)

So what you could name the “boss” of a developer is the executive producer. Often this person is not working for the developer, but for the publisher. He or she decides what projects (so an entire game) to fund and how much funding is granted for this project. This person needs to manage every project in the means of making sure all goes well and that the producers of a single projects do their job right.

Under the executive producer are the producers. These are the managers of the development of a single game. They scout for talent and contract new people such as artists, composers and programmers. They make sure the entire process stays on track and on schedule, and tries to have everything working smoothly. The producer also communicates with the partners of a development process, such as the publisher, the hardware manufacturer or the licensor.

The next producer is the associate producer. This producer is directly involved with the development teams, making sure everything runs smoothly. This person also captures screenshots for the public relations team and works with outsourced producers focussing on cinematics for example.

Under the associate producer works the assistant producer, someone doing the paper and administrative work on the budgeting and scheduling. Sometimes the associate producer is the same person as the assistant producer.

Design[[12]](#footnote-12)

First you have the creative director, this is the person that makes sure the games content is consistent with the original plans of the project. When a certain art style has been chosen this person works together with the art director to make sure this art style is constantly used.

The design director focusses more on staff and documentation, and focusses less on the design tasks itself. This role is often taken up by the creative director, the lead designer or a producer.

The lead designer is involved with the daily game design process like level design and gameplay development.

*“As lead designer, the actual duties vary on a day-to-day basis. Overall, I’m responsible for keeping the vision for the game, the game mechanics, and the “fun” of the game; the overall story (and any specific elements about the game designed to propel the overall story, such as companions, key locations, etc.); and then breaking down the remaining elements into digestible chunks for the other designers in terms of area briefs and area overviews (“This planet is X, the following things need to happen on it,” etc.), breaking up the mechanics and play-balancing (“I need you to oversee the feat and class advancement systems as long as they accomplish the following goals,” etc.)- and then managing all the parts so programmers, artists, and the producer are getting everything they need to keep moving.”*

*-Chris Avellon, (Creative Director, Lead Designer & Co-Owner, Obsidian Entertainment)*

Game Development Essentials 3rd Edition; Page 324

Of course a story line has to be designed as well, and this is done by the narrative designer. This are often people who have prior experience in story writing, but are able to adapt to the game industry, because in game the story is dynamic, which is not the case in movies. These people often write all the dialog and character development as well.

The interface of the game needs to be designed as well, and this is done by the interface designer. He determines the layout and gets help by the art team, but they help in a later stage. This person needs to make the interface in such a way that all information is included, but the interface needs to be simplistic and understandable.

Another designer is the level designer, who focuses on building the game environment. Some of the designers only work on the physical world, others work on implementing gameplay on the world as well, such as useable objects. These people are often involved in writing the stories associated with the levels they design.

Art[[13]](#footnote-13)

On the top is the art director, who does almost the same job of the creative director. He or she makes sure the art in a game is in the style of how the game was planned. The game has a certain look, mood or feel and the art should create this mood. The director also schedules, hires new artists and does the budget cuts.

Then you have the lead artists, who are the supervisors of a certain department which handles a part of the development process, such as texture, modelling, animation and storyboard artist. Every department often has its own lead artist.

The concept artist is a person that creates the sketches of models, characters and worlds of a game. This person can create a visual concept of a game before it is developed, to sell the idea to a publisher. For example this is the concept artist of Bethesda Games with some of his work. He recently died of cancer but created the concept of huge releases such as Fallout and Skyrim. See Figure 4.

Another important person is the modeller. This person often converts the 2D concepts to 3D models. These 3D models are created from scratch, or are scanned in from real clay models. The modeller often provides shading texturing and lighting of the model as well.

The texture artist is the person who generates 2D maps to apply to 3D *Figure 4 – Adam Adamowicz* models. The same goes here, the artist can scan a texture of anything and edit it to fit neatly into the game, or he or she can build a texture from scratch.

The animator is the person who animates the models. This is often done with motion capture, where actors have suits on them that register their movement, and they implement these captures straight onto the model, or they can keyframe an animation. Take a model and creating a animation by repositioning the model frame by frame. There are different type of animator, those who do the character animation, those who do the cinematics animations and the animation of effects (such as explosions).

The technical artist is the person who understands the work of 3D models and game engine. This person is often the bridge between the artists and the programmers. The technical artist makes sure everything is stored in the right format so the programmers can work with the art to programme and debug a game.

Programmers[[14]](#footnote-14)

The technical director is someone who oversees the implementation of the plans into the games. This person also selects what code to use, what tools to use and what hardware to use.

Then you have the lead programmer; the person who keeps everything on schedule and reports to the technical director. This person works on a daily basis with the programmers and manages them as well.

*“If I needed to implement a task (from the design team) such as “two chracters are playing golf through the online environment,” then the tasks to be done would include:*

* *3D max plug-in to export character model*
* *Enhance rendering engine*
* *Play scripting*
* *Network sync design*
* *Network protocol design*

*I would make these separate tasks, assign them to the programmers, and check to see how much of each task has been completed on a daily basis.”*

*-KyungMin Bang (General Manager, Electronic Arts – Seoul)*

The next programmer would be the network programmer. This person often works with the online-multiplayer aspect and data management of online databases for things like leader boards and such. Sometimes these programmes need to have knowledge of other type of networking as well such as Bluetooth or Wi-Fi.

The graphics programmer is a mix between a programmer and an artist. This person must be able to work and program animations and work with tools such as Direct3D, the 3D graphics part of DirectX.

The engine programmer, which I am the most interested in, is the person who does the core programming. From rending graphics to the collision detection of in-game objects, which can often be a difficult challenge. *Figure 5 – The logo of the Unreal Engine*  
  
An engine is often used for multiple games. Games such as Modern Warfare 3 are created on the same engine of Modern Warfare 2. Of course this is a logic case, but another licensed engine is the Unreal Engine from Unreal Tournament[[15]](#footnote-15), which is used for all kinds of games.

The tools programmer designs and develops tools or software for the other developers or artists to use. This can be from plug-ins for modelling to entire drawing software.

A very interesting programming field is the AI field, the Artificial Intelligence. This field has its own programmer named the artificial intelligence programmer. This person programmes the ‘fake’ intelligence of NPC’s(non-player characters) in a game who should react logically to input from the player, such as an attack, a command or even speech. AI is an entire study on its own, where the student must understand the working of a brain and the psychology of a human.

The audio programmer works with the sound of a game and implements it into the game. This person should be able to work with the sound card and even program music.

The physics programmer, of course, handles the physics of a game. This ranges from gravity to the light breaking in water. The programmer needs to understand physics completely as well as be able to program. Often physics engines are created as well, such as Physx by Nvidea[[16]](#footnote-16).

An interface programmer is someone who creates interfaces which are logical and usable. This person also works with interfaces such as if the character is actually touching the floor of the world.

The associate programmer is a junior programmer; a person who does small adjustments in the game often trough software developed by the tool programmer. This person works together with other designers or programmers to make certain events happen.

Audio[[17]](#footnote-17)

As all the other sections the audio section has a director as well; the audio director. This person hires new voice overs, composers or licenses songs to use in the game. He also exchanges information with the audio programmer just like the art director to make sure everything is in the right format so the programmers can work with the audio in an instance.

Composers are people that compose scores or entire songs for a game, to fit the mood of the game, or the current situation the player is in at a moment. So country music in a town, or action music in a fight. A well-known composer is Hans Zimmer, who composed the music for the modern warfare series, inception (movie) and other many, many famous entertainment material. This person is often outsourced, because developers do not have their own composers.

A sound designer is someone who records or gets sounds which are used for effects and ambient sounds, such as chirping birds or a barking dog. These people go out in the world to record all kinds of sound, or use sound which have a creative commons license (which means the material is free to use for no costs) which can be found on the internet.

An voiceover actor is someone who is used as a voice for a character. Often multiple people are used, or the same audio fragment is manipulated to sound like different people. These people are often outsourced as well.

Testing & Quality Assurance[[18]](#footnote-18)

The testing manager is someone who manages the process of testing trough budget and time, and who manages the QA. This person also schedules testing sessions.

The lead tester is the supervisor of the testing team, and is responsible for finding out what the exact problem is when a problem occurs. He must also determine if a bug is important enough to be fixed, and if the testers are searching in the right areas of the product for any bugs.

The Compatibility or Format testers are the people who test the game to see if it works well on all the platforms or on all PC’s. Format testers check if the game plays equally on the i.e. the Xbox and the Playstation, even though they have different hardware and controllers. Compatibility testers work on the PC version of the game, if the game works with all the hardware. By coincidence I was at the informatics study today at the University of Amsterdam, and someone had a weird error with his programme on his PC. When he executed the exact same programme on a PC from the university no problems occurred. This is because different hardware have different drivers that work differently. The compatibility testers check if this all works out.

Playability, Usability and Beta testers are people who are volunteers at home, to test the game for these properties in the beta phase. Often there is a closed beta where these testers are family members of the developers. These people give feedback on the game, which the developers use to smoothen out the problems.

Focus testers are people used to test what parts of the game are most appealing, used for marketing purposes. This is often a group of people which are the targeted costumers.

Remember that most function I talked about can be occupied by multiple person. So there could be 10 Engine programmers or just 1. It can also be that the same person is handling 3 positions at the same time. All in all a development team consists of some people, often not more than 30. Sometimes there can be 100 persons involved, but these are rare cases. The positioning of people is very structured to be as efficient as possible, and not have to many tasks on one person.

How I Divided the Tasks

Because I am a single developer I have no team or producers, nor do I outsource anyone to do any work for me. I am not interested in this commercially on a larger scale, so I do not want to invest in this. I improvised mostly by myself and of course did all the tasks by myself, from programming to audio to art. The only thing I used other people for was the testing process. For this process see ***Attachment 1*** or “How I Tested my Game” in Chapter 5; “Testing the Game”.

Conclusion

The difference is huge between how I handled the work and how a developer handles the work, but this is logical. A professional development team has to work on an enormous project with a deadline, while I can take the time I need on developing games. I do not need super-special tasks to be done I can mostly do them myself. I have once worked together with an artist on a game, but out of experience I prefer to do this work by myself as well.

# 4. Designing a Game

Story Line

The story line of the game is an important aspect. But why is this important, and what is needed to make it as good as possible? How do people design a story line?

So how people often think of a story, is brainstorming with teams or alone. The story line is an important part of the game, since the game needs to be appealing to the player. I will take the game “Half Life 2”-series as an example. This is a game with not very high graphics, but a well-executed story line, which really lets the player connect with characters in game, such as in a good movie. When a team is writing a new story, it often should contain new items, to make the game original, but still put in something familiar. When the writers or developers have some idea’s, they are going to structure the ideas into characters, events and a time-line.

Because at this stage they often only have a rough idea of what the game should be about, they now need to work out this idea. The writers or the narrative designer now goes to work. He will write out exactly what should happen and in what order. Story lines can be in chronical order such as Half-Life 2, or in Non-Chronical order such as the Assassins Creed series where you switch between present and past. The narrative designer should write everything from events to precise dialogs, see the precious chapter “Division of Work” and the sub-chapter “Design” for information on the narrative designer.

After this is done, the people from the art department, often the technical artist, should create a story board. For this he uses the concept artists to create the 2D pictures and put them on a board. With this story board the programmers can work after the artists have created some graphics of the game.

Graphics

Of course graphics are important as well, they set the mood and the feeling of the game. A scary game is often darker, while a family game is bright and coloured. So how the artists design the graphics?

First off you have the characters. First the concept artist has to work on the first drawings of a character. The artist must keep in mind that the character must look similar to a person in his role. An evil with should not look like a good-hearted clown. The concept artist first develops concept art from multiple aspects of the character, such as multiple side and from a different angle of height. The features such as properties belonging to a similar character should be included in these drawings as well.

Now comes the modelling of the character, often done in software such as 3Ds Max and Maya. Artists start placing polygons, which together make up a surface. You can place the polygons in such shapes that the entire surface will form a model of for example a car or person. This often takes up lots of work since many polygons need to be placed often by hand.

Now comes the texturing; drawing a “skin” for the character. The models often include the clothing already, except for games where the player can switch clothing. The texturing artist which can be read about in the previouse chapter creates images known as “texture maps”, which are 2D images which form the skin of the 3D model. The texture artist should follow the initial concept drawings, because those are formed with the mood and feel of the game in the back of the artists mind.

Last but not least comes the animation, where the character is made to move. Real people are used to record the movement of actions they do, such as running or swinging a sword. This is converted to animations which can be applied to the model. Another method is keyframing. You can find more information on this in the previous chapter at the animation artist.

The level design is very similar. First there is some concept art, enough to have a view of all the walls or if in open air, all that must be seen. Often a map is included as well, and also a height map. A modeller models this into a 3D model, which is larger than a player. The model is then texture, and animated if necessary (i.e. waterfalls, volcano’s). These animations are all done by key framing. Of course these levels should as well have the same feel or mood as the entire game should set.

*Gameplay*

Gameplay is not just; “Is the game fun or not?” but the entire dynamics, challenges and play rules throughout the game. When is the game over? How do the players interact with each other? Is the game balanced?

Games mostly end with a player winning or losing, but sometimes, a game does not end at all. There are often different conditions or requirements a player should meet to win. Sometimes only one goal has to be met if the player wants to win. When the game mostly revolves around a story line, the story line has to be finished, and the player has won. For example the game Half-Life only wants the player to move on into the game, and is more in the direction of a interactive movie, while the game “Skyrim” is a RPG, which is more in the direction of a “Real-Life Simulation” where the story line and the events are influenced by what the player chooses to do in the game. For example: In Half-Life the bad guy has to be killed, and the game is over. In Skyrim, you can work and help the citizens of one city, or help another city conquer the first city, which influences the story line in a massive scale. The game can also not end at all, such as with Tetris. This game gets increasingly and increasingly difficult until the player has lost.

There are different ways to lose, implicit or explicit. When you use due to a implicit condition is when a player for example wins, but is not the first to win (such as races). Explicit is when the player for example dies before achieving the set goal, which is the case in most games.

In the gaming industry you had 2 interactivity modes in the past, and recently a new one was added. The original modes are Player-to-Game and Player-to-Player. These modes are quite self-explanatory. With Player-to-Game the player interacts with the virtual environment and characters. This always happens in any game, since the player “plays” a game, thus interacting with it. The other mode is the case in Multiplayer game-modes. Here the player does not only interact with the game, but also with other players. This can be through text-chat, voice chat, but also only competing against each other. The last and most recent mode is the Player-to-Platform version, such as the WII, where the player uses a device to register his or her movement for example which will be used as input.

Another thing to consider is how players interact with each other. There are multiple game-theories about this subject, such as the “Zero-Sum” theory[[19]](#footnote-19). In a Zero-Sum game both players have completely opposite goals, such as in chess. In Non Zero-Sum games the player coöporates, where they both have to achieve something together. Such as is the case in the Rockband-series where the players form a band to get a as high score possible.

To keep a game tempting and interesting the player needs to have challenges. These can be explicit or implicit challenges. An explicit challenge is a direct challenge, where the player has to achieve something set, such as find and retrieve a certain item. In RPG games these are often reffered to as Quests. These can be given tasks or suggested task. For example, when you see a door which says “Locked”, the door suggests that you should find a key. Implicit challenges are not specific nor direct. These are challenges such as how to divide recourses in a game or what to upgrade, which can be a challenge, but you won’t find it anywhere as a set task.

To complete these challenges, the player need information. When the player for example needs to find an item, he can get perfect, or imperfect information. If the information is perfect, the person who gives the challenge might say: “The item can be found in that cave in the bookcase to the right”. Imperfect information might be: “I left the item in a cave, but I can’t remember which, can you find the item?”. The player does not have all the information, making the task more difficult.

The last but one of the most important things to consider when designing the gameplay aspect of a game, is the balance. When a position or situation seems unfair to a player, he or she might lose the excitement in a game, and stop playing it or won’t recommend it to friends. Balance is not especially the fairness against other players, but the difficulty as well. The player need to start with easy goals, but yet slightly challenging. When the player goes further into the game, the goals and tasks need to progress in difficulty with the player. If the player gets frustrated the game is too difficult. When looking on the multiplayer aspect balance might mean something else. In a game such as “League of Legends”, the player must have a fair chance against opponents, even though the opponent might be an entire other character. In this game you play in an arena, with 2 teams. Each team consists out of 5 members, which are all different. A situation should occur where 1 player of 1 team can defeat the entire enemy team, since this would be unfair. Creating such balance is very difficult, and these type of games are often updated and characters are “nerved”; making them weaker.

How I designed my game

I set only a single challenge, which was to defeat the opposite player. This can be by weapons or direct impact. I had no story-line, since this is only meant to be a multiplayer-game. I thought a little bit more about the design of the art and graphics, than the design of the gameplay. Because I liked how my “prototype” version looked, I wanted to continue this type of drawing. Fat outlines with a more cartoon and fun style game. I would not choose bright-green or bright-red but more to the white shade, where it would look more like chalkboard colours. The balancing of the game is something which will be more difficult, since the rockets in my game might be hard to control, which would mean the player would choose an easier weapon above my rockets. I have to balance this with giving the rocket more damage or the other item less. I can only find the perfect balance by hearing feedback from other players.

Conclusion

On this matter, the differences are not that great. Of course I have a smaller scale, but the thought in the back of the mind is quite the same. The time taken for this process is for me much less in percentage of total development time compared with professional development.

# 5. Programming

How does a professional program a game?

Programming a game is actually the translating of a game, from idea to a product. But how does a developer program a game? What methods does a professional use to debug the game?

Just like a story needs a storyboard first, the code of a game needs something similar as well. In programming people often use something named UML[[20]](#footnote-20) which is a program made by OMG[[21]](#footnote-21). With this a programmer makes a very detailed scheme about every object in the code, from a save file, to the properties of a lamp post in the corner. Not every programmer works with this, but this is often used. When the diagram is finished, developers often have a “Code Review” phase. In this case a programmer should have his code checked by a colleague and make it more efficient when able.

The programmer should now program and test the game. Programming differs day-to-day. One day the programmer must solve a difficult issue and write an enormous algorithm, and the next he only needs to trigger some events. When the programmer has written a piece of code, he needs to debug this. It is important when the programmer encounters a problem, that he fixes it by finding the source of the problem, and not fixing the symptom. So when you forgot a minus, or divided instead of multiplied. Don’t add code to solve the issue, but find out what you did wrong, and correct that.

How I programmed my game

I programmed it without any use of a scheme. I made schemes in my head, and used ***Attachment 1*** to keep track of what my ideas were in a previous stage. The programming became more difficult when I had to do the collision checking. Read the chapter “Car Collision” on page 43 in ***Attachment 1*** for how I tackled this problem. I found this part particularly interesting.

Conclusion

Programming takes a lot of time and skill, but the process is almost always the same. The only way this can be described well is by your own experience and an example. With bigger games it is important to use UML, so you won’t lose track of what you have done, and work more efficient.

# 6. Testing the Game

How does a professional developer test his game?

Games cannot be released immediately without testing, because while everything might seem fine in the eyes of the developers, others might have another opinion. The game can contain bugs which occur only rarely, which can only be found by massive testing by multiple people, or the game can be boring or frustrating in the eyes of a player because of something. To make the game as good as possible when released to mass public the game goes through stages of testing. First comes the debugging, this happens in the process of development. Often multiple test phases come after the game is completed to ensure a quality product. To see more on what type of testers there are, see the chapter “Division of Work” with the subchapter “Testing & Quality Assurance” on page 17.

Debugging is a very, very important process, but a logical one as well. Debugging is when the programmer writes some code, and tests the code for bugs after completing some piece. I.e. the programmer writes a code to make a character move around in the game, and tests this. He finds out that when the player presses forward, the player goes backward. This can mean for example that the programmer has turned around the speed value, so forgot a “-“ in front of some variable. The programmer has to debug very often, since he will lose track of what he has done wrong if he writes to much at the same time, and only tests the code every so often.

When the game is playable from start to finish, the alpha phase begins. In this phase outsiders are brought in by recruiters or outsourced playtesting companies to test the game. These testers should test out every single item in the game, from the screen options menu to the firing of bullets. These testers tell the devlopers if they should add anything, or leave out anything. The players should also test if they can find ways to solve for example a puzzle which is not the way a player should do this. The players also set up a “bug-database” which contains the type of bugs that occurred and how they might have occurred. A list of what elements need to be completed is written down in the book *Game Development Essentials*  by *Jeannie Novak*.

* One Gameplay Path (playable from beginning to end)
* Primary language text
* Basic interface with preliminary documentation
* Compatibility with most specified hardware and software configurations
* Minimum system requirements tested
* Most manual interfaces tested for compatibility
* Placeholder art and audio
* Multiplayer functionality tested (if applicable)
* Draft of game manual

Then comes the beta phase, which is often public. There are 2 types of beta’s; a closed beta, and an open beta. When a beta is closed you can only join by invitation, when the beta is open anyone can join. Games which are entirely new and have no publicity yet, often try an open beta to get a wider market before actually releasing the game. This happens very often in PC-MMO games. This phase only focusses on testing and fixing bugs. The game is also tested by the hardware manufacturer in this process, by its own testing team. The developer might have had some rules to follow that were made up by the hardware developer to ensure quality of the product.

For indie games the games might not be tested by a the hardware manufacturer, but by other developers. This is named Peer Testing. I will take for example the XNA (Windows) way to test these games. All XNA developer have an app on the Xbox to manage their games. They can send in their games for peer testing, where other developers can open their app and test any random game. The game needs to follow a checklist with items such as “Player 1 can be any controller”. That is why you see a “Press Start” screen so often, where the Player 1 is decided by who presses start. Microsoft does not check any of these games at all in the end.

How I tested my game

This is directly taken from ***Attachment 1*** in which I fully describe my development process of my game:

“Now the game is at the stage where the Alpha Testing phase begins! I have asked 3 friends of mine to come over in my vacation. I made sure every save file was deleted before they started the game, and joined them. I did not explain anything to them, since any new player who would buy this game have no-one to explain the way the game is played to them. I asked them to test everything thoroughly. They should sign out on the Xbox and sign in again. Another new thing was that the game would be played by 4 people, and not the way I did it; test the game myself with 2 controllers. We had a great time and laugh, and I found some new bugs and had some feedback.

Bugs:

* The wrong player would have the wrong resources in the shop (e.g. player 1 would shop for player 3).
* Some of the cars were too strong.
* Some of the cars would keep doing damage if it would drive against a non-moving player.
* When restarting the game, wrong save files would be loaded

Feedback:

* Create different arenas in different shapes (Circles, Triangle, Random shapes)
* Unlock Arena’s not by buying, but by playing games.
* Start with a small amount of money

I especially found the feedback on how to unlock the arenas very interesting. It would attract the players to play more. I also had the problem that new arenas could be unlocked by player 1, so this person would have to use up his money on arenas, while others do not have to do this. They can spend this money on other upgrades, so player one would be disadvantaged. The start with a small amount of money was a good idea as well. When we started playing the game, it took at least 2 games to have enough money to buy something. With a small amount of starting money you could buy something immediately, or save for more. This made the game more exciting since the player does not have to play on the most boring level for multiple games.”

Conclusion

Not everything from the list that professionals should test applied on my game. I do not have to have a game manual, compatible with hardware and software or have minimal system requirements. Since my game is not very complicated and does not have many sides to it, I do not have to test is as extensive as a developer should. The last thing that developers should handle is the Marketing and Publishing.

# 7. Marketing and Publishing

How does a publisher market and publish his game?

This is a very important aspect of the process of developing a commercial game. While the game can be great and completely finished and ready for sales, not many people will know about its development. The players need to be reached, and amass. This can be done by social networks, advertising on media, through partners, and many more ways. There are also post-publishing steps such as support and keeping in contact with the players.

So the game is finished, but not for sale yet. The publisher which hired a development team needs to market the game to get as high sale rates as possible and maximize the revenue. First comes the most obvious step; advertising on media, such as TV, radio and game-sites. This takes up most part of the marketing, and costs the most as well. The publisher should be very careful what to set in the spotlight in the advertisement, and what to leave out. Games need to seem interesting and innovating. This is why publishers should show appealing and new content, such as new themes or new systems. Choose the right screenshots and gameplay, and you are bound to have a good start.

You need to involve the press and public as well. This can be done by a promotion team, which will set up events to promote the game, and who goes to game events. Promotion often includes handing out t-shirts and goodies with the logos of the game on the item. It also includes creating i.e. “30-Day trials” for a game, where a player can play the full game, but only for 30 days.

An important team is the public-relations team, the team that handles the press and public. They set up small first-look events for press so these people can play the game early to review the item prior to the release date. They also handle press-related advertising, through individuals who have for example a big audience on YouTube.

How did I market and publish my game?

There is one simple answer for this: I didn’t. This is because my games was only created out of educational interests and not commercial interests. Since I did not have to publish my game, which would take even more investment, and this would include material investment, I did not have to market it as well. If I would have published this game, I did not have to market it, and the publishing would only include handing it to Microsoft to let it be peer tested (See Chapter 5: Testing the Game).

Conclusion

Marketing is very important, because while a game can be perfect, you still have to make it popular. I find it interesting to see what marketing actions have most impact. While most money is spend on media-advertising, the game gets most knowledge by friend-to-friend advertising.

# 8. Overall Conclusion

After many, many hours of hard work I final In the meantime I created a game myself, which runs smoothly and is fun to play. It was very interesting to write, and fun to create, and I am happy I choose this subject. This is because while I worked more than necessary I had much fun creating this paper. So what did I learn and what stood out?

Well at first the thing that surprised me the most, is what I expected from the development of a game, and what actually has to be done for creating a game. I more expected that the hardest and most difficult part would be the programming of the game, where they have to solve difficult issues and come up with mathematical algorithms to execute certain operations. The case is actually the opposite. What seems and from what I have read from different statements of developers, the hardest part is the design. There is a clear explanation for this; the designing is productive work, and the programming is reproductive work. When you start to program the game, the entire game is visualized, and the game is already fully formed. The only thing is that the game has to be translated to work on a console. You can compare it with a book. The designing is the writing of the book, the programming is the translation of the book. While the translation is much work, it is only work already done, but bringing it to another state. It surprised me very much at first, but after more and more reading, it seemed clear and logical.

I found it very interesting as well, to be in contact with one of the programmers at Guerrilla Games, who helped me much in the project. For my full interview with him, see ***Attachment 3***. **WARNING: this is Dutch**. What surprised me is that he rejected the idea of trial & error, he stated that solving a symptom is not what you want, but to solve the core problem. You can go in discussion on this subject, since I believe the removal of a core symptom is what I do, and I still call it trial and error. When I come up with a problem, I check and correct what I did wrong, and do not create overlapping code to only solve the problem in a certain situation.

# Afterword

I want to thank you for reading this paper, which I hope was educational and interesting. For in any case this was very interesting and educational for me. I want to thank Ken van Hoeylandt with his help on this subject, and time he took for contacting me and explaining the process to a great extent.

Beer van der Drift.

Sources:

# Books:

* Game Development Essentials by Jeannie Novak. 3rd edition
  + Canada, Delmar Cengage Learning, 2012

# Websites:

* Unified Modeling Language on Wikipedia
  + <http://en.wikipedia.org/wiki/Unified_Modeling_Language>
* Zero-Sum game on Wikipedia
  + <http://en.wikipedia.org/wiki/Zero%E2%80%93sum_game>
* The Game Development Process on Worcester Polytechnic Institute
  + <http://web.cs.wpi.edu/~id111x/c05/>
  + By Mark Claypool
  + Published: Unknown
* The Process of Game Development & the Game Design Document
  + <http://digitalworlds.wordpress.com/2008/04/10/the-process-of-game-creation-the-game-design-document/>
  + Writer Unknown
  + Published: 10/4/2008

# Documents:

* The Process of Game Development
  + <http://www.pearsonhighered.com/samplechapter/0672326922.pdf>
  + Writer Unknown
  + Published: Unknown
* GGD Template
  + On CD with Game Development Essentials
  + By Chris Taylor

# Interviews:

* Interview on Game Development & Programming
  + With van Hoeylandt, Ken
  + By Drift, Beer van der
  + 2/2/2012-27/2/2012
  + See ***Attachment 3***

# Images:

* Figure 1,2 & 3
  + Screenshots from the Prototype of Epic Grip
* Figure 4
  + <http://s1.stliq.com/c/l/d/df/18262016_adam-adamowicz-1970-2012-0.jpg>
* Figure 5
  + <http://media.giantbomb.com/uploads/0/2069/179870-unreal_technology1_large.jpg>

Attachments:

* Attachment 1: Development of Epic Grip
* Attachment 2: GDD Template
* Attachment 3: Interview on Game Development & Programming (DUTCH)

1. Game Development Essentials by Jeannie Novak; Page 352 [↑](#footnote-ref-1)
2. Interview with van Hoeylandt, Ken; Attachment 3, Page 3, Bulletpoint 5.1.A. [↑](#footnote-ref-2)
3. Game Development Essentials by Jeannie Novak; Page 38-83 [↑](#footnote-ref-3)
4. Game Development Essentials by Jeannie Novak; Page 352-365 [↑](#footnote-ref-4)
5. Interview with van Hoeylandt, Ken; Attachment 3, Page 3, Bulletpoint 5.1 [↑](#footnote-ref-5)
6. Game Development Essentials by Jeannie Novak; Page 391 [↑](#footnote-ref-6)
7. <http://www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-csharp-express> [↑](#footnote-ref-7)
8. <http://create.msdn.com/en-US/> [↑](#footnote-ref-8)
9. <http://www.getpaint.net/> [↑](#footnote-ref-9)
10. ***Attachment 2*** [↑](#footnote-ref-10)
11. Game Development Essentials by Jeannie Novak; Page 319-322 [↑](#footnote-ref-11)
12. Game Development Essentials by Jeannie Novak; Page 323-327 [↑](#footnote-ref-12)
13. Game Development Essentials by Jeannie Novak; Page 329-331 [↑](#footnote-ref-13)
14. Game Development Essentials by Jeannie Novak; Page 332-337 [↑](#footnote-ref-14)
15. <http://www.unrealengine.com/> [↑](#footnote-ref-15)
16. <http://www.geforce.com/Hardware/Technologies/physx> [↑](#footnote-ref-16)
17. Game Development Essentials by Jeannie Novak; Page 338 [↑](#footnote-ref-17)
18. Game Development Essentials by Jeannie Novak; Page 339-341 [↑](#footnote-ref-18)
19. <http://en.wikipedia.org/wiki/Zero%E2%80%93sum_game> [↑](#footnote-ref-19)
20. Interview with van Hoeylandt, Ken [↑](#footnote-ref-20)
21. <http://en.wikipedia.org/wiki/Unified_Modeling_Language> [↑](#footnote-ref-21)