03/2015

making games

DESIGN | BUSINESS | ART | TECHNOLOGY

THE NEXT GENERATION OF MOBILE

FREE2PLAY-ANALYSIS
HOW BLIZZARD, POPCAP, UBISOFT AND KING BIND THEIR PLAYERS.

CASE STUDY NEVERWINTER
HOW TO PORT AN ONLINE-RPG FROM PC TO CONSOLES.

LORDS OF THE FALLEN
WHAT LEARNINGS DECK13 GAINED FROM CREATING A AAA-GAME.

03
4 197050406909

THE FUTURE OF MOBILE? MOBILE IS THE FUTURE!

At the beginning of March, what seemed like the entire global games industry gathered in San Francisco for GDC in order to exchange their knowledge and look out for any emerging innovations and trends. For the public, there was one topic that dominated the conference: VR! Oculus presented its Rift, Sony introduced their latest version of Morpheus, and Steam inventor Valve topped both competitors with SteamVR, incited visitors and journalists to heaps of praise, and even made people from outside the industry erupt in enthusiastic cheers. But one trend that seemed to fly very much under the radar for most was the buzzword »emerging markets«. While core gamers threw themselves at »nerdy gadgets«, high-end graphics and AAA games, big publishers like Ubisoft or Warner – just like probably many others – were working meticulously on a master plan for how to best tap into markets such as Latin America, Asia or Russia. The problem is that in these countries consumers generally have less money than gamers in Germany, the UK, France or the US and thus own less powerful hardware or even no PC or console at all.

The final frontier

Both said publishers have revealed the answer to this dilemma in a recent interview: Android smart phones and tablets in particular are relatively inexpensive these days and thus have a much broader »install base« than traditional gaming platforms in these countries. Consequently, publishers will focus more and more on mobile games in the future in order to reach these new target markets. They're assuming to tap markets of potentially hundreds of millions of gamers after all. Many gamers suspect quick free2play rip offs in this area and turn their back on mobile gaming, but there are plenty of examples which show that there are other ways, too: Daedalic, for example, are fully porting their adventure games from PC to iPad (p. 12), Ubisoft is even working on a mobile game to be used for medical treatment (p. 20), and a so-called »serious game« is to be used by teachers in the classroom to teach students about tolerance and moral courage (p. 28). So the next generation of mobile games is probably set to take a big step forward, not only in terms of revenue for the publishers, but also in terms of relevance for the players – even though there will probably always be one black sheep or the other.

Enjoy the latest issue!

Your Making Games Team

Sebastian Weber
is Managing Editor of Making Games Magazin.

»Publishers focus more and more on mobile games in order to reach new target markets.«

FROM MOUSE TO FINGER
ADVENTURE-PORTING FOR MOBILE DEVICES

Daedalic Entertainment's adventure experts are in the process of gradually porting their PC games one-to-one to mobile platforms and, while doing so, they had to learn that the effort of porting is often on a par with developing a game from scratch.

Matthias Zorn
Senior Producer &
Localization Director
at Daedalic Entertainment

Matthias, who has a degree in mathematics, used to be editor-in-chief of the specialist gaming website Adventurecorner.de before he started to work at Daedalic Entertainment in 2008 where he was a project lead and producer for titles such as »A New Beginning«, »Satinavs Ketten« (Chains of Satinav) or »The Night of the Rabbit«. In May 2013 he became Localization Director, supervising all of the Hamburg-based developer's localization jobs, and since mid-2014 he's also been in charge of the mobile porting of their adventure portfolio.

For developers of video games, tablet PCs have become increasingly important in the last few years, but with the growing number of devices the pressure of competition in the market is growing as well. In 2014, an average of 500 new iOS games and 250 Android games were said to have been published daily in the respective app stores. Obviously, a lot of these releases go by the consumer unnoticed. These numbers demonstrate how much harder it is to tell how successful mobile development is going to be than it was only a few years ago.

Apart from numerous new developments, there are more and more releases of ported games which were originally developed for other platforms and now make a fresh start on mobile devices.

A few years ago already, Daedalic Entertainment released their first adventure porting »Edna Bricht Aus« (title of the English PC version: Edna & Harvey: The Breakout) for iOS in the app store. The release was followed by several mobile in-house developments like the »Living Stories« games »Das verlorene Herz« (The Lost Heart) and »Das verlorene Lied« (The Lost Song) as well as puzzle games to accompany the »Deponia« trilogy and the »Edna Bricht Aus« (Edna & Harvey: The Breakout) series. Mobile versions of »Deponia«, »The Whispered World« and »Fire« are eventually to follow this summer as part of a series of games that are being ported for iOS, before Daedalic's entire point&click adventure portfolio is gradually to be ported and released.

Input modes: Individually or all-in-one?

It's probably every developer's dream to publish their games on as many platforms as possible in order to reach as many gamers as possible. What has obviously to be taken into account is that not every game is suitable for every platform. A turn-based strategy RPG like Daedalic's »Blackguards« where a single fight can last as long as 30 minutes and more is certainly less suitable for a mobile porting than titles which can be interrupted more often and be finished in shorter periods of time.

Daedalic's first adventure port for iPad was »Edna Bricht Aus« (Edna & Harvey: The Breakout) which required a new interface and new controls compared to the PC version.

From summer 2015, Daedalic's entire adventure portfolio will gradually be published for iPad, starting with »The Whispered World«.

In terms of the game controls it is also necessary during development to take the characteristics of the respective platform into account: While a PC game would obviously be played with mouse and keyboard, console games need to be controlled with a gamepad, while mobile devices require touchpad controls. That's why it is advisable to consider early how the controls and the corresponding interface should look like on different systems. It makes no sense trying to use the same controls on all systems – controller, mouse, keyboard and touchpads are simply too different. In many cases, controls reduced to work on all systems would limit the gameplay mechanics in a way that there would hardly be anything left of the game.

It's for this reason that we at Daedalic try to regard the controls for each system individually and adjust them according to the given features. It's particularly important for us that the game is fun to play on each target platform and that the controls are intuitive.

Point&click adventures have the advantage that the general type of game automatically feels very natural using touchpad controls. Therefore, it's not mandatory for a successful port to feature major gameplay adjustments. Quite the contrary: Adventures have a more intuitive feeling to them on tablets than with mouse controls on PC. Yet, the porting process is, of course, no no-brainer that can be done on the fly.

Apart from facing technical problems such as memory management and possibly limited hardware functions, we have to adjust the controls as well in order to emulate commands carried out with the right mouse button or keyboard on PC. Due to the limited screen size of tablet PCs, the developer has to make sure that all items and interactive areas in the game are presented large enough and have enough space between each other in order to be able to comfortably tap them with your finger.

In most cases, it's sufficient to keep these problems in mind when developing the PC version. However, since the possibility of porting to tablets couldn't be foreseen in times of, e.g., »The Whispered World« or »A New Beginning«, the porting required comprehensive adjustments.

Porting: External partner or do-it-yourself?

If a development studio is focused more on classic PC games rather than on mobile games as is the case with Daedalic Entertainment, they should thoroughly consider at the beginning of production whether to do the porting themselves or transferring the task to a partner.

For Daedalic's first few adventure portings this question was still easy to answer: The engine that was used wasn't innately able to run on mobile devices. Changes to the source code couldn't be made by our in-house studio, but only by the engine developer, and all programmers were already busy with other projects so that they couldn't just spend a part of their time on a project of such scale.

Soon the decision was made to work with external partners that were to port the games to an engine capable for mobile – a step that didn't turn out to be ideal for us.

The puzzleventure »Fire« is also to be published for iPad by the end of summer. The PC version will hit the market a few months earlier in April in order to prevent diluted prices in the various markets.

The iOS interface of »Deponia« in the first draft and in the beta version of the game.

The effort it takes to port entire adventure games to a different engine is hard to estimate. In addition to the basic mechanics that are usually quite easy to port, there are numerous special cases such as mini games, close-up puzzles or more complicated puzzle structures which constantly push the game engine to the limit. Due to this complexity it's not easy to work out a time and cost estimate.

During development we had to learn that without constant contact to our partners for providing direct feedback not much would happen. Milestones were delayed further and further and schedules could only be kept when programmers of the partner worked on the porting directly in our studio. Although we were happy with the result in the end, the path to get there was unnecessarily complicated.

Even though these problems surely can't be generalized and there are certainly some good reasons to work with external partners, we still decided to carry out all mobile portings in our own studio, starting with »Fire« and »The Whispered World«.

Visionaire Engine

Over the years, Visionaire Engine which was used for most of Daedalic's adventure games has taken a few considerable steps forward. For example, when developing our first Visionaire project, the PC version of »The Whispered World«, it wasn't possible to work on the game with more than one scripter at a time. Porting games to other platforms was something we basically never took into consideration. But over time, all problems were solved and more and more platforms were added. So now not only Windows, Mac and Linux versions of the games can be developed, but iOS portings are also possible in principle. While the Visionaire team is in charge of integrating new features such as a zoom function and 64-bit support requested by Apple, all changes and adjustments in terms of graphics, scripting and game design can be implemented directly at Daedalic.

Game design adjustments

Over time, different interface types have been used for Daedalic adventure games: from the verb system in »Edna Bricht Aus« (Edna & Harvey: The Breakout) to the context-sensitive coin system in »A New Beginning«; left-click/right-click controls in »Deponia« or »Satinavs Ketten« (Chains of Satinav) to one-click controls in »The Night of the Rabbit«.

All these systems have their pros and cons. While touch systems are begging for comfortable one-click controls, for the developer it takes away some options in terms of puzzle design and thus some of the game's complexity.

But since consistent controls are to be used for all of Daedalic's mobile adventures, without having to put up with reduced and limited functions of the original game version, the coin interface turned out to be the most user-friendly solution.

By tapping a hotspot, a menu opens up that offers several interaction options like »View«, »Use«, »Take« or »Talk to«. By tapping the respective option again, it will be executed. This system has been used for a number of PC adventure games since »Monkey Island 3«.

On tablet, the finger of a player can easily replace the mouse pointer in order to interact with items and people in the game or to scan the screen. Nevertheless, this system may also cause problems in terms of game mechanics.

In most Daedalic adventures, for example, there's usually a situation at some point during gameplay where a player picks up an object which they have to use straight away with another object in the game world without adding it to their inventory. On the PC, that's no problem; the object is simply attached to the cursor and can be used by clicking left or be deselected by clicking right. On tablets, none of that is possible. Neither can you attach the object to the non-existent cursor, nor can you just simply deselect it. And that's only one of the obstacles that have to be overcome by amending game design and functions.

For the porting process, we also have to think carefully about how to transfer functions which originally were controlled via keyboard like, for example, opening the main menu or activation of the hotspot button. One option would be, for example, to integrate a separate control element for all functions into the game and display it permanently. However, this may quickly result in the game being overloaded with interface elements and thus losing a lot of its charm. Gesture commands like swipe gestures and touching the screen with two fingers are another good way to integrate functions like the hotspot display or skipping cutscenes and longer animations. At the end of the day, you have to consider for each game individually which porting method makes most sense and is the most intuitive one.

Technical adjustments

Apart from the gameplay adjustments there are, of course, some technical problems that need to be addressed as well.

Hand-drawn 2D adventures need a lot of memory space for obvious reasons. Each background and animation is not calculated in real-time, but needs to be loaded as an image sequence in the game. A longer animation can therefore easily consist of dozens of individual images which not only increase the file size of the game but, of course, also the loading times. However, especially on mobile devices long loading times and huge app files are far from desirable. For that reason it's important to reduce the amount of data created by the graphics as much as possible.

When porting older titles like »The Whispered World« or »A New Beginning« this is slightly easier due to the lower original resolution of 1024 x 768 pixels compared to new titles that were developed in full HD. Converting the previously used PNG graphics to memory-saving WebP format plays an important part in solving this problem.

Yet, additional options like reducing the game resolution or removing individual animation frames need to be considered in order to reach a good compromise between graphic quality and data volume.

Why not Android?

Despite the rise of the Android market, we focus on ports for iPad for now. The main reason for this is the enormous variety of Android devices. While it is good for the consumer to have various options for choosing the right device, it's a downright nightmare for developers to test a product so that it runs smoothly on every single device. Since most of our adventure games with 2D backgrounds are developed in a fixed aspect ratio, it takes a lot of effort to support all the different resolutions and hardware configurations. At the same time, every time a new device is released and the operating system is updated, we would have to test whether the ported games still work without problems.

And then there is the additional time it takes to copy a test version of the game to a device. The effort is disproportionately greater than with a PC game, and the debug possibilities are also significantly restricted. Even though we will be happy to port to Android in the long term, at the moment it is simply too time-consuming and complex to support such devices.

Pricing and time of release

Apart from the actual porting of an adventure game from PC to mobile devices, the pricing and the right timing for release may cause considerable issues, too.

For the customer it would certainly be best if a game was published simultaneously for all platforms and they could choose which system they want to buy and play the game on. Sounds good, but is almost impossible to put into practice. The retail price of an adventure game at release has dropped continuously in recent years. What used to be 50 euros originally is now not even half the price. Additionally, a lot of games become available at dumping prices just a few months after release through sales campaigns by various digital vendors. Consequently, a lot of gamers rather wait for such sales instead of buying the games as soon as they're on the market.

But since developers are dependent on earnings especially in the early stages in order to compensate for the costs of subsequent projects and to keep the team together, sales at full price are absolutely required.

A simultaneous release on tablets would add to the problem. The price perception in the various app stores doesn't have a lot to do with that of a physical release any more. A game that costs more than 5 euros is already considered to be terribly overpriced. If said game was published at the same time with the same conditions at a significantly lower price than for PC though, PC sales would drop even more. Prices would consequently continue to drop and everything would be done to achieve the highest sales numbers possible for a product to pay off at all. This means, however, that developers could not only serve smaller niches like the adventure market, but also would have to meet a rather broad taste with as little access barriers as possible.

Last but not least

It may seem easy to »simply port« a finished game, and yet a lot of things have to be considered.

While most technical adjustments can generally be used for all sorts of games of the portfolio, it's an entirely different story in terms of graphical and game design adjustments which need to be thoroughly assessed individually for each title. Therefore, depending on the scope and detail work, a mobile port is a lot more than just a medial adjustment; it may take almost the same amount of time it would to create a whole new game.

When looking at the mobile market though, iPad and Android ports are definitely worth the effort and are indeed feasible if you have the necessary patience for making the required adjustments.

For this reason, from summer 2015 Daedalic's entire adventure portfolio is gradually to be ported to iPad, and we think in the long term we will also manage to handle the challenges of Android portings.

Matthias Zorn

A look at Daedalic Entertainment's QA department while testing the iPad version of »Deponia«; the adjustments of both controls and partly the game design have to feel intuitive for each platform.

PENGUIN BREAKOUT A GAME MADE (MOSTLY) IN THE SUBWAY!

How do you develop a game when you have a day job, spend two hours per day on public transportation and your team is spread across two continents? You do the coding on your tablet, for example. The team of pixelbizarre explains how they created their first game »Penguin Breakout«.

Corrado Longoni is Co-Founder of pixelbizzare.

Other than being co-founder, Corrado is also pen tool doodler, Codea zen garden keeper and gameplay handicraftsman at pixelbizarre. While some panhandle in the subway, he prefers to code in the subway.

Xavier Damon is Co-Founder of pixelbizzare.

Xavier is chief xCode player (and prayer), tunes and iTunes guru at pixelbizarre. He loves penguins, amongst other things.

Olivier Rozay is Co-Founder of pixelpizzare.

Olivier's duties include being the chief marketing minion and level design bondservant at pixelbizarre. He codes with Adobe Photoshop and Microsoft Word.

We founded pixelbizarre, a small indie game studio, in June 2014. Our aim is to craft small games (hence the »pixel«) that explore new game designs (hence the »bizarre«). Our team is split between Paris and San Francisco. We met in college, but went separate ways with our professional careers. We had always dreamed of making video games but it was too expensive to enter the business. With the arrival of app stores and cheaper development environments, our dream suddenly looked achievable. After seeing simple, but still exciting, games like »Cut The Rope« or »Fruit Ninja« we decided to get started and created pixelbizarre. Since others succeeded, we decided to give it a shot. Our game studio is a genuine home brew indie structure. We did not take any external investment and from the beginning decided to not take things too seriously! More importantly, we make sure to have fun while making games, and hopefully some of that good spirit comes across our games. This is the story of how we made our first game, »Penguin« Breakout, in the subway.

Game design

We try to make games that appeal to the broadest audience: our friends, our parents, and our kids. Our games are aimed to be played in short sessions: while standing in line, while waiting for the elevator or even while sitting in the bathroom. We design games that are very accessible but still offer real challenges in the long run.

The game design started with the mechanics. We mixed two well-known classics: the 15-puzzle sliding tile and the marble labyrinth to come out with a special breed. With these two elements, we caught the smarts of a puzzle game, but also the fun and speed of an arcade game – all wrapped in our own unique gameplay.

As we borrowed from classics, the game mechanics only take a few seconds to learn. Slide the labyrinth pieces like in a 15-puzzle, tilt your device like a marble labyrinth, and you got it.

The theme came after; we wanted something more appealing than the cold and classy marble design. After several trials (lovers, balloons, etc.) we eventually found the »raison d'être« for the game. You guide a penguin home through a labyrinth of icy tiles.

To provide a longer game experience, we then added game elements for more variety and additional challenges. As you progress through the game, the penguin will have to use special features like teleporters, springboards, and other nasty elements to get you to its destination. The game progresses through 120 levels organized in four seasons. Each season introduces new add-ons, obstacles, features, and a specific graphic theme.

The mechanics are simple, natural, and still provide for progressive challenges and complex gameplay. The game controls take full advantage of what makes the experience unique on a mobile device: the accelerometer and the touchscreen. We incorporated both mechanisms in the design of »Penguin Breakout«. The movements of the penguin are controlled by the accelerometer, and the movements of the tiles are controlled via the touchscreen. We debated for a long time whether we should introduce D-pad like controls for the penguin (something that a few beta users had suggested), but in the end we decided against it and kept a simple UI.

We have been receiving many compliments on how much fun and natural it is to control the main character.

To guide the penguin home, you have to solve increasingly difficult puzzles. The game progresses through 120 levels organized into four seasons.

Game development

Due to our situation (spending a lot of time in the subway/bus commuting to our day job), we needed a development environment that could be used in public transportation. Coding on a PC or on a laptop was out of the question due to the lack of space in »tiny Europa« crowded wagons. We had been playing with Codea for some time. Codea is a fantastic, handy, and performant LUA-based framework from Two Lives Left that made this possible. The math is quite simple: two hours of commuting each day and you immediately get ten additional hours on your game a week. Much more productive than staring at your feet while dreaming of a game you could have made.

»Coding on an iPad, you are nuts!« Well, we did not have any choice. To finish the game, we have to optimize our free time at best.

»No keyboard, how do you type myHundredTwentyCharsLongVariableName?« Codea runs quite efficiently on an iPad. It provides a handy code editor to deal with writing code. It can also handle reasonably vast projects. Of course we made some arrangements, our variables names are shorter than usual and not camel compliant, but in the end we got used to it.

»LUA-based, it must perform like a turtle on Valium?« Well, once you know a few basic tweaks and tricks, you can easily reach the holy 60 fps for a 2D game. From the framework, you can program the shaders. For this first game, the animations were done using spritesheets. A simple vertex shader sets the proper vertex texture, so the CPU was left to handle user inputs and physics. For our upcoming title, we are working on running Spine skeleton animations from within the shader to greatly improve the animations' quality.

»How many zillions of libraries did you have to write?« Through Codea, we have access to the shaders, box2D physics, basic network access, accelerometer, etc. We still had to write a few higher-level libraries: a fast font engine, a level editor, UI elements, and a navigation framework. We made sure these libraries could be reused for all our next developments. We hope to open source these libraries, once the code is more tidy and clean. The Codea community is quite active: we used a community developed tool to access GitHub from within Codea.

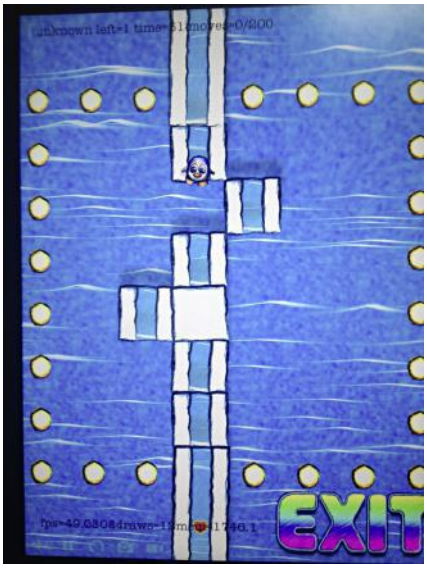
»It must take ages to fine tune the game?« This is one strong point of the framework: WYSIWYP (What You See Is What You Play). Code, press play, and you immediately see the final rendering! This proves very efficient to fine tune the gameplay. No lengthy compile-build-upload cycles just to fine tune game elements. As our gameplay is 100 per cent tactile and accelerometer based, testing on a PC is quite useless. In the same prospect, including the game editor inside the game has proven efficient as well. Create your level and then test it right away wherever you are (in the subway amongst others).

Graphics and sound

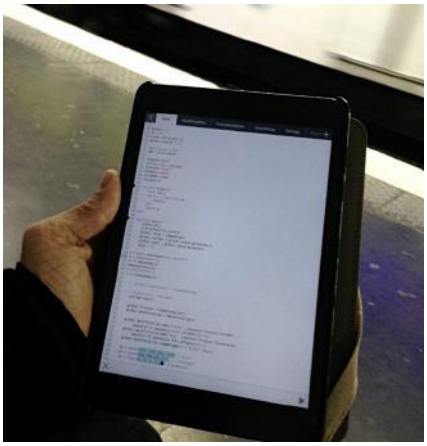
Being a puzzle game, there can be not much happening on the screen for a while. In order to breathe some life while the player is scratching his head we gently animate every game element. The tiles gently float over the water, the background shivers hypnotically, and the characters are drawn using sketchy lo-fi lines that vibrate with each frame to breathe.

We added specific animation to provide a subtle feedback for the players' actions. A small animation runs whenever a tile is caught or dropped. Every event comes with its animation. In order to render the tilting, we use a simple parallax effect on the background and simply shift the shadows. These simple effects are sufficient to make the game react without impacting the performance.

For the soundtrack and sound effects, we were fortunate to have experience with music. Two of us play in a band, so it was relatively quick to compose the soundtrack for »Penguin Breakout«. We goofed around with dubstep music and industrial tunes, but in the end we decided to stick to a lively and cartoonish tune.



The very first design concept for »Penguin Breakout«.



The team used the LUA-based framework Codea for the game and did all of the coding on their tablets while using public transportation.

The team used Penguin Awareness Day and World Penguin Day to spread the word about their game.

Level design and testing

In order to facilitate the level design, we decided early on to build a level editor inside the game. This took some work within Codea, but enabled us to have a very flexible and distributed process for level design. One of us could spend time coding new functions, while the other one, idle in the subway, could spend time designing and testing game levels. Each level has its own puzzle or solution and comes with its own layout and challenge (maximum time allowed or maximum number of moves to complete).

Since this was our first game, we highly underestimated the time it would take to calibrate each level. We tried to automate user testing by getting our friends to test a beta version that included feedback forms at the end of each level that the user could fill out on their own. However, we quickly realized that friends were not always willing to spend time providing constructive feedback! We ended up doing assisted user testing where we had to sit near each tester and take notes as they were playing (and often struggling to finish a level).

As absolute beginners, our initial level design was far too difficult. The surprise came one week before submitting the game to Apple – we were very happy with our collection of 160+ brain twisting levels. We had tested each level individually and had calibrated their difficulty to a level we believed was achievable. However, in doing additional testing with fresh new users, we realized that the game was way too hard! So, a couple of days before submitting to Apple, we completely rearranged all the levels, removed 40+ levels, and made many of them much easier to complete.

We are glad we made this move, as the progression between levels feels a lot smoother now. From time to time, we do find that certain levels could be re-arranged – this ends up being a continuous improvement process that fortunately we can manage through software updates.

Business model

Penguin Breakout was our first game. When we started the project, we didn't know much about game mechanics, and even less about game economics. We focused our time and energy on game design and opted for a very simple monetization model: freemium. We thought that we should make a free version with one season (40 levels) that would get the game out and discovered. Once users got hooked, they would want to buy the full paid version which included four seasons, 120 levels and provided many hours of gameplay.

This model certainly worked well a few years ago with games such as »Angry Birds«, and »Cut the Rope«. However, these days no one wants to pay anymore for any apps upfront, particularly in the puzzle/arcade game market category. Our conversion rates from Free to Paid are very low and would certainly not look very appealing to any large studio or even any indie publisher.

It seems Free2Play is the way to go to these days, but the game mechanics in »Penguin Breakout« were not optimized for that kind of impulsive in-app purchases. The game doesn't have any boosters or power-ups, since each level is meant to be completed using pure imagination and skills. So, to better monetize »Penguin Breakout«, we would need to redesign the game from scratch with monetization in mind.

We're happy with the current state of »Penguin Breakout« (users, particularly those who are into puzzle games, love it), so we've decided to experiment with new monetization models in future games and not to touch the current »Penguin Breakout« design. In our next games, we will certainly focus more on small in-app purchases for power-ups or other consumables that enhance the gameplay, but still do not spoil the game experience.

We are fortunate that our current cost structure is close to zero so we are not too worried about monetization and conversion rates. In the early days, we experienced with Facebook user acquisition – the system is pretty easy: you set it up in very little time and it feels very empowering to be able to target »puzzle game enthusiasts« in »the UK, Italy, France« with a few clicks. This proved very rewarding initially with user acquisition costs being below 1 Dollar, but then when we looked at the conversion rates to our paid app, it didn't make sense to continue investing. We will certainly experience more with our next games, but the Facebook acquisition model did not work very well for us considering »Penguin Breakout« is a freemium game.

Marketing

We launched »Penguin Breakout« with a marketing budget of zero dollars! Similar to most indie game developers, we didn't have a big marketing department to support our launch, so we had to do it all alone the hard way! We set up the usual Facebook and Twitter accounts and befriended as many »friends« as we could. We spent a good amount of time developing a 15 second trailer video – this became a very useful piece of marketing that we uploaded onto YouTube and other video sites. We are now trying to stay active on social networks, but it takes a lot of efforts to produce content and keep activities flowing.

We reached out to as many gaming sites and online publications as we could. Not having a PR firm meant we had to spend time ourselves scavenging the Internet for email addresses and other contact details. We did a press release and sent hundreds of App Store promo codes to journalists and app review sites. In the end, we got a very limited coverage – we received many emails from app review sites asking for money to get reviewed. The majority of the emails we received said the same thing: »we are swamped with app review requests, and the only way

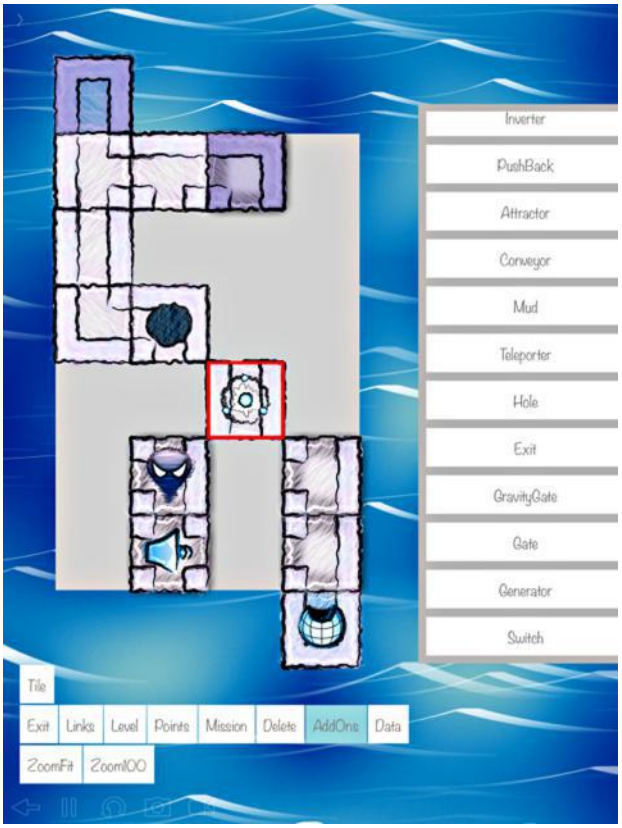
The studio

Founded in 2014, pixelbizarre is an indie game studio developing fun and lively games. It focuses its energy on original gameplay and content development, with a »don't take yourself too seriously« attitude. Its first title, »Penguin Breakout«, is a casual iOS game at the intersection of arcade and puzzle. Its second title, »Pumpkin Blaster«, is a fun arcade game about blasting pumpkins.

www.pixelbizarre.com

pixelbizarre

The game's debugging also happened directly on the iPad in Codea.



The level editor of »Penguin Breakout«.

to ›jump the queue‹ and get a review is to pay for the expedited package.« We also got many emails asking us to pay to get reviewed in the App Store. In the end, we didn't buy into any of these schemes (or scams?) because they required money (which we didn't have), and also all these paid services didn't seem right.

We have gotten good download numbers from making our paid version free for a limited time. We did this promotion for Black Friday in the US (end November) and during Casual Connect (in February). It was not a sustainable practice, but helped us get the word out. In addition, we have been capitalizing on international events to drive awareness and downloads of »Penguin Breakout«. For example, during Penguin Awareness Day on January 25th and World Penguin Day on April 25th, we tweeted cute videos of »Penguin Breakout«.

To help get the word out, we also submitted »Penguin Breakout« to a number of international indie game festivals and showcases. We got super excited when our game got selected to compete in the IndiePrize show at Casual Connect Europe in Amsterdam in February 2015. This is one of the premier gaming events in the world and we were stoked since we were selected to present »Penguin Breakout« there! The tight deadlines of the show forced us to get our marketing material in shape – we designed and printed flyers, stickers, t-shirts, and even updated the website! The show was a great experience – we got a chance to demo »Penguin Breakout« to hundreds of gamers and developers. This was an invaluable experience – not only did it help get concrete feedback on how to make our games better, but it also showed us how much creativity and talent there is in the indie game world! We also met some potential publishers who gave us direct feedback on how to make our games more monetizable.

Conclusion

We have learned a LOT while making »Penguin Breakout«. In the 15 months it took to develop this first game, we've been exposed to the widest range of Do-It-Yourself projects: from drafting a shareholder agreement, to animating bones in Spine, to setting up a user acquisition campaign in Facebook, to building an iPad safety enclosure for a tradeshow! Overall, this has been a very enriching experience, and we can now use our newly gained knowledge and energy to focus on designing our next game!

The amazing part of this project is how easy and cheap the essentials of making a video game have become – free Dropbox, free Skype, easy distribution to millions of devices, inexpensive game editing software, etc. It's great to see all these tools available to everyone – every aspiring indie game developer can now have their chance at making a game. However, gamers' expectations for mobile games have risen dramatically in the past three years. The barrier to be successful is now much higher. When designing a game, you need to project yourself a year into the future. You have to envision the idea, the gameplay, the monetization and the promotional channels that make sense not today, but next year! At pixelbizarre, we are now working on our third game and hope the adventure will continue to be as much fun as it has been for our first two games!

Corrado Longoni, Xavier Damon, Olivier Rozay

pixelbizzare schowcased their game at Casual Connect Europe in February 2015 and competed in the IndiePrize.

TRANSFORMING GAMES INTO MEDICAL TREATMENTS

Ubisoft is working on a game that will need to be prescribed just as a normal medication. We talked to the designer behind this concept to find out how Ubisoft ended up at working on »medicine« and what challenges this kind of games bring up.

Mathieu Ferland is Senior Producer at Ubisoft Montreal.

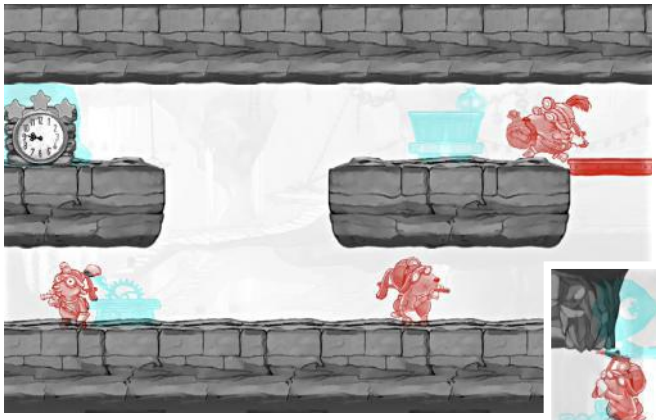
Mathieu Ferland joined Ubisoft Montreal in 1997. He played a significant role in the development of Tom Clancy games, including the creation of the highly acclaimed »Tom Clancy's Splinter Cell«. In 2008, Ferland has focused on defining transmedia and convergence strategies for Ubisoft's intellectual properties. He's been influential in leveraging Ubisoft's video game IPs into new entertainment mediums, including expanding Ubisoft's blockbuster video game franchise »Assassin's Creed« into non-gaming consumer products, such as short films, novels and comic books with the foundation of Ubiworkshop.com. Mathieu is now in charge of the development of creative and technological solutions.

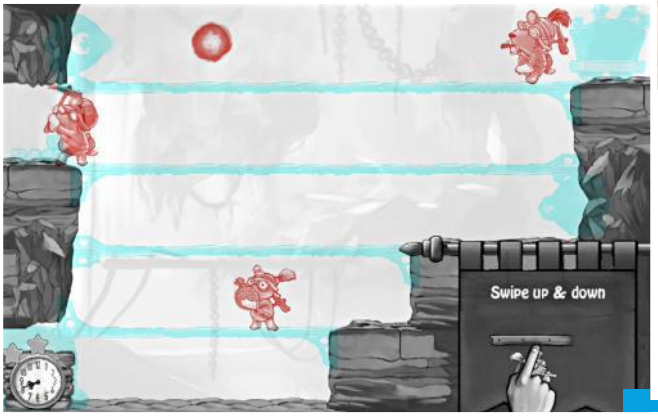
Making Games First of all, could you quickly summarize what »Dig Rush« is all about?
Mathieu Ferland »Dig Rush« is the result of a collaboration between McGill University in Montreal, Amblyotech inc. and Ubisoft. It is a new medical solution, a therapeutic game to treat amblyopia. This condition is also known as »lazy eye« and it affects 3 percent of the worldwide population. Every person has a dominant eye. Amblyopia is a condition where the domination of one eye becomes absolute. The vision in the »amblyopic« eye is suppressed by the other dominant eye. Gradually the amblyopic eye reduces visual acuity and can become legally blind. It is the number 1 cause of blindness for children, and until now, amblyopia was very hard to treat for adults.

Making Games Could you describe how the unusual collaboration between Ubisoft and Amblyotech started?
Mathieu Ferland I was participating in a talk focusing on technologies for health at McGill University, where I met Dr. Hess (from McGill University), who is responsible for developing this new form of treatment for amblyopia. Dr. Hess was presenting his innovation and expressed the need for a more engaging game format. I was giving a presentation about the engagement within a gaming experience, so the link was clear for everyone. Dr. Hess introduced me to the team at Amblyotech as the commercial partner for the technology, and we then worked together to transform the medical principal of the treatment into a commercial product.

Making Games »Dig Rush« is a very different game compared to what Ubisoft did in the past. Why did you decide to work with Amblyotech to do a game to treat amblyopia?
Mathieu Ferland This project has been a good opportunity for us to use our expertise in creating a gaming experience that engages players, in this case patients. Throughout this project, each partner has focused on its strengths. Ubisoft is good at developing engaging games. McGill University is at the origin of the innovation and manages all scientific aspects of the project. Amblyotech brings its pharma experience, is coordinating the regulatory aspects of the project, and distribution of the final product. Thanks to these partnerships, we were able to accelerate our work in this domain and to push video games beyond the realm of pure entertainment.

Making Games After you decided to work on »Dig Rush«, how did you approach the development of this game and how did developing a game like this differ from a »normal« game?
Mathieu Ferland Thinking about the fun factor of the game was quite similar to what you envision for any other tablet game. However, to properly integrate the principle of the treatment, the design of the game is quite different: it's been designed as a »right & left eye« game. The user is wearing old fashion bicolor 3D glasses, but not to generate a 3D image. All that matters is the color of the lenses: red and blue. We visually designed the game over a black and white background, where only the interactive ingredients are either red or blue. This is how we can target these to the right or to the left eye. We've balanced this design in such way that it is impossible for the user to play without using both eyes. Forcing the user





»Dig Rush« uses quite minimalistic graphics. This way, each eye only sees a specific part of the game and therefore the brain has to learn to use them both to see the whole picture.

to play with both eyes is the first step to train the brain to recover a binocular vision.

Making Games Were there any game design decisions made that you had to have in mind for a game to treat amblyopia?
Mathieu Ferland We had to keep in mind that all clinical data that demonstrates the efficacy of the treatment are based on gaming principles we needed to stick to. For example, we wanted a 2D slow paced game to make it easier for the patient to play using the lazy eye. We wanted to have fixed and moving objects to interact with. We wanted a game that appeals to everyone, that is easy to pick up, but hard to master.

Making Games How hard was this process of game design and development for you if you compare it to other games where you don't have to think about which elements should be seen by which eye at a certain moment so that the gameplay works?
Mathieu Ferland There was some trial and error during the prototyping stage. However, we realized that it was easier to first think about the gameplay ingredients as a whole game, and after only, to make it a »right eye / left eye game«. With this in mind, our creative process was quite similar to developing any other game to begin with. We even had full color within the game. Then we went through different iterations to end up with a well-balanced game where the two eyes are constantly and equally involved.

> »Our creative process was quite similar to developing any other game.«

Making Games How long did the development of »Dig Rush« take and did you have to hire developers specialized in certain methods or technologies to develop a game like this? Or would you say that any game developer might be able to work on games for health?
Mathieu Ferland It took about one year to develop »Dig Rush«. We needed to work closely with the researchers at McGill to get a full understanding of the science that makes this an efficient solution, and to evaluate the best way to implement it within Ubisoft's way of developing games. This was part of our R&D and prototyping period. Once a good prototype was validated by McGill's team, we were ready to proceed to production. One of the strengths about such a partnership is that we didn't need to hire doctors, or specialized health professionals for development; we've been able to support each other at every step.

Making Games Can you describe the technical differences of »Dig Rush« compared to a »normal« game and how exactly the game works from a technical point of view?
Mathieu Ferland The first obvious difference is the visuals: red and blue over a black and white background. Moreover, the game needs to be prescribed and calibrated by a doctor before the patient can play the game as a treatment. We've implemented a physician interface to track user data and to calibrate the game. A person suffering from amblyopia will not see the interactive ingredients targeted to the lazy eye, unless we radically reduce the level of contrast of the strong eye. Based on the clinical protocol developed by McGill University, we implemented a system allowing a progressive contrast change throughout the therapy. This is how the brain progressively reaccepts the vision stimulus coming from the lazy eye, until the contrast level is similar to both eyes.

Also, we had to integrate triggers into the game to identify if the contrast level progression was appropriate for the patient on a daily basis. The game is smart and adapts itself to the condition of the patient, after the doctor's initial calibration based on the individual diagnostic.

Making Games Was there any specific reason why you decided to use mobile platforms for the game instead e.g. a PC?
Mathieu Ferland We need to think of this as being a medical device. This is not a game

Matthieu Ferland joined Ubisoft Montreal in 1997 and was amongst others involved in the creation of »Splinter Cell«. Later on he took over new responsibilities and is now in charge the development of creative and technological solutions.

that will be downloaded from the App Store or Google Play. The game will be embedded into the tablets, and the tablets provided to doctors, for their patients. With this mindset, even if the install base of a specific tablet is not an issue, it is important to use a device that most users will be familiar with. Phones and tablets are now part of our daily life and almost everybody knows how to use them, or can quickly learn. This means the patient adapts to the therapy more easily.

Making Games When talking about technical requirements: Are there specific requirements for the tablet pcs or smartphones that are used?
Mathieu Ferland Bigger sized devices may be easier to read for some patients with amblyopia. This is why we recommend the use of a tablet (vs. another mobile option). We have not seen resolution restrictions with common tablets yet; it's more a matter of comfort. The available tablets will be selected, purchased and preset by Amblyotech. The therapeutic treatment will be pre-installed on these tablets and distributed to clinics by Amblyotech.

Making Games What are the next steps for »Dig Rush« and what has to be done before its final release? How will the treatment work once authorities have approved the game?
Mathieu Ferland We are still working on the game's calibration to make sure the difficulty level is accessible because the game needs to adapt to any type of gamer. We are also working on the physician interface based on the feedback we are receiving from current clinical tests. The next step is to proceed to regulatory approvals with authorities in the US and Canada. Then we'll target Europe. Amblyotech is also in charge of distribution, meaning that they will seek out clinics and eye specialists for all territories.

Making Games You said that the game has to adapt to any type of gamer. How do you approach the balancing of a game because I think that the fun and success of therapy are strongly connected?
Mathieu Ferland This is where Ubisoft's expertise brings value to the solution. There is no unique recipe to simply apply our »easy pick up / hard to master«-philosophy. Prototypes need to be tested and this is always an iterative process. Our experienced game design team's challenge is to create fun and simple game mechanics that have a high potential for depth when layered with new gameplay ingredients. Another interesting fact is that we are using many classical »free to play« techniques such as retention, tracking, etc. but not for the purpose of monetization. In this game all these engagement tools are meant to maintain high motivation in the user, and ensure they play for enough time.

Making Games Are there specific reasons that »Dig Rush« is a kind of puzzle / jump 'n' run game or could any kind of game work for a therapy focused game like this?
Mathieu Ferland Our first interest was to find a game concept that meets medical and clinical requirements, and the »Dig Rush«-concept works in those respects. I believe that we could find other game concepts that treat players efficiently, but our priority is now to finalize the game and to make it available. Then we may investigate potential future improvements or ways to vary the game treatment.

Making Games Talking about calibration of the game itself. You said that a doctor has to calibrate the game before a patient can start playing and by that start the therapy. Do you already know what kind of effects it may have if this calibration is done wrong or not accurate enough for example?
Mathieu Ferland A diagnostic is the starting point for this therapy because the prognostic may not be similar for all patients. »Dig Rush« is a new tool for physicians to treat amblyopia, but it may not be the first step for some patients, depending on multiple factors. This is why each patient has to get their own prognostic and then follow a personalized course of treatment. Once the patient is ready to play »Dig Rush«, the calibration is important to make an appropriate display alignment, to select the right 3D glasses (depending if the patient's lazy eye is the right or left eye), and to optimize the contrast level. A calibration that is not done properly will affect the efficacy of the therapy.

Making Games Can you share any insights in the effectiveness of the treatment with »Dig Rush« compared to traditional treatments?

»We need to think of this as being a medical device.«

Mathieu Ferland Studies on children and adults have shown significant improvement in all visual functions following the new treatment. The results from these studies show 1.5 to 2 line improvement in visual acuity following just four weeks (with a playtime of about four hours per week) of playing the treatment games. In contrast, with other treatment options like patching it takes up to four months, to be able to see treatment effects. Moreover, other conventional treatments are able to bring improvements in only visual acuity, while having no effect on other important visual functions like 3D vision (stereopsis). This new treatment has shown significant improvements in stereopsis, better co-ordination between the two eyes and even reduction in suppression of the brain level. Furthermore, unlike other treatments that result in recurrence of amblyopia, the improvements seen following the therapy were maintained even three months after ending treatment.

This new therapy promises to provide rapid and stable improvement in all visual functions and thereby provide a better quality of life.

Making Games What do you think why »Dig Rush« is even a successful treatment for adults when traditional treatments don't work and what are the requirements that a patient can use the game?

Mathieu Ferland The current treatment relies on wearing an eye patch to cover the dominant eye, in order for the lazy eye to catch up. However, this technique doesn't lead to positive results for adults who have reached their biological development maturity. The main difference with the game is the fact that instead of training the weak eye only, »Dig Rush« trains the brain to use both eyes. Focusing on the reconditioning of the brain is a method that works for all age groups, and shows great results for adults, which is a major breakthrough.

> »This new therapy promises to provide rapid and stable improvement in all visual functions.«

Making Games Beside of »Dig Rush« and amblyopia, are there any other fields that you think that games might be able to help patients to get better?

Mathieu Ferland Hopefully yes. As consumers, we are always expecting more and more. I believe that sooner or later, the patient will be at the center of many medical treatments, and that interactive experiences will be the best vehicle to convey such a purpose.

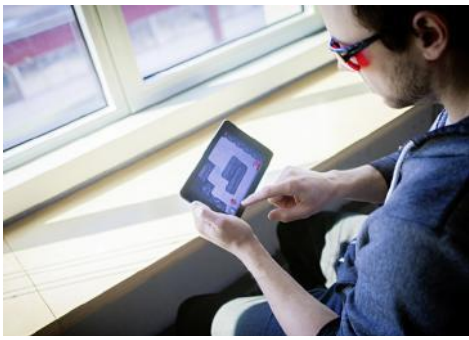
Making Games But there are no specific illnesses that Ubisoft or you personally can think of treating with games?

Mathieu Ferland Time will tell. It's hard to envision the future on topics as complex as life science. More collaborations of this type with researchers could lead us to new opportunities.

Making Games Do you think that Ubisoft might do research in this kind of »medical games« in the future or is »Dig Rush« yet only an experiment for you?

Mathieu Ferland What's important for us is to focus our expertise and to create great engaging experiences for players. Ubisoft is always looking for new ways to innovate and try new things. We're not focusing particularly on games for medical purposes, but we're open to considering other opportunities.

Interview: Sebastian Weber



Each patient has to wear special glasses like 3D glasses that have to be adjusted to the specific needs for his treatment. Doctors will have a special interface to set up the game for every single patient individually.

USER-GENERATED CONTENT IN THE V-PLAY GAME ENGINE

The era of mobile gaming and F2P has raised some new challenges. How do we acquire new players with so many other games out there? How do we make these players return? And how do we turn them into fans? Christian Feldbacher explains how you can increase downloads and player retention with user-generated content powered by the V-Play Game Engine.

Christian Feldbacher is CEO & Co-Founder at V-Play.

Christian has more than 15 years of experience in software development with a strong focus on game and mobile app development for all kinds of platforms including iOS, Android, Windows Phone, native Symbian, Java ME, MeeGo, and BlackBerry. His passion for games and experience in Qt, game, and mobile development led to the creation of V-Play Game Engine. He is speaker of international conferences like GDC or Qt Developer Days, loves game jams and teaches university students how to develop games.

Once you've published a mobile Free2Play game in the app stores, the real work begins. Because the key to having a successful game is to constantly improve its retention metrics and revenues per player. If you get these metrics right, investing in user acquisition becomes a driver of growth – and combined with social elements in your game – gives rise to huge, previously unexplored business opportunities.

The tricky question is: HOW can you get players to return to your game and how can you reach a social level where players are telling their friends to download this cool, new game – your game?

V-Play Game Engine

With V-Play Game Engine, we created a tool to solve these challenges. V-Play is a cross-platform engine specialized for mobile 2D and 2.5D games, currently used by more than 10,000 developers. It is built for rapid game development – not only during production up until the point the game is released, but also afterwards, by allowing user-generated content in games of all genres. But first, let's start with current game production challenges.

In-Game editor for fast balancing

The challenges of game development start early in production: Creating new levels and balancing the gameplay are often the trickiest things to get right. Most studios create their own tools and editors to allow fast iterations. Ideally, content creation can then work without the need for programmers, instead being done by level designers and game designers. Therefore you could argue that an ideal level editor is simple to use and tailored for your specific type of game.

It's for exactly this reason that we have built editor components that give you the ability to create and balance content rapidly, independent of the game genre. So one specialty is you can create levels for platformers, sidescrollers, Match-3-games or any other 2D or 2.5D game with the same set of editor components.

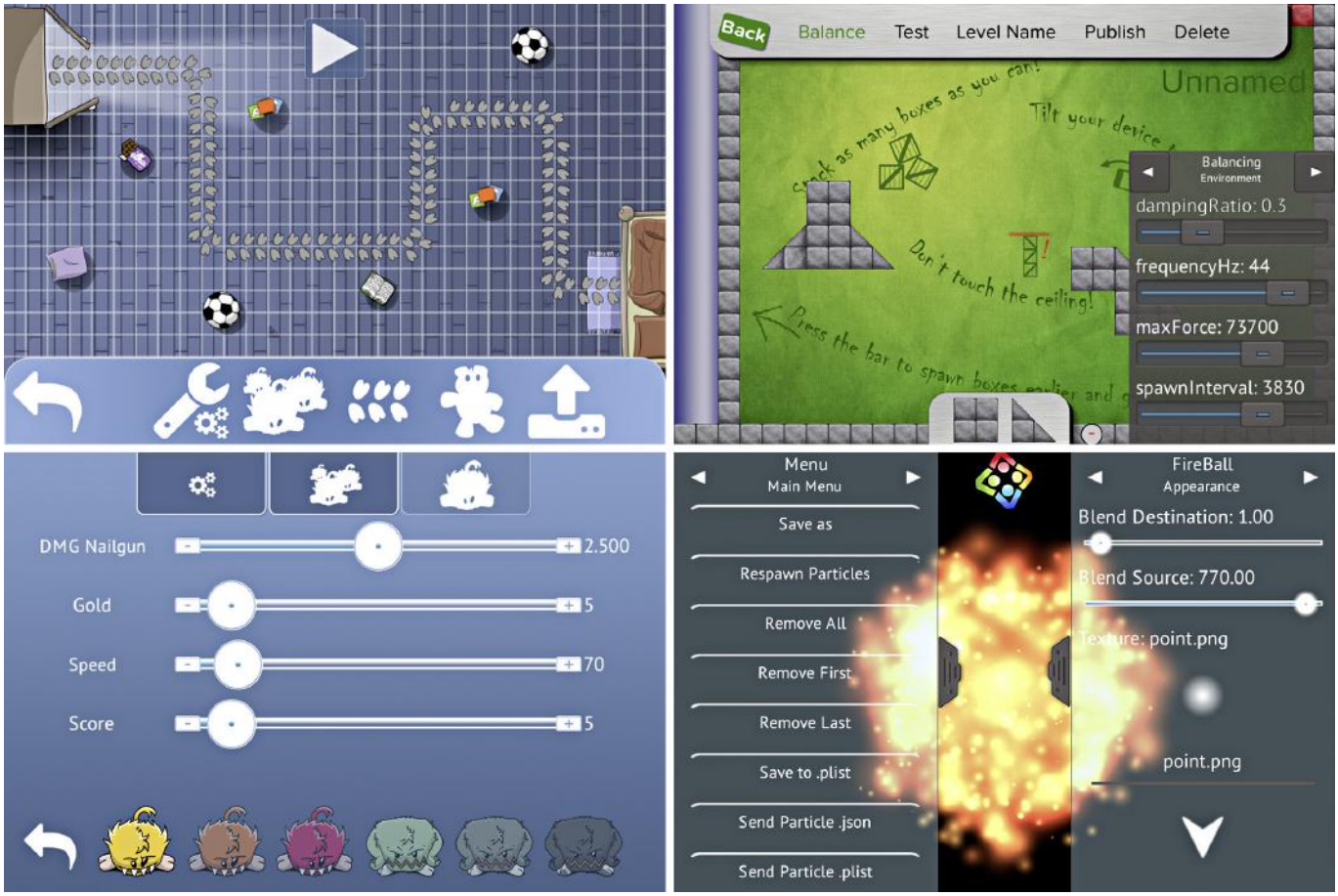
The second specialty is that this content creation and balancing works while the game is running. This in-game editor has major benefits compared with traditional »offline« editors:

- You see balancing changes instantly, even on mobile devices: for example change the friction value and see how the physics bodies react, or change the accelerometer multiplier to test the ideal input settings on your mobile device. This saves time because you don't need to constantly stop, tweak, restart and retest the game. Instead just test it all in one go.
- The saving, loading and exporting of levels are automatically handled for you.
- You can easily customize the look of the in-game level editor to best match your game style, or use the default V-Play skin for quick results.

Let's dive a little deeper into how the in-game editor works in practice. First, you define the properties that you want to balance at runtime. As V-Play is component-based, you simply add an »EditableComponent« as a child of the component you want to balance. For example if you want to modify the properties of a physics component like density, friction

The series at a glance

- Part 1
  User-generated content in the V-Play Game Engine
- Part 2
  Rapid Game Development with V-Play



Three games of different genres, all using the same V-Play components that allow in-game level creation and balancing. The UI of the editor components is customized to match the game's style.

or restitution, just add these to the editable properties list.

All that's then left to do is to define where the UI for the property modification shall be placed in your Scene. The UI component called »ItemEditor« searches in all components for EditableComponent instances and adds a property modifier for them. If the property value is a number, it automatically creates a slider with (customizable) bounds. If it is a color, it creates a color picker and if it is a string it shows a text input field when clicking on it, etc.

Listing 1 shows the full source code of a game with an in-game editor that allows defining the physics properties at runtime. The full application only has 30 lines of code and 400 characters – that's why development with V-Play is called rapid.

Customize your editor

The V-Play system detects the basic type of the balancing property and shows a UI property modifier for it. To allow modification of custom array types, you can then create your own UI modifiers. For example, we used that to change the settings for each wave in »Squaby«, a tower defense game made with V-Play. In this game each wave has a certain amount of monsters that it spawns, a delay between these monsters, and a pause between the next waves. It also contains the probability of how likely it is that it will spawn different monster types.

For the best user experience, we recommend styling the property modifiers so the UI matches your game style. You can either replace the graphics – of, for example, the sliders – with your own, or just change the colors of the existing ones. However, if you only use the editor for internal content creation during development, you can just use the default skin provided by V-Play.

Placing entities

What else makes a level in games except balancing properties? The position of game entities in the level. V-Play provides components that allow you to Drag & Drop different entity types into the level. After adding an entity to the world, you can change its position or rotation, or remove it if you don't want it in the level anymore.

Just as with the balancing UI, the appearance of the »BuildEntityButton« can be customized. In addition you can use the built-in support for snapping items to a grid and you can use the entity's bounding box to make sure elements do not overlap.

The final piece in the jigsaw for any good level editor is the ability to manage your current level settings. A level is the combination of the entities' positions in the world plus the balancing settings for the entities and the game world. There are ready-made functions available in V-Play for saving and loading all these settings. When you are happy with the results and want to ship the level with your game you can export a JSON file and place it into your game's binary.

The company

Vienna-based V-Play GmbH, the creators of the V-Play Game Engine, was founded in 2012. The cross-platform engine was designed as a rapid game development framework for mobile 2D & 2.5D games. More than 10,000 developers are using V-Play to create games with the in-game editor components that allow user-generated content. For more resources on user-generated content like a step-by-step tutorial on how to add an in-game editor and a LevelStore to your game, see **www.v-play.net/editor**.

GAME ENGINE V-PLAY

```
import VPlay 2.0
import QtQuick 2.0

GameWindow {

 Scene {

  ItemEditor {
   // position the in-game editor at the bottom right of the Scene
   anchors.right: parent.right
   anchors.bottom: parent.bottom
  }


  PhysicsWorld {
   gravity.y: -10 // makes the box fall down
  }

  // game entity base component, with built-in level saving & loading support
  EntityBase {
   entityType: "Box"

   Image {
    source: "box.png"
    // property bindings:
    // when the collider width changes, this width property changes too
    width: collider.width
    height: collider.height
   }

   // a Box2D body with a rectangular box fixture
   BoxCollider {
    id: collider
    width: 32
    height: 32

    EditableComponent {
     properties: {

      // these properties can be changed at runtime
      "density": {"min": 200, "max": 1000},
      "friction": {"min": 0, "max": 100},
      "restitution": {"min": 0, "max": 200}
     }
    }// EditableComponent

   } // BoxCollider

  }// EntityBase
 }// Scene
}// GameWindow
```

Listing 1: Full source code of a game with a box-shaped physics body and an in-game editor to change the density, friction and restitution at runtime.

In-game editor on the bottom right for the physics properties of the box entity from Listing 1. The component that generates the UI is called »ItemEditor«.

Take your game to the next level with user-generated content

As you now already have a content creation pipeline that works in-game, you can easily allow your players to create content, too! By enabling your players to add user-generated content, you'll receive the following benefits:

- Players return to your game more often because there's a never-ending stream of content and always something new to explore.
- Your game downloads will increase due to the word-of-mouth marketing of user-generated levels created by your players.
- There's no need to create constant level updates to keep your game interesting after you've published it – the community does this for you!
- Earn money with user-generated levels by adding a level store!

What's more, it's also beneficial to your players:

- Players are more engaged because they can create their own levels and tell their friends about it. You can reward them for great content and further motivate them to build levels!
- Players get an endless stream of new levels.

Sounds like a win-win situation, right? Well, it is! There's a huge potential for user-generated content in mobile games that currently remains unexplored. With V-Play, you can create a game powering UGC rapidly, as publishing a level is just a single API call with the in-game editor.

To make the most out of your player community and UGC, V-Play allows you to:

- Allow players to rank levels to create a level quality rating determined by the community.
- Order player levels based on download numbers, level rating, or by date to find the newest levels.
- Filter the levels to only show levels made by friends.
- Add a leaderboard for each user-generated level, so that players can compete with high scores in infinite user-generated levels.
- Reward players for good levels.

The last point is a very effective one in motivating your players to create good content! With V-Play, you can reward players when their published levels get high ratings or when they get a lot of downloads. In this way you can add a competitive element to the level publishing process: list the best level creators and feature them in your game.

To motivate players to download levels, you can also reward them for downloading new levels or for submitting a rating. These rewards can be some kind of achievements. Or why not go one step further and reward them with in-game currency?

Hot to monetize user-generated content

Rewarding players with virtual currency opens up a wealth of new opportunities: you can now set a price for downloading a level. Now those of your players who make great levels will be able to download the levels of other players without investing real money. However, those players who do not want to create new levels but still enjoy playing them can continue doing so by purchasing virtual currency that can then be used for level downloads.

The V-Play component »LevelStore« wraps all this functionality and combines it with a cross-platform in-app purchase plugin. It also ensures that once the player has downloaded a level, that it's playable offline. If your player is using multiple devices, it'll also ensure that all the purchased levels are synchronized across platforms with the V-Play Game Network cloud sync components.

You can also add level sharing without any purchases, although you could then be missing out on a valuable business opportunity. Also note that motivation for creating good levels is higher for players when they get a reward for it (in the form of currency or achievements.)

You can also view this concept as an app store for levels – your game contains an own ecosystem of downloadable levels that can be purchased and rated. It's just like Apple's or Google's app store, but with user-generated levels instead of apps.

In essence, we have built a back-end for all of this level ordering, storing and analytics functionality. Plus the game engine editor components as front-end that you can use in your games. The V-Play components are communicating with the server backend, so you are all set with just a couple of method calls on the client side.

Try live games and see the source code

The screenshots featured in this article have all been made with live games in the app stores called »Squaby« and »Stack and Friends«. To help you get started quickly you will receive the full source code of these games using the in-game editor and user-generated content when downloading the V-Play SDK.

There are few other mobile games that leverage the power of the community and UGC like »Createria«, »Geometry Dash« or »Bike Baron«. And of course »Minecraft« and »LittleBigPlanet«, the pioneers of creating player communities with user-generated content. Even Nintendo is heading towards this direction with the announcement of »Mario Maker« – a Super Mario game that comes complete with an in-game editor and UGC support.

The good news is that you don't need to have the engineering power of Nintendo to create a game with UGC on your own – you can just use V-Play! There are so many untapped opportunities in mobile games, regardless of genre. For example, why not allow your users to customize the color scheme and let them share it with their friends and the game community? Or let them create their own levels for a Match 3 game? I firmly believe that the future of mobile gaming lies in involving the players in this way, and in doing so, further fostering the creation and growth of gamer communities.

Recap

We've seen how to integrate user-generated content and an in-game editor into any kind of game. The main benefits are that you get more downloads, more active players, higher player retention and you can leverage the creativity of your passionate players.

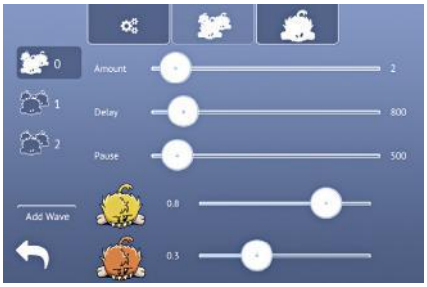
The main reason why UGC is not yet being used in mobile games is simply because building the client and server logic remains a huge stumbling block. It took us more than a year to have all the editor components ready and to make them available independent of the game genre. The main challenge for us was to define the API in a way that makes it easy to integrate and add UGC into your games in just a matter of days.

Even if you don't use the level sharing functionality, using an in-game editor has big advantages over offline editor tools: much shorter content creation time leads to better games, as balancing and level creation work on the fly on all platforms, even on tablets and phones.

What's Next?

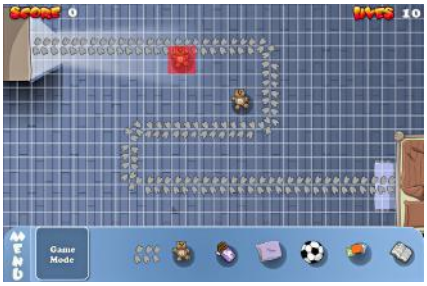
In the upcoming second part of this series, we'll explain how game development with the V-Play Game Engine works and why the Qt framework was chosen as the underlying C++ core. It will also cover which kinds of games you can make with V-Play, and which other features help in rapid mobile game development besides the in-game content creation tools.

Christian Feldbacher

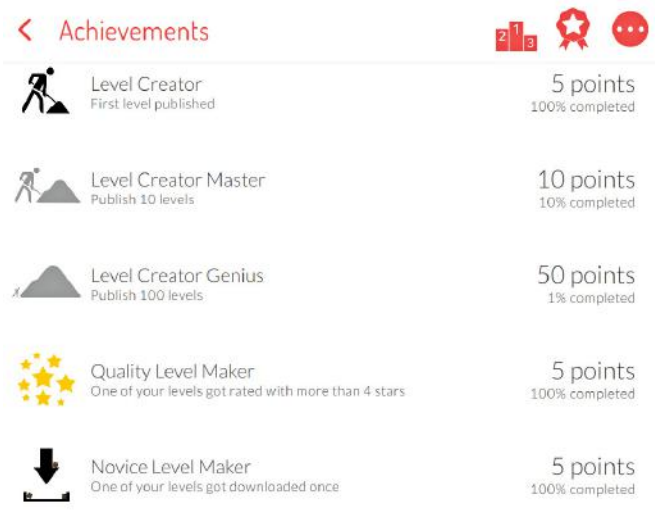


Wave settings for the tower defense game »Squaby«. You can create your own UI modifiers for any data type like arrays. For the best user experience, you can style the in-game editor UI to match your game style.

In the V-Play game »Stack And Friends«, players can create their own levels by drag-and-drop different entity types (on the bottom), change their positions and rotate them.



With the »BuildEntityButton« component, you can drag-and-drop different entities into the game. The UI of the button is fully customizable to match your game style.



Reward players with achievements and in-game currency for creating good levels. These credits can then be used to download new levels.

TEACHING TOLERANCE AND CIVIL COURAGE THROUGH GAME AND FILM

A mobile serious game accompanied by a short film is meant to engage young adults with Holocaust Remembrance in a modern relevant way. It is a challenging, yet exciting undertaking, transforming history into a multiplatform experience especially when the makers first need to find a common language.

Michael Geidel
is producer
at MiriquidiFilm.

Michael Geidel is a film and interactive producer with a finance and distribution background and a strong technology focus, working over 12 years in the industry. He also spend some time in research and teaching at universities in the last years, is member of film festival juries, researches in sustainability in film, is Berlinale Talents Expert and alumni of The Multi-Platform Business School and The Pixel Lab.

Annekathrin Wetzel
is director, screenwriter,
and producer
at MiriquidiFilm.

Annekathrin Wetzel is a writer, director, producer and the initiator of the project. She has 15 years of experience working for TV and digital media. She focuses on multiplatform story architecture, combining fiction film, documentary, game, music/audio and social media. She has been an advisor for the use of gamification at the German State Channel (ARD-Alpha and BR).

Using games for educational purposes has a long tradition, as learning through play is how we all grew up. And serious games become more attractive than ever before and have gone beyond the stereotype of focusing on being solely educational with no fun attached. Still, creating a game around the theme of Holocaust Remembrance that is both fun to play and cleverly educates is a challenging subject. Within the story world of our game »Best friends« belonging to a Holocaust Remembrance project, we needed to think outside of the box to be able to create an engaging game that connects with a young audience today.

Why work on a game about Holocaust Remembrance?

First of all: Holocaust Remembrance is not just about Jewish people. It's about this kind of behavior towards the Jews. And Holocaust Remembrance is to remind people of today not to behave like that towards ALL people. It's about tolerance and civil courage.

A good place to start is to educate young people in school. In fact, politicians as well as educational and religious leaders demand new projects that educate young people about tolerance and Holocaust Remembrance. In the near future there won't be any survivors of the Holocaust alive anymore. So we have to come up with new forms of transporting their memories and experiences. There are tons of books, films, documentaries, and TV-programs, but we haven't seen a real GAME on this.

O.k., one could say that young people learn about World War 2 from games like »Call of Duty«. But we believe it is time to find other games for learning from history. There is a demand for that. And also in the history of the game industry, it's time to embark on this.

How did we approach the development of a game like this?

First we defined: What is the aim of the game? What do we want to do? We want people to engage with the theme and to be touched and therefore defined three core elements to sum up our work:

- What do we need for this? A Story.
- We need to find the matching technology.
- And the suitable game mechanics

As those three aspects work together and influence each other, there has been a back and forward between these 3 elements. As this is going to be one of the first games around Holocaust Remembrance we are aiming high. We want it to be:

- Enjoyable and fun to play
- Relevant for today
- Easy to use
- Making an impact in schools and also in museums

Finding the Story

We are content makers, creative storytellers and our initial background is filmmaking. So we started out with a strong story of a Holocaust survivor based on a true story. We made a short film in 3D, called »Call her Lotte!«.

It's about two best friends, the Jewish girl Lea and non-Jewish Maria living under Hitler's dictatorship in Nazi-Germany. The non-Jewish girl Maria falls in love with a young policeman, Hans, who then makes a career as an SS-officer. Hans soon demands to stop contact with her Jewish friend. This puts Maria in a strong dilemma situation. On the infamous day of the first deportation of the Jews from Munich, Maria has

The game that is meant to be used in school comes together with a short film. Having a film ready helped a lot when it came to convince partners about the project.

to decide: Is she going to help her best friend Lea? Or will she obey Hans and ultimately the Nazi-Regime? In the end, Maria can't save Lea, but she takes her child Charlotte up, risks her own life by doing so, and separates from Hans.

The childhood of Charlotte Knobloch, the former president of the Central Council of Jews in Germany, inspired us to make the film. Mrs. Knobloch is a Holocaust survivor and is one of the leading people demanding new products for young people to engage with the topic of Holocaust Remembrance.

The film shows this part of German history historically correct through this story and it functions to touch the emotions of the audience. Therefore the film project was invited to the Festival de Cannes and over 35 more international festivals and received 13 awards so far. The length of 18 minutes is perfect for educational purposes, because it fits into a school lesson of up to 45 minutes and teachers have time enough to have a discussion afterwards.

Starting from this moving true story, we began developing ideas for the game in addition to the film.

Trials and errors

We started out thinking from the »story«-perspective and that we should stick close to the original characters and the story of the film. The first ideas evolved around the players experiencing the world of the film's characters. We aimed for a role playing game, which is very common in educational games, for example about refugees living in challenging conditions.

To intensify the experience, we wanted the players to go to the real locations where history actually happened to show: This is not fiction, but a real story. So we wanted to make a location based game. For example: one day in the life of Lea, Maria and Hans.

The technology agency »Vectorform«, based in Germany and the U.S., offered us to make a location based game with latest technology. In 2013 Google Glass was the first hands free device, which could be used like a smartphone. And this agency was one of few companies in Germany having a Google Glass and also having experience to program apps for it.

We developed ideas for going through Munich with Google Glass and at the original locations the players would receive tasks and information. But soon we discovered, if you can only play the game in Munich, this is not going to reach a large audience, which was one of our objectives.

Now we were pulling our hairs, working for months to find a solution: How to adapt it to other cities like Berlin, London, Paris, San Francisco or Los Angeles? We found the following solution: Every major city has a theatre, a synagogue, and a town hall. We would make those sites archetypal spots that everyone can relate to. But in the end by testing it, it became clear that these were all compromises, because, if it didn't really happen there, it's not authentic. We missed the point!

On top, we discovered after first tests that Google Glass wasn't ideal for what we wanted. It had a very small screen, was not easy to handle, came with a short battery life, heated up quickly – just to name a few. Although we have to admit, the pupils loved playing with it. In the end, we we realized that just because you want to use the latest or innovative technology that doesn't mean that you get a very good game.

Going back to the objectives

We decided to make a cut and looked at our initial aims: relevance, fun, easy to use. Also we discovered that sticking to historical facts and places is not making it relevant enough for today and the world the young people live in 2015.

So how could we make the game relevant for today? What would engage young adults in 2015? We came up with the idea to take the themes of the film and re-create the emotions of the characters instead of just having the player re-experiencing the film plot. By this it would get more universal as well, looking at the underlying universal principles of behavior and reactions towards certain people and groups that lead to intolerance and Anti-Semitism. By not relating directly to history the players would learn in their environment of today and at the same time gain understanding about history, too. We had to find a way that the players themselves engage with the game deeply.

Fun and relevance are closely linked and go hand in hand. So after knowing how to make it relevant, we had to find a way to make it fun? In order to be enjoyable, the game has to be compelling and to create involvement of the players. There are games out there which do that very well. But for serious games it's really hard. We

Who is Charlotte Knobloch?

Prof. Dr. h.c. Charlotte Knobloch, President of the Jewish community of Munich and upper Bavaria, former President of the Central Council of Jews in Germany and former Vice-President of the World Jewish Congress, survived the Holocaust because a german lady took her up. Presented as an illegitimate child, Charlotte was hidden by the catholic Zenzi Hummel on a farm from 1942 on until the end of the war. Charlotte Knobloch has made a major contribution to the reconsiliation process of Jews and Non-Jews in Germany after the Holocaust. She especially encouraged young people to have a healthy pride for their country, to not feel guilty and at the same time learn from the past to form a better future. She was honored for her outstanding contribution for tolerance, reconsilitation and strengthening democratic values becoming Honorary Citizen of Munich in 2005. Charlotte Knobloch supports the project.

»I encourage every smart and responsible project that speaks to the young people and reaches them where they are: in the internet, through games, in the cinema, through television or at school. Multiplatform projects are the future. And of course, it should not lack fun.«

The very first concept for the game centered around Google Glass and was meant to be location based. Macromedia University for Media and Communication in Munich offered to create a user experience for it.

While the target audience liked playing around with this gadget, it quickly became clear that the newest tech is not the best way to go for - simply as schools not necessarily have things like those at hand.

knew that. Interestingly, most serious games where educators were deeply involved don't work well with the audience, we were told. A moving and thereby learning experience happens if you are personally affected and are emotionally attached. By linking the relevant questions to the players and using classical game mechanics the players would immerse into a world of challenges – and these would be about questions like: How does it happen, that you include someone into your peer group, but exclude and reject another one? How does discrimination start? Would you sacrifice a friendship in order to keep your status in society? How far would you go for your best friend, if he or she would be declared as an enemy?

We found a good example from the 1970s that inspired us and confirmed our new approach: The brown eyes, blue eyes experiment in a 3rd grade school class: »A Class Divided« (you can find a documentary about it on YouTube).

Getting the game playful and educational is one challenge, but who decides if the game is played? It's the teachers! Kids should like it, but teachers need to be able to handle and operate it. They'll only use it if it makes life easier and not more complicated for them.

How to make the game easy for teachers to operate?

We did some research on what the environment and the needs of teachers are. They have little time. They look for things, which aid them, which are easy to use, and which engage their students. Also, we have to be aware which facilities schools have. We don't want the game to have a high entry barrier.

Today most schools are not yet set up for having tablets or smartphones for everyone in class. But they are more likely to have smartphones and tablets than Google Glasses or Oculus Rifts. We can't serve everyone, but in order to reach our aims we had to look into the future of the digital classroom becoming a mainstream principle of teaching. At the same time a non-digital version is possible in addition to the digital one. Also smartphones can be used for VR Experiences nowadays, so we had lots of possibilities still.

Working in an interdisciplinary team

Another question that came up: How can we find game developers to cooperate with? How do we know, who would be the right person for this game? And how should we communicate with them?

We started asking around, we went to game designers that were recommended to us or that we met at parties, through friends, at meetings, and conferences. At the same time we asked universities that teach game design. We showed them the film and explained, what the game narrative should be and what functions it should have.

This lead to the following collaborations: Macromedia University for Media and Communication in Munich offered to create a user experience with a class of students for the location based game using Google Glass. From the university in Erfurt, Prof. Rolf Kruse from the department of digital media and design started creating ideas with game designer Kelvin Autenrieth for a class room mode game using »Mixed Media« with analog and digital devices combined. But the first sketches and ideas didn't really match our objectives in terms of storytelling. And without additional finance we couldn't go further.

The frustration of failure and how to overcome it

At that time, we were often frustrated, because we had the feeling that the game developers didn't understand us. But the same was true vice versa. They later told us they were thinking: »What the heck are they talking about?«.

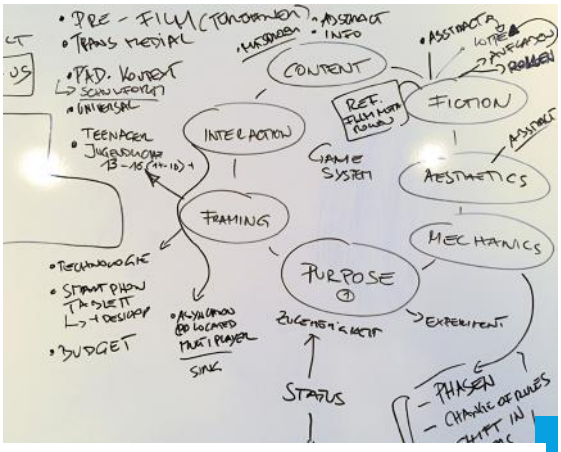
Looking back, we now understand that communication is not just talking to each other. It needs more! It needs time! Sitting down, spending time with each other and really check, how the other person did understand your arguments. One thing that helps immensely: doing a grill party.

We as »transmedia people« really had to learn how to fail. For a film you have a treatment and you can pretty much see if this has the potential for a good film. But we couldn't do that with game ideas. We couldn't evaluate, if the game ideas would work or not. In a game there is much more work involved before you know if your idea will work or not, and lots of things change on the way for sure.

So what was the solution for this? Well, we didn't give up! We decided to get every stakeholder of the project to a table and lock them up for two days in a Game Lab. We invited:

- Dr. Konstantin Mitgutsch, a game coach as facilitator, and a Research Affiliate at the MIT GAME LAB at the Massachusetts Institute of Technology (MIT) in Boston
- Dominik Philp, a teenage game player – our target group
- Jaron Schulz, from NUTS, a game designer, who personally relates to the topic living as a Jew in Germany
- Dr. Martin Ganguly, an expert for media in the educational system, who knows what teachers in schools want and need
- Adam Sigel, a U.S. writer and content strategist, to get the international perspective
- And us as producers and content creators.

We conducted two days of brainstorming using the following system: It's called »Serious Game Design – Assessment Frame Work« which Dr. Konstantin Mitgutsch developed at MIT to evaluate serious games and to create better new ones. The main focus of this »framework« is the exploration, how different



As the key people behind the project are coming from the movie industry in the end they gathered a lot of experts in a so called game lab to settle down on a final concept. This took some days but paid off in the end.

elements of a serious game are related to each other and how the purpose of the game can be channeled in all aspects of the game. This is how the frame work (see image page 32) works and builds upon each other:

1. **THE PURPOSE:** First you define what is the purpose of the game? What do you want to reach with this game? What should the impact be? What would be a success? Why is this purpose relevant for the players?
2. **FRAMEING – TARGET GROUP AND SETTING:** Who is the target group? How would you know that you reached it? What is the context the game will be played in? What form of engagement are we going for? How does it fit to the target group's behavior?
3. **FORMS OF INTERACTION:** You explore how the purpose of the game can reach the audience in forms of interaction. Like: multiplayer vs. singleplayer? What platform should be used? Even: Why does the game need to be digital? What interaction experience should the players take away from playing it? What form of communication does the player do in the game? Also: How long should they play the game?
4. **CONTENT AND INFORMATION:** What should be the core content of the game? Should it be abstract or realistic? What is the story of the game? Which information should the players take away from the game? What is specific about the narration of the game? What characters will be used?
5. **MECHANICS:** Learning curve, rules, goals, rewards, and verbs. What are the players actually »doing« in the game? How do these work towards the purpose? What is happening at which time in the game?
6. **FICTION/NARRATIVE:** It's all about which characters are in the game and what plot is laid out. Of course this all goes very much together with the before mentioned aspects.
7. **AESTHETICS:** Last but not least for sure, the setting and the visualization of the game are an important factor that influence the emotional journey of the players deeply.

The results from the lab

The core experiences of the games should be the emotions of »love, fear, loss, courage, victory, and remorse«. The core message the game should offer as an experience is: »Would you sacrifice US to belong to THEM or would you sacrifice THEM to belong to US?«

These are some core ideas that we want to develop further:

- The game will be a multiplayer experience.
- It will be played in a closed group of players (e.g. a school class with 10 to 30 players).
- The game will be played for a restricted time, like a period of one week.
- It will be framed by a facilitator giving an introduction and a discussion afterwards about the play experience (for example a teacher).
- The game will be played outside the classroom (communication among the players about the game should be a positive side effect).
- The game should be mobile and be playable on tablets and smartphones. At the same time we are looking for mechanics that get the players up and going beyond a mere computer experience, if possible they should be able to interact.
- A change of rules and shift in powers as core mechanics of a moral dilemma meta-system.

All together: The game is to confront the players with group dynamics like inclusion, exclusion, belonging to a group, being rejected by a group, scarifying friendships or loosing status under changing conditions – which means under a change of rules in the game.

By playing the game the players get to feel the conditions and situations the young characters in the movie »Call her Lotte« experience – within their own context of every-day life and friendship.

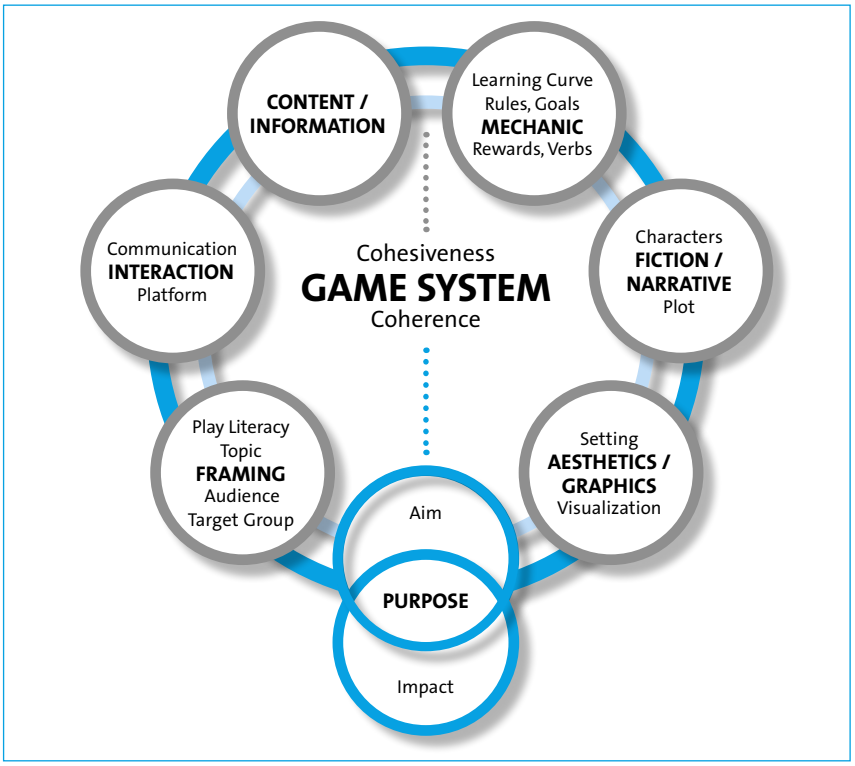
We also came up with some first explorations for the user experience:

- A teacher introduces the game on a Friday and the players sign up for the game within class.
- The game starts with the »Check in« in which the players have to answer questions about themselves. The system generates a »passport« according to the answers which defines to which group each player belongs. All players start from the same level.
- The information about their status, their attributes like education, looks, wealth, health or disabilities, and beliefs are now in the

Summary

The benefits of a multiplatform project like this are:

- You reach more people.
- With a film as your background, you always have something at hand to show before the game is out, which helps in terms of financing and getting partners.
- To make a game both engaging, fun to play and at the same time educational, you have to be ready to kill your darlings, always focus on the core of the game and don't be tempted by technology for the sake of technology.
- To work best in an interdisciplinary team, you should get together and lock yourself up for a couple of days in the beginning.
- We realized that teachers and politicians are becoming more open to using games ...
- ... and using games for educational purposes makes financing and international distribution easier.



This chart shows the »Serious Game Design - Assessment Frame Work«, which Dr. Konstantin Mitgutsch developed at MIT GAME LAB at the Massachusetts Institute of Technology (MIT) in Boston to evaluate serious games and to create better new ones.

settings. Those will have an impact once the change of rules is activated.

- Each player is visualized through an »avatar«.
- The avatar shows clear signs of gender, status and all the attributes (for example, if a player gains wealth, this is represented on his appearance).
- The players collect points by following tasks, solving challenges, liking or disliking trends and slogans of group-members, or by giving out tasks and slogans, which others then follow – it's about gaining popularity, status and finally power.
- One rule of the game is that you need to have friends to win. But in order to win friends and maintain friendships, the player has to spend points. It's a strategy game, therefore each player faces the challenge that he needs the most points and still needs the support from his friends. Otherwise he or she can't win.
- Once two players agree to be friends, they can interact with each other within entertaining and asynchronous multiplayer mini games to gain points and built up a relationship thereby – but only until the change of rules puts the relationship to a test like in the film.
- Later in the game the tasks get more difficult. The demand to sacrifice certain people and also friends in order to gain status increases.
- At the same time, best friends can exchange secret messages and access secret information. These players are the »underground movement«: they are in the group with the lowest status (like in the Ghettos) – they can't move up again, but they can disturb other players and groups, and to create campaigns.
- At the end of the day each player gets to vote »IN or OUT« which players stay in his group.
- The longer you are maintaining a friendship with someone, the more information you get about this person. You can collect points and raise status by revealing information about your friend to the group. If your friend is from another group the player can reveal even more! This has an impact on the person, being voted »IN or OUT«.
- At a certain point, each friendship is put to a DECISION: If you want to stay in the group, please cancel your friendship to X (e.g. if this player is in a lower group or has certain attributes, which are denied by your group).
- At the end of the week the results of the play experience lead over to a group discussion and then they can watch the film.

All the mechanics we use in the game are like an analogy for the principles that were used in the Nazi-system to establish their power, to brainwash and suppress people, and for how people reacted.

How to finance and market a game like this

The advantage is, we are creating a game for a clearly defined target group of young adults. The demand by teachers and the society is obvious and statistics show that our target group is very much into playing games. Still, it has to be a serious game. But since there are not many serious games used in schools so far and by dealing with the topic of Holocaust Remembrance and our approach to it, we see a big advantage for our project.

Distribution is very different on the one hand to a classical boxed product or to online distribution. The game is not intended to appear at GameStop or on Steam. The film on the other hand has a distributor for the educational sector in Germany and German speaking educational organizations worldwide. They will also distribute the game to our target group and do the marketing, so that the educators will know about it. So it's all very focused on this target group.

Besides of marketing the game, raising funds is also very different to Indie- or AAA-Games. We accessed some traditional game funding from Medienboard Berlin/Brandenburg, received support from the International Holocaust Remembrance Alliance and also from the BKM, the State Ministry for Culture and Media in Germany. In order to create an even richer experience, we are looking into additional funding possibilities, because part of our multi-platform project is an interactive museum experience that has gained a lot of international interest already.

Michael Geidel, Annekathrin Wetzel

Call for proposals

MiriquidiFilm is looking for game developers from different parts of the world who would like to collaborate to create and adapt versions of the game for their country. If you are interested, please come up and speak to us or email us.

Also, any feedback from you is welcome. Do these results from the Game Lab make sense to you? Where do you see things that we should consider for making the game more interesting for the use in other countries?

Please contact: **munich@miriquidifilm.de**

For updates and more information on the project visit **www.call-her-lotte.de** or the Facebook-page »Call her Lotte«.

making games

YOUR KEY TO THE

GERMAN GAMES INDUSTRY

MAKING GAMES MAGAZINE: 10,000 READERS
CENTRAL EUROPE'S MOST RELEVANT MAGAZINE FOR GAME DEVELOPERS

MAKINGGAMES.BIZ: 10,000 UNIQUE VISITORS
GERMANY'S BIGGEST WEBSITE ABOUT GAME DEVELOPMENT

FACEBOOK.COM/MAKINGGAMES: 10,500 FANS
EUROPE'S LARGEST GAME DEVELOPER COMMUNITY ON FACEBOOK

MAKING GAMES MAIL: EMAIL DATABASE
WITH MORE THAN 9,000 B2B-CONTACTS

MAKING GAMES PROFESSIONALS: LEAD DATABASE
WITH MORE THAN 700 FULLY QUALIFIED GAMES PROFESSIONALS

MAKING GAMES TALENTS: GERMANY'S MOST SUCCESSFUL
RECRUITING EVENT FOR THE GAMES INDUSTRY

KEY PLAYERS: THE WORLD'S BIGGEST GAMES INDUSTRY
COMPENDIUM WITH MORE THAN 80 COMPANY PORTRAITS

TALK TO US!

PROJECTS@MAKINGGAMES.DE

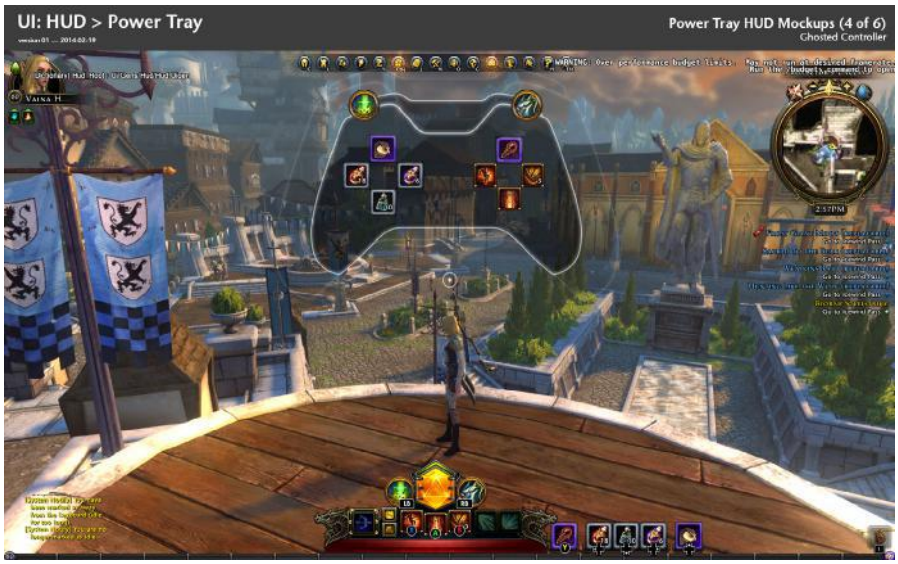
NEVERWINTER ADVENTURES
ONTO THE CONSOLE

Cryptic Studios is known for MMOs on PC and did well with that so far. Now the developer brings its MMO Neverwinter to Xbox One and quickly had to learn hard lessons on controls or even HUD design.

Andy Velasquez
is Director of International and Console Production at Cryptic Studios.

Andy Velasquez leads a team focused on expanding Cryptic titles to new regions and platforms. Andy joined Cryptic in 2008 where he initially joined as a lead for the »Star Trek Online« launch, and soon after served as lead producer for »Neverwinter« from the prototype phase to post launch and most recently to the console version.

Cryptic Studios has a long reputation of releasing successful MMOs for the PC and we were seeking out our next large challenge when we decided to bring our expertise in MMOs to the relatively uncharted console landscape. Fully featured MMOs on the consoles are very few and far between, so we saw a lot of exciting possibilities for our game as well as a ton of interesting problems that we would almost certainly need to overcome in order to be successful on a new platform. In this article I'll take some time to go over a few of the bigger issues we tackled with this project and provide some insights into our methodology.



To develop easy to use controls for Xbox One the whole team started iterating the controls early by using controllers when playing the PC version and taking notes which actions are necessary for the player and which ones are not needed all the time. By this workaround the team had a solid controller scheme ready even before a stable build of the console version was around.

Input!

The thing that immediately jumps to most people's minds when thinking of bringing a PC title, specifically an MMO, to the console is »how are you going to make that work with a controller?!?« This question became our primary focus for much of our early explorations of making the game work on the console. Our primary development pillar for our initial PC version of »Neverwinter« was an engaging Action Combat system, so it was only natural that the same system was our first focus of development for the console version as well. We needed it to feel great to raise your fighter's shield and block that Dragon's breath attack or to take your rangers bow back and then bury an arrow in an orc all with a controller in hand.

Back to Pen and Paper Roots

With our goal clearly in sight, our initial steps in development were fairly academic; since we had far fewer buttons and no longer had the finite control of the mouse available to us, we had to do some prioritization of actions that we expected our players to want to do. To begin we just spent time cruising through our internal documentation and playing the PC version of the product and keeping careful track of all of the things that we were doing while in game. More specifically we tried to keep careful tabs on more detailed metrics like how often were




With »Neverwinter«, PC developer Cryptic tries to bring an MMO to consoles but quickly had to realize that this meant a lot of changes had to be made to the game to fit the different play styles of this new audience.

we performing specific actions? How important was it to have access to that command instantly or was it an action that happened often but wasn't particularly time sensitive? What we ended up with was a massive list of inputs with categories like »need direct / immediate access« or »Not required to have immediate access«.

Another thing that we found particularly important to do was to have members of our dev team come up with their lists independently. By definition MMO's are massive in scope and much of their appeal is couched in the idea that there is a gameplay experience appropriate to the myriad tastes of the players who enjoy the game. For example I am a solo MMO player, I love progression, getting more levels, getting cooler looking gear, and exploring customization options. Contrast this with one of our producers on the console project who is a diehard PVP player. So by making sure we had a good spread of player types amongst our key team members we were able to aggregate a solid prioritized list of functionality that we needed to make work on the new input device.

In the early stages of development we continued with our academic explorations by taking our prioritized list and a picture of a console controller and just plugging in as many things as possible; again having key team members do this as individuals before bringing their ideas to the main group. This was also a time where »research« saw the most scheduled time as we spent hours and hours playing anything even remotely close to what we thought our gameplay experience might be. What we found was that there were two well defined camps that quickly identified themselves. These groups were separated mostly by whether or not they felt the game followed the paradigm most similar to a button heavy game that focused on the controller's face buttons as opposed to those who gravitated to a control scheme that relied more heavily on the triggers for basic combat actions. At the end of this section of development we ended up with a ton of paper iterations of proposed control schemes and it was time to make a few of them more real.

Making it Real

While our technical group was in the thick of simply getting the engine to run on the console, we knew we couldn't wait until that portion of the project was complete before we started getting hands on experience with these proposed control schemes so we immediately started exploring alternative solutions. We ended up just getting controllers that would plug into our PCs via a USB port, downloading a freeware button mapping application and started playing on our LIVE PC shard as best we could, again focusing on the combat and controller experience as much as possible. After individually iterating on our proposed button mapping for a bit we started passing the keybind files around, trying each other's out, iterating with the creator, and

Even little things had to be thought about: When taking a look at other action and RPG games for consoles, the team quickly learned that elements of the HUD were placed at the very edge of the screen or had reduced opacity when not in use in most other games.

stealing what others had come up with and incorporating the best parts into your own design. The pickup from our devs here was incredible as we had people streaming in, even from outside the main team, grabbing the keybind files and playing at their desk while they worked or taking the files home and iterating on the controls with their character from the LIVE shard. This was an incredibly important time for our development since we had found an absurdly light weight way to get a ton of iteration in as well as a relatively broad range of feedback, all before the game was even up and running on the actual consoles.

Cryptic Studios came up with three different designs for the HUD of »Neverwinter« on consoles which ranged from unaltered compared to PC (Option C, lowest image) to completely reworked to look like a typical console game and not a PC MMO (Option A, upper image).

What followed was a long road of iteration as we continued to get the game closer to an official release. Incrementally our engine got up and running on the console, the game became more and more performant, we entered an internal alpha, followed by a limited technical beta, all the while keeping our focus on making sure we delivered a great action combat experience on a controller by quickly iterating over and over through every stage of development. We were explicit from very early on that we had high expectations for combat on the controller specifically, this was reiterated in almost every meeting and playtest that we had. As a result we saw a good amount of excitement and engagement from our development team and in the end we have something that internally we are very happy with and judging by the feedback from our technical beta most players are satisfied with as well.

Lessons learned

The most important takeaway from this experience is to identify what is truly important for the success of your project and to start on it as early as possible, because you are going to need it. We went through countless iterations, starting from mouse and keyboard, to pen and paper, to an emulated version on a PC and finally to the console itself before we ended up with something we felt we were satisfied with; and we needed every bit of that time.

Think of the users!

From Cryptic's history releasing titles on the PC we know the importance of UI and UX design when it comes to having an experience that our players recognize as high quality so it was an obvious place for us to focus our attention when working on the port for »Neverwinter«. The bonus test that we had as a development studio is that though we have tons of experience doing UIs built for mouse and keyboard, we've not done much for a console; so considerations for very basic things like: couch and controller experience vs. sitting at a desk, screen size, expectations for the platform, etc. all needed to be made. Challenge accepted.

Defining our strategy

Way back when we looked at doing our initial UI/X design for »Neverwinter« on the PC we looked at the landscape of MMO's in general and noticed how UI design could be described as »windows EVERYWHERE!«, which was an aesthetic and gameplay experience that we were not interested in pursuing. As such our mentality was one of simplification. We endeavored to keep the main gameplay area free of clutter so players play their characters and not their HUD. Fortunately this was a mindset that we were able to continue pursuing on the consoles. So in the end our design directive was able to stay close to our original intent with the game's UI: we were going to match the traditional console aesthetic as much as possible without sacrificing the functionality that MMO players expect.

The HUD problem

When we talk about MMO players playing the UI instead of the game itself we are mostly referring to their interactions with the HUD, and the all too standard clutter that comes with that statement. Where this seems to have become common place, and even something that PC players are opting into with custom mods, the exact opposite seems to be true for console players and their gaming preferences. We spent a lot of time looking at console games on the market, particularly 3rd person RPG or action gameplay titles, and we consistently saw a minimalist approach to HUD design specifically. Often we saw elements shoved to the very edges of the screen and often many elements having significantly reduced opacity when not in direct use.

It was clear where we were trying to get to but we also knew that we wouldn't be able to get all the way to some of these other games we were looking at with the nature of our title. So our approach mirrored what we did with the controller and we started by just putting in a ton of time playing our game and evaluating our features into high level buckets like »need to see at all times« or »Can be tucked away in a deeper menu«. What we quickly found was that there was a surprisingly large number of features that were omnipresent on the PC version of the HUD that didn't actually seem to be necessary for us; perhaps it was just convention that made us feel like these were needed.

Tactically handling our list of »Must haves«

After our initial evaluation we had in our hands a giant list of features that we felt needed to be present, we had a strategic goal of simplicity but we needed a tactical plan for getting everything to work with those goals. Because we knew the UI was so important we decided to take the more time consuming, but ultimately more valuable, approach of doing three distinct designs for the HUD to really allow ourselves some exploration and iteration space.

Option A would come to represent the idea where we were ruthless in our deviation from the PC games, entire sections of the UI were cut without replacement, many other elements were buried behind menus and we explored a lot with opacity and completely fading elements out unless they were in use. Looking at some of these early mocks it was hardly recognizable as an MMO.

Option C represented a preservation of as much of the existing PC paradigms as possible, trying to make as smooth a transition as possible for players who already had experience playing PC MMOs and »Neverwinter« in particular. This manifested itself largely as shrinking of certain UI elements, shuffling things around, but not nearly as many large scale changes as Option A.

Of course then **Option B** ended up being our 50 / 50 split and we attempted to make a nice blend between the two very different poles we had designed previously.

Interestingly the final product of the HUD wasn't just Option B but rather elements from each. As we entered implementation and iteration time we would often say things like »I really like the power tray from Option C« or »Let's try swapping out the B version of the party window with the A version to see what it looks like with more screenspace«.

Lessons Learned

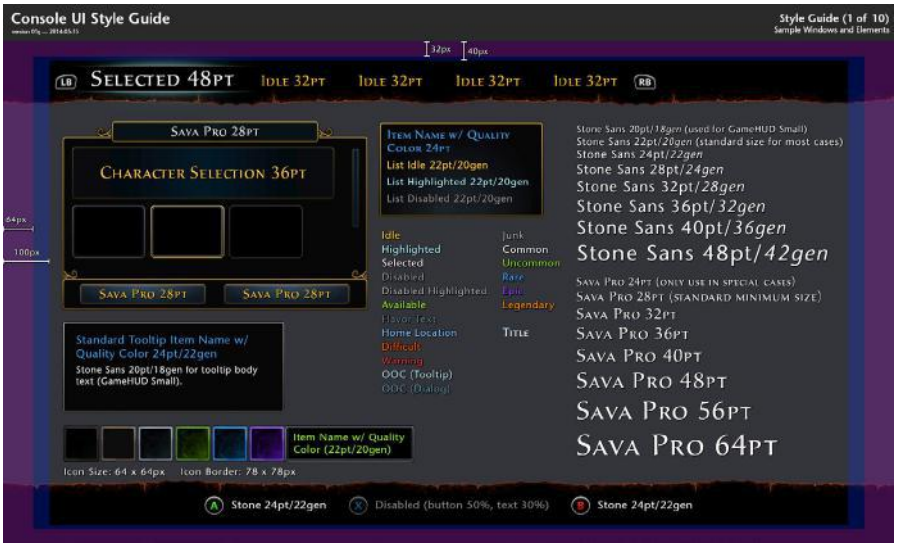
By allowing ourselves this freedom to explore some of the things we ended up implementing were honestly very surprising and yet everyone felt very confident that we were moving in the right direction since we had already explored other options. This was a process that we loved and ended up taking beyond just the HUD but applied to every UI element that we updated for the console experience. We were surprised again and again how some windows would lean heavily towards the initial A option where others ended up being basically Option C's implemented with no additional changes.

Wrap it up

There were many other challenges that we had to overcome with this project but these were a few of my favorites to tackle. Certainly our methodology here wasn't particularly groundbreaking or innovative; at the end of the day it can really just be broken down to prioritization and iteration. But what I do feel is important to note is that everyone knows that those are two important development processes but often we forget or don't prioritize for them as we deal with deadlines or other excuses that are all too common in development. These challenges are some of my favorites because we approached them in the »right« way and we came away very happy with the results. I hope that I will remember these examples for the next challenge.

Andy Velasquez

While controls on PC are quite easy using mouse and keyboard, browsing windows and menus on console might be more complex. Those examples show how Cryptic Studios reworked the controls for »Neverwinter« on consoles.



An in-depth style guide helped the team to define the general layout of all the console menus, showing the distance between the edge of the display and windows, etc. Some of those are even required by Microsoft to get a game approved.

ESSENTIAL RETENTION MECHANICS TO KEEP PLAYERS ENGAGED

Motivating and encouraging players to come back to a game over a longer period of time is difficult, especially in the exponentially growing free2play market. Mark Robinson explains the retention mechanics that help to keep players interested.

Mark Robinson is CEO of deltaDNA.

Mark Robinson has over 15 years' data mining experience industry, across utilities, retail, FMCG and finance sectors. A born statistical guru, Mark co-founded deltaDNA (formerly GamesAnalytics) in 2010 with games industry veteran Chris Wright. Mark has made it his personal mission to evangelize how analytics can change the games industry learning CRM techniques from other market sectors.

»Hearthstone: Heroes of Warcraft« is a great example of a well-executed onboarding process: it hosts a well-structured, engaging tutorial that unveils the game's complexity over time with explicit rewards and good use of signposting.

The rise of F2P gaming is undeniable. The F2P game market is now estimated to be valued in the billion of Dollars, and around 80 per cent of mobile app store revenue is generated by F2P in-app purchases. Importantly a large part of this growth has come from unlocking key demographics and regions which have always been overlooked by the traditional games industry. However, despite gaudy headline grabbing statistics it is still the case that the vast majority of F2P games fail. While a game like Candy Crush Saga might have over 100 million monthly users, the »typical« F2P game in the App store probably only has around ten thousand monthly users. This reality goes against the »founding principle« of F2P, that you can acquire many more players cheaply by giving your game away for free. The harsh reality of F2P games is that 5,000 new games hit the App store every month, and the revenue-based ranking system results in existing popular games getting the most exposure. Putting this together, it means that on release it is very hard to stick out from the crowd and attract new users organically.

For this reason, most newly minted games spend substantial amounts on advertising to acquire new users. In most cases this is costed-in per install, with a typical spend of 1-2 Dollars per newly installed player. A newly released game might source as much as half of its new players from paid acquisition. So it is clear the cost of releasing a game for »free« can be quite substantial.

The »free« aspect of F2P games also makes it very difficult to recoup these costs. In a typical F2P game only 1-2 per cent of players will ever spend, combining this with a typical spend of 5-10 Dollars means that the return from each installing player may only be around 0,10 Dollars. Comparing this to the acquisition costs, and considering that App stores pocket 30 per cent of all player spend, this means many games will actually lose significant amounts of money per installing player!

While there are various strategies for optimizing paid user acquisition, as well as new avenues for revenue generation (for example incentivized advertising, offer walls and cross-promotion etc.) ultimately for a game to return a profit it must be able to get players to spend in game. There are two reasons that games fail to monetize players. The first is that the in-game items are just not enticing enough, the »free« elements of the game are only enough to sustain gameplay. The second, and most common, reason is that games simply fail to retain enough users to the point where they might monetize. A typical F2P game would expect around 30 per cent of players to return to the game one day after install, and probably less than 15 per cent of players will still be active after a week. Much of this player churn

happens very early on; for many games, as much as 30 per cent of players will not make it past the first 60 seconds of gameplay!

Of course, retention isn't just important in F2P, it applies to the economies utilized by almost every game. Many AAA companies are now embracing the business model across all gaming platforms with the support of Microsoft, Sony and most recently Nintendo. With titles such as Daybreak Games epic shooter »Planetside 2« and popular MOBA »Smite« making their way onto current Gen consoles, F2P concepts are now reaching the heart of the games industry.

The end-of-level-celebrations of »Peggle« might be over the top, but they make winning feel like a momentous occasion.

Top 5 retention mechanics you need to keep your players

Any game that monetizes using in-app purchases relies heavily on player by retention, but so do those that employ advertising as well as games that utilize their own player base as an acquisition channel for newly published games. Today, the player has an unending choice of game experiences, mostly free and has several games actively competing for his or her attention every day. What will make your game stand-up to this competition and stand-out? Based on our analysis of hundreds of games on our analytics platform and our consultancy engagements, we have identified the five most common, and successful techniques, available to F2P game designers to boost retention.

1. Make your onboarding process fun, engaging and rewarding

The most important element to building a successful F2P game is providing not just a well-structured onboarding experience, but also one that is fun and engaging for the player.

Since the majority of players will not come back after their first session it is very important that you show-off your game's best features within the first few minutes of gameplay. Avoid an »on-rails« tutorial, keep text descriptions light (to avoid losing player attention as well as boredom), use clear signposting and above all, make sure the player has fun!

If your game has some cool content like a »Cloak of Awesomeness« vanity item or an exciting »Ultimate Flying Zombie Cat« bonus level, do you really want to wait until 90 per cent of players have abandoned your game before it comes into play?

On the flip side, if your game is complex, such as a MMORPG or CCG, a longer onboarding process may be necessary. Keeping your more casual and impatient players interested will be very difficult, but there are some methods to keep in mind to give your game the best chance of retaining more than just your core players past the tutorial. Methods such as advertising a tutorial completion reward or using a task system can be effective ways of introducing new game mechanics, while rewarding the player for learning.

Hearthstone: Heroes of Warcraft by Blizzard Entertainment is a great example of a well-executed onboarding process. Hearthstone is an addictive, highly accessible CCG, that takes a traditionally niche genre into the mainstream. Hosting a well-structured, engaging tutorial that unveils the game's complexity over time with explicit rewards and good use of signposting to ensure each step is crystal clear to the player. Bundled with quirky dialog between friend and foe during initial battles, it gives the game both charm and excitement to keep all players engaged.

So if you are you planning to release a niche title into the wild, take a step back and really think about your novice and non-target audiences. Will they understand your game and is the tutorial fun and engaging enough to prevent them from early abandonment?

A three-star tiered completion system like in »Candy Crush Saga« helps to increase the longevity of a game.

2. Increase the difficulty of your game gradually

The biggest cause of retention issues is due to game difficulty. It is extremely hard to get the difficulty curve right, especially for casual players. Developers are often the worst people to set the difficulty as they are very close to the game and have too much experience or inside knowledge on how the game works. As a general rule of thumb, a game that is too easy will cause less retention issues than a game that is too difficult. Balance is key!

The reason a smooth difficulty curve is so important is to give players a chance to make good progress and experience as much of the gameplay as possible before they begin to feel challenged. Players enjoy success through progression and taking this away too early will only result in poor early retention rates. Ideally, initial levels should never face the player with failure.

Although having some pinch points where the player should spend money is useful; these should be relatively deeper into the game and ideally with good signposting so players understand that they should spend money to aid progression. This is very much the carrot approach to monetization, which generally works much better than the stick.

The company

deltaDNA is a big data game analytics platform that gives game designers the capability to create, test and implement campaigns that engage with player segments line in the game. By analyzing and engaging live with many hundreds of games, we have identified patterns in game performance, player retention and player behavior which form the basis of this research. Find out more at **www.deltaDNA.com**

deltaDNA™ CHANGE THE GAME

That being said, one very important factor is to really get to know your players. More than half the players joining any game are novices, and so it's impossible to have a single balanced game. More often, we are seeing game developers taking a segmented approach to their players, delivering different experiences to novices than to experts for example. A one size fits all approach doesn't always work, if you provide a linear approach to your game-balancing, the experience desired by more proficient casual gamers could alienate your novice players by upping the difficulty too soon. Conversely, more advanced players could get bored because the game isn't challenging enough. Active segmentation and engagement allows you to analyze player behaviors and tailor the experience based on playing style to ensure everyone has a good experience.

Not only can it help monetization by presenting specific bundles or offers to defined players based on their style, it can be extended to look at those players who are struggling and losing momentum, determine what the problem is and offer focused support, to keep them engaged.

»Dragon City« adds a collection mechanic by splitting 369 characters into mini-collections and offering rewards for completing collections.

3. Rewarding

Players need to be rewarded for playing your game, they need to feel good. Rewarding is a way of helping players understand what they have to do.

Victories and rewarding should be made explicit. There is nothing wrong with being over the top as winning should feel like a momentous occasion, think »Peggle« end of level celebrations!

Making every success and reward a fun experience will boost morale and give the player a great sense of achievement. It is also important to detail any item, location unlocks or perks in a clear way to encourage use of such items or to direct gameplay. Even highlight upcoming rewards to let players know what's coming up will help to further encourage progression.

In regards to the quantity of virtual currency rewards, it really is a difficult balancing act. On one hand, you don't want to cause an imbalance to your game economy by rewarding too much. But on the other, you don't want to reward too little, resulting in a punishing experience which causes a retention issue. Finding that sweet spot is a tough decision for all game economists but getting the players into your game and having fun is vital to success for anyone. So if you are not sure, make your game more rewarding until you find your feet and balance the economy correctly, so that it fits how your players are spending and progressing in the game.

4. Host repeat play opportunities to increase the longevity of your game

Ideally a freetoplay game should be endless, it should avoid having a defined end, as that limits how far a player can go. This makes it hard for narrative games to become successful free to play games. The games that work the best are simple, arcade games with no defined end.

Being able to repeat parts of your game is a great way to ensure that your players come back, otherwise they'll stop as soon as they've completed certain areas. Give players a reason to replay missions or levels. Replaying could reward players with unlockables such as a new vanity item, access to hidden missions, or reward them for achieving a higher score.

Many games such as the much beloved »Candy Crush Saga« and stunt bike title »Trials Frontier« have adopted the three-star tiered completion system that offers a beginner, intermediate or expert completion level. Another approach is to achieve multiple tiers of completion by setting mutually incompatible objectives that require the player to re-play the same level several times in different ways. In addition to that, friend leaderboards are used to enhance the replay value that gives players the chance to better their friends' high scores and have the gratification of being better than someone they know in real life.

If your game features a wide selection of items or characters that the player can collect over time, it is good to build it into a rewarding collection mechanic with the ability to review progress. This can increase engagement and provide added goals with replay opportunities that appeal to completionists.

A good example of this is demonstrated in »Dragon City«. It presents their collection mechanic in the form of a »Dragon Book« which splits a long list of 369 characters up into a series of mini-collections with rewards for completing collections. With multiple collection rewards it makes the process achievable for both novice and expert players.

Although a larger development effort, multiple game modes, online, PvP and PvE lends itself

to repeat play. Allowing friends to battle it out or even work together cooperatively can help build social stickiness if strong social structures are built around it, with features such as Chat, Clan building and regular tournaments.

Any plans to release extra content in future updates should be made explicit as it will encourage long term retention and show that your game is still being supported by its developers.

5. Remind players to return to your game with good appointment settings

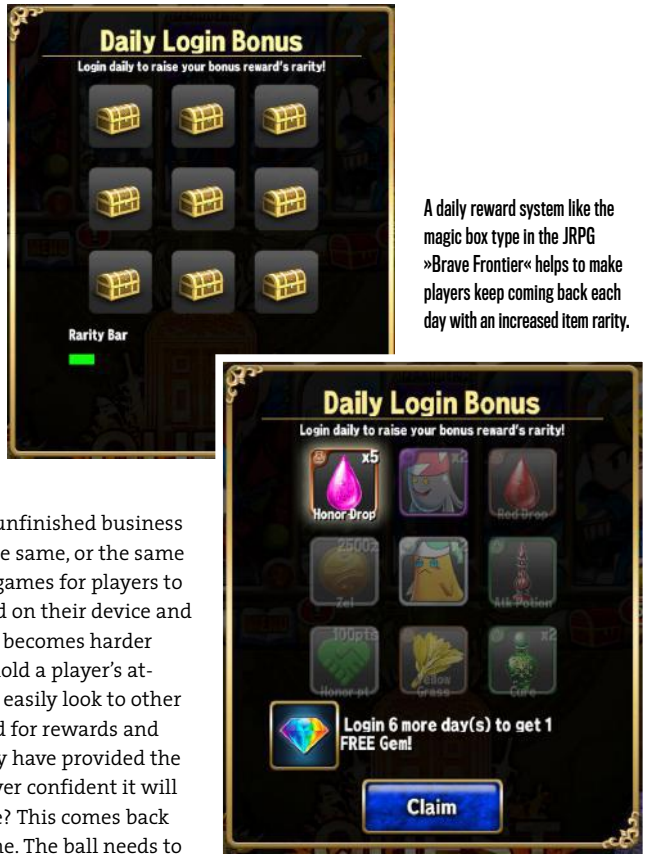
It is important to have some sort of incentive to pull players back to your game each day. With more and more premium games featuring monetization, even blockbuster premium titles such as »Destiny« (Daily Bounties), »Call of Duty: Advanced Warfare« (Daily Challenges) and »GTA 5 Online« (Daily Objectives) have some sort of mechanic to encourage players to jump on for a few hours each day. Much like F2P, the aim is to give players something to look forward to and something of value to reward their engagement.

The daily bonus is a well-understood and widely used mechanic in F2P, however, it is also beneficial to add extra options such as specific days or hours where rewards are doubled to give real value to returning players.

Providing and announcing rewards that incrementally increase in value for each consecutive day returned is a good way to reward your most loyal players and further encourage them to keep coming back to earn something special. As an example, mobile turn-based JRPG »Brave Frontier« uses a magic box type daily reward where players are presented with a selection of 9 treasure chests and prompted to choose one. They could win currency, new monsters or items. However, the rarity of the items in each box increases each subsequent day the player returns, encouraging them to come back for an even better reward.

Another strong daily reward structure can be found in »Hearthstone: Heroes of Warcraft«. Hearthstone opts for a Daily Quest appointment setting where players are provided one task per day, such as winning a match using a certain class, or to cause a defined amount of damage to an enemy hero. Quests give each day's play a structure and encourages players to try out all of the game's available classes with a virtual currency reward for completion. Quests can be stacked up to a maximum of three, however, the game allows players to swap out one quest per day if they find it too difficult or would like to try their luck at replacing it with something more challenging for a larger reward.

Ultimately, the objective is to get the user to remember to return to the game once they have left it. You should make daily-rewards explicit on day one, ensure they offer good value and integrate them into the theme of the game. Nobody likes a generic pop-up with a measly reward that grants just a few more seconds of gameplay, make it worth coming back for.



A daily reward system like the magic box type in the JRPG »Brave Frontier« helps to make players keep coming back each day with an increased item rarity.

Summary

Player retention is important for all games, not just F2P and should be about player enjoyment. It's often misconstrued that enjoyment is just about the satisfaction of completion, but it's more than that. The innate rewards derived from completion is only half of the process, there also has to be a feeling of unfinished business and a desire for more of the same, or the same but better! With so many games for players to choose from, both installed on their device and available from the store, it becomes harder and harder for a game to hold a player's attention. Players can all too easily look to other games to satisfy their need for rewards and challenges. Your game may have provided the satisfaction, but is the player confident it will provide the next challenge? This comes back to the structure of the game. The ball needs to start rolling with a first impression that meets or exceeds expectations, the onboarding process must be well balanced and engaging, the game must be rewarding and challenging and there must be reasons to keep returning.

This sounds simple, but unless you know your player types in detail and can engage with them in ways that motivate and encourage them, you are reliant on good fortune and guesswork for the success of your game. Big data game analytics platforms allow you to collect, process and act on player data, providing personalized game experiences to players, live as they play, so that you can adjust the game to suit the requirements of all your players, so they keep coming back.

Mark Robinson

»Hearthstone« has adapted the daily reward system into the Daily Quests, providing players with one task per day.

LORDS OF THE FALLEN
TACKLING A NEW GEN GAME WITH AN EMERGING STUDIO

About six month after the release of their well-received RPG Deck13 is looking back at how a small German studio managed to create a AAA game and talks about their lessons for the future.

Jan Klose
is Creative Director
at Deck13 Interactive.

Jan was one of the men in charge of the development of »Lords of the Fallen«. In 2001 he founded Deck13 Interactive together with Dr. Florian Stadlbauer and worked on about 20 games that were released since then.

Thorsten Lange
is Tech Lead
at Deck13 Interactive.

Thorsten directed the tech team at Deck13 and is responsible for the creation of the Fledge engine.

So, for some reason your small but growing game development studio feels the urge to move to major productions? Read on to find out how that worked for Deck13. After having released the Action RPG »Lords of the Fallen« worldwide on the current generation of consoles, we don't really feel that we are »there yet« in terms of being a worldwide established »AAA developer«. But we did take a major leap in terms of team professionalism, capabilities, and the reputation we gained. It took us three years and quite some blood, sweat and tears. But we feel that it was worth it. Well, in fact that probably depends on what you are after in life because there are easier things to achieve that might turn out to be more rewarding (see the end of this article). But in the end we all want to go big, don't we ... ?

Capturing the contract with a »small« studio

Deck13 started with a team of about 40 people. In terms of a AAA production, that's not huge but having 40 people on the payroll, we were already under a lot of pressure to pay the monthly bills on time. After a considerable down-sizing of the company in 2009 when we failed to secure a major deal after the release of our Action-RPG »Venetica«, we were slowly getting back to the size where we felt we had people working in every relevant position of the studio. It was around 2010 when we started talks about the next Action-RPG project. In fact, talks started at GDC in San Francisco when publisher CI Games was looking for an RPG to publish, and they were asking us if we »had a concept ready«. Hell, of course we had! Well, at least the outlines, but who would ever say »no« to such a question? So we agreed to keep talking and quickly updated and improved our RPG pitch. It was clear that for such a project we would need more people, more outsourcing support, and ... more experience.

Team transformation: Adding that »management« layer

It's not easy for a games studio to grow in size, at least it wasn't for Deck13. Firstly because good people are always hard to find, and if you are required to find a lot of good people quickly (because you finally have your contract and need to get started), it can be even harder. Also, employing more people meant that those who were already at Deck13 had to take on new roles that were not always those which people originally planned to work in. For many people this was a big part of what the change really meant: Letting go of things they loved doing. Lead programmers and lead artists had to become more involved into management, studio leads shifted towards PR representatives instead of actually working on their games, and 3D artists became outsourcing coordinators, giving feedback on other peoples' work.

In the beginning this caused a lot of problems, but later it turned out that it was the only way for us to create working pipelines and really good content in the timeframe that we had.

It also meant giving away responsibility to other people because there were so many tasks to keep track of that we weren't able to bundle them at just a few people. But people who were empowered in their jobs that way for the first time made mistakes. For us it was important to allow them to make these mistakes and not always step in to »correct« them because it seems hard to learn to be responsible and make the right decisions if always someone steps in when things go slightly wrong.

Also, we needed to strongly improve our communication channels towards the publisher. In the beginning, we underestimated how much communication and sharing of information was necessary to let the publisher keep up with what we were doing and let them understand why we were doing it. Later in the process we had team leads who actually weren't doing much else than talking to their publisher counterparts every day to inform them about the steps and find solutions to new surprising requests.

All in all, we ramped up to about 70 people, supported by more than 100 people at outsourcing companies at peak times. Still, when compared with the task at hand, it always felt like we were lacking quite a few people in each department.

Moving the studio's pipeline to »next gen«

Aiming for realistic graphics required investment of time and resources into a lot of different areas, apart (of course) from creating a fast and capable engine. The new consoles offered so much in terms of performance that the demand of delivering adequate complex content was really high, at least for a title like »Lords of the Fallen«. Therefore, a lot of work went into creating a »next gen look«, something that we hadn't done before, mainly because ... there were no »next gen« consoles around before.

Lighting was a huge area. There was a tremendous amount of light sources used by the artists to create a realistic looking light setup. Even though we had middleware support for lighting, the light setups created took a lot of time.

Pixel density was another issue. As we had a 3rd person gameplay, almost everything could end up in the camera close to the player, so we needed to create very detailed materials and textures so that the game did not look blurry. The same goes for characters. As the combat system is focused around fighting one or two enemies in close combat, it was important that we had a decent quality regarding their looks. That alone would have been enough work but we made the mistake to decide to show the characters close-up in cutscenes and conversations, too, so we needed even more details. Considering that the game was mainly about fighting and not about emotions in characters' faces, it just increased the workload and we should really have avoided that. At least we decided against adding facial motion capture, which was also in discussion with the publisher for quite some time.

Developing a AAA-game like »Lords of the Fallen« meant a lot of hard work for a small team like Deck13. In the end the team did manage to finish this huge project, gained a lot of experience but would not necessarily do it again.

Deck13 could not secure a publishing deal directly after the release of »Venetica« but started talks about »Lords of the Fallen« at GDC 2010.

Also, having to create such detailed objects made us switch from polygon based modeling to using sculpting which resulted in great visuals that would then be broken down to objects that were digestible by the rendering engine using different LOD meshes.

However, motion capturing in general was of course required, and we ended up with thousands of different animations for the hero and the enemies who could all fight using different weapon classes. Oh, and there were the boss fights. We created eleven boss fights that really pushed us to our limits.

Fledge, our own New Gen engine

People were asking us if we were serious about using our own engine for such an ambitious pro-

Further reading

Deck13 Interactive has published another Post Mortem on »Lords of the Fallen« that deals with the organizational issues between developers and publishers. If you are interested in this aspect of the project, go check it out here: **www.makinggames.biz/postmortem-lordsofthefallen**

To be able to compete with other big budget games, Deck13 created a proprietary engine that is able to fulfill next-gen expectations, e.g by using tremendous amounts of lights or very detailed materials.

The very first prototype level featured enemies looking like ancient Romans and was set in the wilderness. This turned out to be the wrong direction but the main gameplay concept was carried on to »Lords of the Fallen«.

ject. Why not use Unreal Engine or CryEngine?

Well, we wanted to be fully in control when it comes to technology and we felt that the only way to achieve that was to use our own. We started developing the current iteration of our tech back in 2009 and successfully shipped on both PS3 and Xbox 360, so it was not like we had to start from scratch. In addition, there are some 3rd party components which we used that would have been impossible to create on our own like NVIDIA's APEX for destructible objects and cloth simulation or Geomerics' GI solution Enlighten. While we did not know up front that we would have to switch platforms during production, in hindsight we think that it would have proven much more difficult to do that if we had chosen a 3rd party engine.

Whenever evaluating whether to use a 3rd party engine versus rolling your own, one major argument for an engine like Unreal is the number of shipped titles. But while the catalog of shipped titles using Unreal is certainly impressive for the last generation of consoles, the same obviously did not hold true for the new gen, since the platforms weren't even out yet.

But the most important argument for our own engine is that it's just in our culture – it's what the people in the tech team want to do. For the major part of the development cycle, the tech team consisted of only five to seven people taking care of everything related to programming. We think that this is a testament of what you can achieve if the people can work on (and focus on) things they love.

Having said that, it should be clear that »Lords of the Fallen« was not exactly a walk in the park for the tech team. We had to define what »Next Gen« would mean for us given the scope of the project and it was always a challenge to define and stay in budgets to make sure the content would work properly across all supported platforms. It was not only the »more of everything« approach that we took compared to last gen, we also developed techniques and features that just would have not been efficiently possible at all on older hardware (like our volumetric lighting solution). Keeping the tools up to date to support these new features was quite challenging as well.

FledgEd, our »What You See Is What You Play«-Editor

We needed an editor that could display the game in the same way that a player would experience it, as we couldn't allow for differences in the experience of, say, a level artist and the gamer, or a game designer and a gamer. Every in-between step would just have meant more work, and there was already more than enough work to do. So our editor was designed in a way that it was running the actual game. You could also say that the game was able to display the editor environment around it, from a user's point of view. Basically you can start the game in »game mode« and »editor mode«. In return, this means that a 3D artist actually plunges into the game when placing props, tweaking materials, placing lights, and then he can simply pick up the controller and walk around with the player character and check how this feels from the gamer's perspective. He can even start a fight and observe how it impacts the work he just did. Also, it meant that artists and game/level designers used the same view on all objects. If an area where someone placed a chest was too dark for the player to see it, the game designer would see that right away and was able to show it to the artist who would improve the lighting with a couple of mouse clicks.

Also, the platform-specific rendering code was created in a way that graphics looked as identical as possible on PC, Xbox and PlayStation so that it was basically not necessary for an artist to check if his new work would look right on other platforms, too. If it looked good on one, it looked the same on the others. If it didn't, it was mainly the programmers who cared for adjusting the rendering so that things came back together. This was possible for almost all graphical features but not for all: There were some differences where we actually needed unique graphical content, for example regard-

ing the particle effects because we were using GPU accelerated features of NVIDIA's APEX on PC which were not available on the consoles.

Feature driven game development

Back at the time when we were finishing developing »Venetica«, we didn't really have the technology in place to improve upon for future projects. While »Venetica« was indeed a multiplatform title successfully released on PS3 and Xbox 360, the console versions really were ports done by a contracting partner. So in 2009 (while »Venetica« was still in development) we began working on a new technical basis, which in the beginning was only a leisure project, aside or rather on top of the »real« work. There were no set goals or timetables; it was just about the fun of programming. At this point our programming team consisted of only a few people.

Besides getting the basics right and putting the multiplatform support front and center, we also wanted to improve the actual process of getting gameplay logic done. Hence, we needed to create and put the right tools into the hands of game (logic) designers to empower them to build the majority of the game logic without dedicated programming support.

We did not and do not have to this day a programmer who is solely dedicated to gameplay logic programming. Instead of hard coding gameplay, we prefer implementing small data driven building blocks, which can be combined to form the actual game logic. The game logic is therefore mainly represented by raw data, data that – conveniently – does not have to be ported to work on multiple platforms.

While we stand by this approach and would really like to take it even further in future projects, we do recognize some problems that emerged from the way we put it to use. For starters, we underestimated the programming efforts to get the boss fights right.

The major problem with boss fights was that almost every one required at least one unique feature which was not so easily generalized and implemented as one of the aforementioned building blocks – and even if it were, it would still have been used only once for a specific boss fight and in turn could not have contributed to the overall quality of the game on a wider level. A second example would be the general enemy AI, which would have benefitted from a dedicated AI programmer for sure to make the general enemy behavior more interesting and fun.

Still, this data driven programming and design approach was a key to keeping the team slim, keeping development time (and thus the budget) down and keeping bugs at a manageable rate. At peak times, we only had about half a dozen core programmers in the team, and they were at the same time also dealing with two next-gen platforms they never worked on before. In return, these few guys needed to be exceptionally good and work together very well.

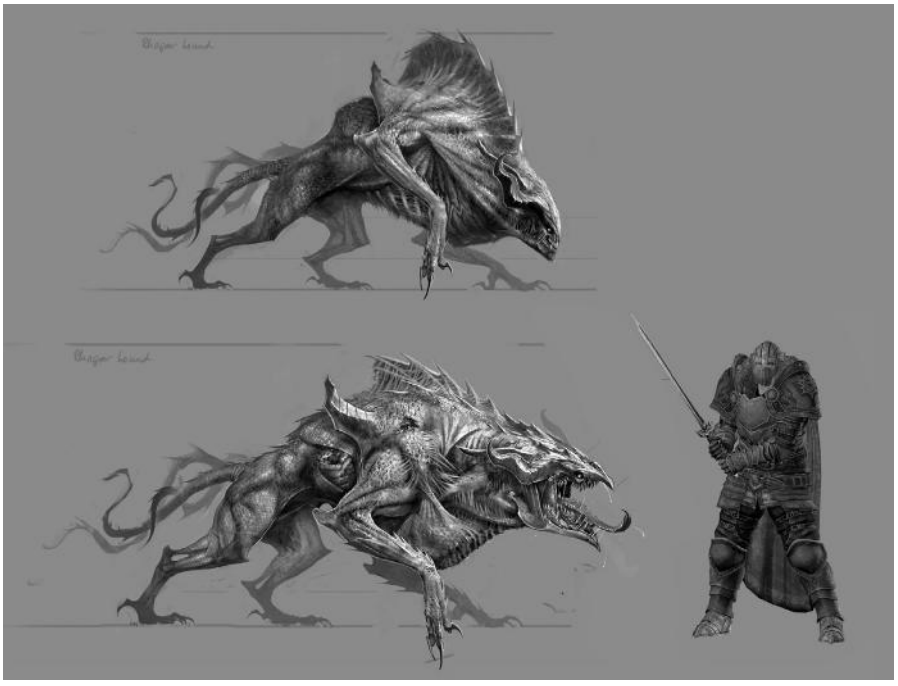
Proving it all: The Vertical Slice

It was all just theory before we saw our game in action. It was this one demo level that looks like the final game that we needed. But it didn't come easy.

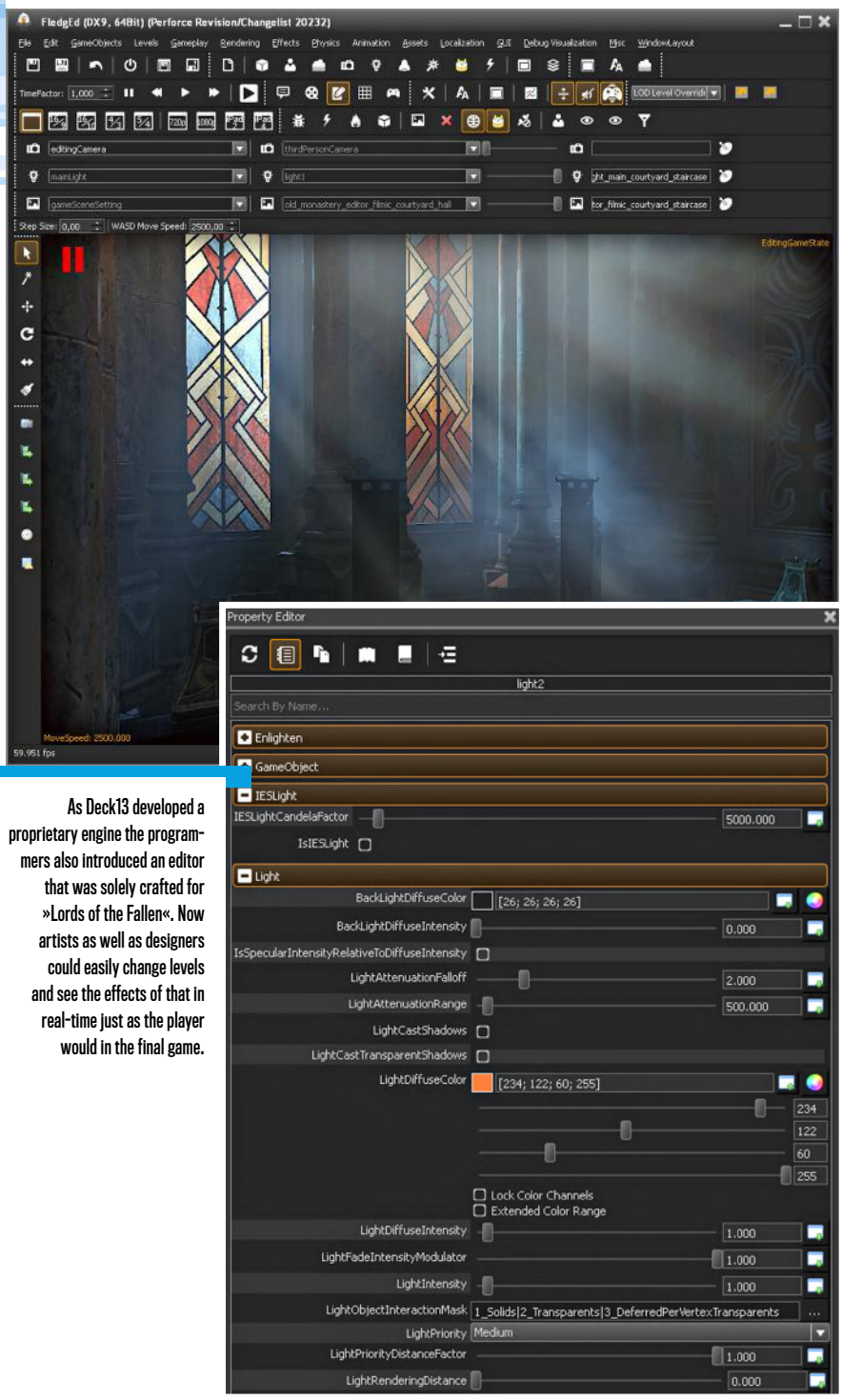
Actually, we did have a first »preproduction« build almost ready after about six months. It was set on an island, the enemies looked like ancient Romans, and the player was walking along spacious paths in the wilderness. Wait, you might say: »Lords of the Fallen« actually doesn't feature any of that. Right – almost.

After we reviewed this build together with the publisher, it became clear that the vision was not actually convincing enough to simply continue with it. It simply did not really feel »to the point« due to a variety of reasons, one surely being that there were quite some differences in the visions of some people on both the publisher's side and Deck13. Anyway, there was one thing that was working well and that was carried over to the next iteration: The tactical duel-based combat. The controls, pacing, and visual feedback were working really well. So this formed the basis for our second attempt, and this time we really needed to be absolutely on spot. There wouldn't be a third opportunity.

So we put all our efforts into the next version, luckily having the backing of the publisher for it. Everybody needed to prove that they were serious about this project and that they were ready to go all-in to make the leap. Endless night shifts, exhausted team members and lots of intense fights between developer and publisher dominated the next couple of months. But in the end we made it: We created a small piece of the game, a demo that you could show to everyone without having to explain that they should



Before having management structures installed a lot of decisions were made by top management (e.g. when art related), what gave the team a hard time. Later on certain members of the team were granted management responsibilities to smoothen the development.



As Deck13 developed a proprietary engine the programmers also introduced an editor that was solely crafted for »Lords of the Fallen«. Now artists as well as designers could easily change levels and see the effects of that in real-time just as the player would in the final game.

»please not look here« or that »the UI is of course still a dummy« or any of that. It actually was almost a finished part of the final game.

And there was another lesson that we had learned from our previous Action-RPG »Venetica«: If you have one part of your game ready after such a preproduction, and if it's really working well (which was the case for both of these games) then don't try to start from scratch afterwards, don't throw this part away. It's working, it's cool, you won't get any better than that unless you're freakishly lucky, that's what we have learned. So we built up on just that.

With »Venetica«, we hadn't done this. Instead, we had considered our preproduction level »one of the smaller ones of the final game but we will do much more, and greater stuff«; which we did but none of that we could deliver in perfect quality. This time we really tried to stick to what we had running in a good quality, and this decision turned out to be a very helpful one.

Balancing and release

There are surely games out there that are better balanced than »Lords of the Fallen« but still we were not doing too bad in that respect. We included a lot of press and player feedback and continued to listen and improve after the release. By the time the game hit the shelves, we weren't quite sure how it would be received and we were happy and grateful when the good reviews came in. Of course we had made a lot of mistakes on our way, but in the end the whole thing turned out to become something immensely positive for us.

Conclusion

We've learned an unimaginable amount of things on our way, and there are a lot of things we would not want to do again. At times, communication between developer and publisher was really critical, and at other times our internal structure was lacking management, communication, and simply faith. But yes, it was worth it. A flagship project is an awesome thing. It opens doors, and it gives confidence. Would we do it again? Yes, but not in the same way. We'll try to do it in a more »grown up« way next time, because maybe (only maybe) we grew up a little bit in the process. Or maybe we just got older!

We don't think that a studio is actually required to try and take on a big project like this one; there aren't many occasions to secure a fitting deal that a studio can sign with dignity. So why not try to do something smaller? Comparing it with the film industry, why aim for a Hollywood movie, compete with a handful of companies that are funded way better than yours, when at the same time you could produce the greatest short film or even music video of all time?

With new distribution platforms and financing models, size doesn't matter anymore and it really comes back to innovation and quality. When offered the perfect deal to do another »Hollywood movie«, we would never say no. But at the same time we're opening up for partnerships with other, smaller Indie studios that can profit from our current standing. Having a big release in your back does open up a couple of doors. If you're one of those teams, if you have a project that's promising and far advanced, think twice of signing it away and talk to a developer who might, in the end, be the better partner. Drop us a line!

Jan Klose, Thorsten Lange

TOOLS

You need great tools in order to build great games. Experts from the industry share their view on recent updates as well as new software solutions and reveal their favorite little helpers.

ARTICY:DRAFT WWW.NEVIGO.COM/EN/ARTICYDRAFT/OVERVIEW/

Daniel Sawitzki
Senior Programmer
at Blue Byte

articy:draft is a suite for game design. Developer Nevigo breaks it down into the topics story, entities and locations. Every object type can be equipped with a set consisting of attributes and be adjusted to one's own project. – NPCs can act as VO in dialogues, refer to their armor and be placed directly in locations. Through the implementation of Perforce and SVN, teams work simultaneously and version-controlled. In our project, we use articy as a hub. While, for example, we create NPCs directly in articy, we import other things – e.g. from Excel or the Loc database. Different exporters then lead to our build pipeline from this central hub.

The game design suite articy:draft differentiates between the topics story, entities and locations which can all be linked to each other.

PIXLR EDITOR WWW.PIXLR.COM

Pixlr Editor is a comprehensive app which makes the most important functions of major image editing programs available on the go.

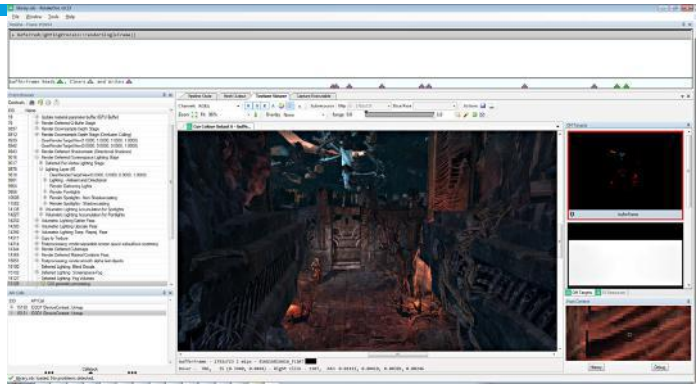
One of the most important tools for me is the Pixlr Editor which isn't a software that you need to install, but is available as both browser and app versions. For me, this makes Pixlr Editor even more useful since I constantly switch between Windows and OS X, depending on the work environment. Pixlr Editor is a comprehensive image-editing tool – changing the image size, inserting watermarks, cropping images and inserting text and effects are just a few of the features that the web app has to offer. The user can therefore comfortably make small adjustments whenever the art/design team isn't available.

Milan Đukić
PR Manager
at Nordeus

RENDERDOC V0.23 WWW.RENDERDOC.ORG/BUILDS

Philip Hammer
Senior Graphics Programmer
at Deck13

RenderDoc is a tool to check a (render) frame, showing all commands and data sent to the GPU by the application. It allows detailed analysis of all drawcalls, buffers, textures, renderstates etc. which are part of the final image. Debug overlays, too, such as depth test, clip ranges or quad overdraws are important features of this tool. RenderDoc is not only a useful tool for rendering programmers; (technical) artists, too, can use this tool for early detection of problems in 3D objects and/or entire levels.



RenderDoc helps artists with early detection of problems in 3D objects or entire levels.

STARPOINT GEMINI 2
LIVING THE DREAM ...

Three brothers started to work on a big scale space game six years ago. After finishing this first project and releasing an even more popular sequel the team knows how hard game development can be.

Mario Mihokovic
is CEO and co-owner
of Little Green Men Games.

Mario Mihokovic is a long time gamer and space enthusiast. His mission is to be part of virtual space until the real one is within reach. Mario was born on May 24th 1980 in Zagreb, Croatia. Only one year after finishing his studies on history and social studies in 2005 Mario started to work professionally on games. In 2007 Little Green Men Games was founded and Mario is part of the team ever since. Mario is married but with no kids yet, and therefore hopes to use that »peaceful time« to make more games.

This is the story about three brothers who thought that dreams can come true. This is the story about »Starpoint Gemini« and its successor. A story not yet told, at least not from my point of view. We loved »Freelancer«, »X-Wing vs. Tie Fighter«, »Star Trek: Starfleet Academy« and many other space simulation games. At the start of our work, in the corner of small apartment's kitchen, the topic of many discussions was »Are we insane?« But then again, we were young, eager and so determined to give it a go.

With almost no budget and limited manpower, these three created a small miracle. The first game had its issues, it was not perfect, but it was something special. »Starpoint Gemini« was a game you will either love or hate. Many loved it, simply because of its imperfections (e.g. self-made voice-overs) and mostly because they knew how few people and how much sacrifice stood behind it.

And then, it was time for release. Tired but relieved, we watched as the game went live on Gamersgate. But hey, why all the negative reviews?! Doesn't work? Can't run the game? A disaster was unfolding in front of our eyes. The game was drowned in bad reviews and complaints, not because of its imperfections but because of procedural mistakes as the applied DRM corrupted the ».exe«-file. There was nothing that could bring it back to where it should have been from the start. Sales slowly grew again later on and some money arrived, but it was far below expectations. Our emotions were mixed at that point – on the one hand we actually made a game that reached a worldwide audience, on the other hand initial reviews weren't as good as we had expected.

What Went Right?

»Deal with it and move on« would be the first thought that comes to your mind. And we did.

In a galaxy far, far away ...

We decided to make a sequel and invested everything that we had earned from the original game in »Starpoint Gemini 2«. Within a year, the project was recognized by the Ministry of Entrepreneurship in Croatia and we received a financial boost for the game. We hired more people, one of them today known as »Oliver the Sleepless«, our CTO and lead programmer who never sleeps, and we continued to work hard on a new game. We introduced a new self-made engine called »Whale 2«, and decided that if we got lucky, we would push the game to Early Access on Steam. With the help of Iceberg Interactive, who published the game, Steam accepted us and an incredible journey began. A journey that is still underway and we hope it will last for a long, long time.

Our community

The process of creating a second game was even more complicated than the first attempt. We felt a lot of game elements needed to be added, and we knew that we could do a much better job than before. We invested everything we had, and made huge ground breaking changes in engine, visuals and gameplay ... and we had absolutely no idea if we were on the right track at all. Later, the launch on Early Access gave us some financial relief, and more important massive amounts of feedback that provided good orientation where we were heading with the project. From there it was just important to interact with the growing community and steer

development accordingly. Since our team is small, we didn't have enough money for big marketing campaigns and the game's success depended mostly on the community. We were amazed when we saw the first reactions: so many people enjoyed the game, talked about it, players were active in forums, and positive reviews started to come in. That of course pushed the team to work harder and faster and our community also became something much more than a simple bunch of players. The time spent with the community was a relaxing, enthusiastic experience that spawned dozens of spectacular ideas.

Modding

»Starpoint Gemini 2« is moddable and that turned out to be fantastic. We already have so many mods on Steam Workshop that I cannot even count them all. Many popular ships from famous Sci-Fi-movies have been created, and we're looking forward to see what the community will come up with in the future. Modding is a big part of this game and we hope it will grow even more.

Release day

Just like the twin suns Castor and Pollux in the game, Early Access and the official release are two different things. And there lies a danger. Selling the game to people in Early Access means selling the game to people who voted for you, read about you, know what the game is all about and are usually extremely patient. Selling the game to a broader audience on release day means there will be people who will hate the game, who will love it, who will buy it without even knowing what they are buying just because it's marked with a 20 or 50 percent discount. But most of all, people will judge the game more harshly and that is understandable and expected.

Luckily, the game was a success, and most people loved it. There were tons of question and we stayed up late every day, helping to cover the forums properly, because our players deserved it. That was no small task, because Steam is a global market and people all over the world are playing the game. When we go to bed, America wakes up and you cannot leave those people without an answer for six or eight hours. We tried to cover as much hours per day as we could and it paid off. New positive reviews began to fly in, some big magazines gave us high scores and for the first time, we really started to believe that our story as a development team is only just beginning. Not to mention that some people called our game a »spiritual successor to Freelancer«.

What went wrong?

Years ago, as we first came up with the idea of an unusual part of space – the Gemini system with an incredible number of habitable worlds and its strange relations with a far away

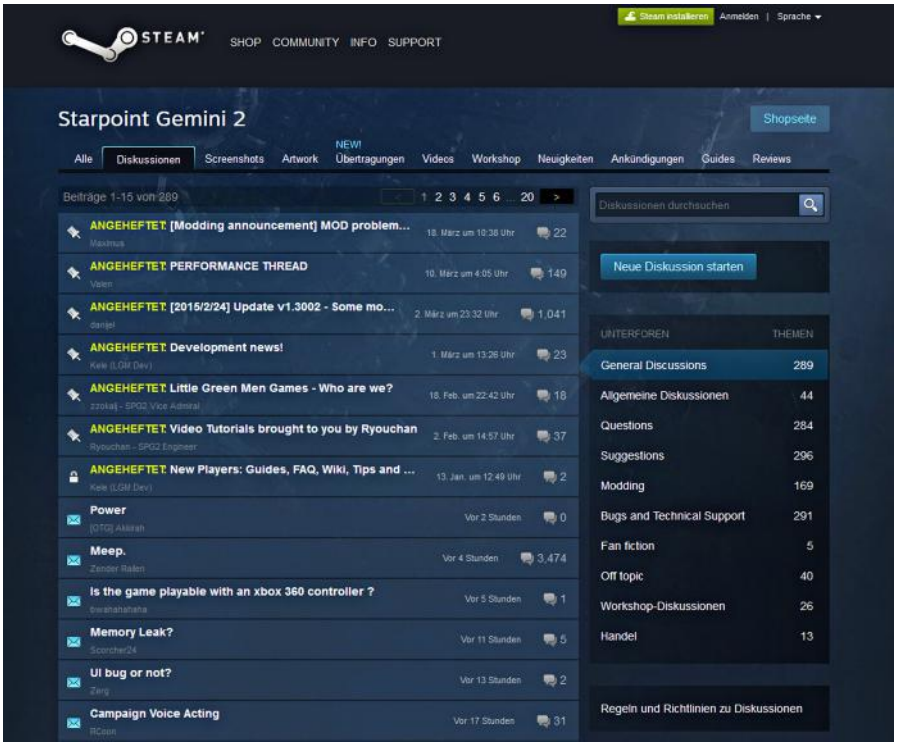
The development of »Starpoint Gemini« started with only three brothers working on the game. Even if they knew that a big scale space game was a more than just ambitious idea, the game was successful enough to start working on a sequel (see image).

Empire – we already knew that this story could not be told in a single game.

Working on a sequel

So, sequels were mandatory in our opinion. And we already made mistakes at that step – we treated each game separately, and in case of a single player game that is not the best idea. You need to think ahead and include the fact that the game will have sequels in any of your plans. In addition, if the franchise will continue, be careful and think about where the storyline and technical demands could take you. No matter what you do, always be aware that your plan will inevitably change over time.

One important lesson here is: absolutely never be stubborn. When you reach a stage where something just won't go according to



After »Starpoint Gemini 2« had entered Early Access on Steam the team quickly learned that it is very important to monitor the forums - be it to solve problems or to gather feedback. The team even tried to achieve some kind of 24/7-support, so that different time zones were covered, too.

your beautiful long term plan, don't fall in the trap of forcing the issue according to your »flawless« idea. This usually has two possible outcomes: either you will end up making a feature that will just function bad, or you will get stuck and solve the issue after such a long time that you will never be able to catch up again. By this another feature might suffer instead. Just accept it and find a workaround.

I can honestly say that our first game was about 50 to 60 per cent of what we intended to make, and even this cut down version of our game was done a lot different from what we first planed. But it was also better in some aspects than projected.

Second important lesson: if you're going to make sequels, make sure the quality of all the work done is satisfying enough. The first game in a franchise will always remain the corner stone of any future sequel. If you get that wrong, the starting position for your second game will be greatly diminished. We did exactly that, and though we succeeded to correct it mostly, the amount of effort and time that was needed to remedy the flaws of the original game were just immense. If possible, try to avoid that.

Technical Abyss

Anyone developing a game on their own engine knows all the curses of this path. PC isn't a console, and hardware parts come in so many variations that you basically optimize your game for countless machines. Couple that with a countless number of available drivers, and you'll see what you might be dealing with.

When you test the hardware demands of your engine and game, be very careful what you will list as minimum configuration. We stated a minimum configuration that was really just that – minimum. We played the game with these hardware specs and it worked. Not best and not with many details, but it worked. However, someone will always have a tiny program running in the background or an outdated driver installed so that a crucial amount of memory will be in use, just enough for the game to stop running.

Through Steam you can get excellent and useful information on what configurations and operating systems players use. Take that into consideration, but be careful how you draw conclusions. We got the data that only 4.5 per cent of our users had still Windows XP running on their machines. Therefore we weren't too concerned that the game will not support XP. Wrong! First day after release, many players with that platform came to forums angry that the game didn't work on Windows XP. At that point we really wanted to help them somehow, but it was way too late. It is better to take into account as many potential issues like that as possible.

The shame of bugs

And then we come to the issue of pure technical bugs ... As the development came to final stages, the list of interconnected variables within the game was staggering and even the smallest changes caused a cascade reaction that needed days or weeks to bring back to balance. Plus, against better judgement, we of course continued to add new content until the very end.

In any case, we had bugs at release. If you don't have the budget to cover detailed professional bug testing sessions, you have to deal with it yourself and that is not an easy task. We had a lot of feedback from a public beta during Early Access, but even that was not enough to cover everything. What is important: be aware of that and on release day be on standby to monitor for issues and fix them »on the fly«. Players will have more patience if you show them from day one on that you will not abandon the project and that you will take care of problems fast and effectively.

Another thing worth mentioning: have a strategy to handle an emerging crisis. No matter if you are an indie team, if you have stretched your budget to the limit or if you haven't slept for weeks before release, you are the developer and you are expected to offer a complete gaming experience to your customers. Every bug, technical problem or missing feature will therefore be seen as the developer's fault. What we did, was trying to keep an honest line of communication. Never try to lie to players. They aren't dumb and most of them played a lot of games and are familiar with common »release day« problems. In fact, we realized that our players have pretty amazing in-depth knowledge of the development process. Moreover, we even used this knowledge and enthusiasm of individuals to address problems faster.

In the end, if you engage your audience and work with them closely, the final result will be a better game, plus you might learn a thing or two. In return, reward your players with features of their choice or free content. Mutual respect can really spawn some incredible results.

This picture shows the team behind the »Starpoint Gemini«-games while they are celebrating the release of the second installment.

Due to hard work and to the loads of feedback that the team could gather during Early Access the game was quickly popular on Steam which lead to good feedback and improving sales figures.

Desirable Bugs?

In our case, we had numerous demands from players to re-introduce specific types of bugs. One example for that is the existence of so called »ghost planets«. They were just the manifestation of a memory leak in early Alpha versions, but a lot of players saw them as a nice unexpected addition to the normal universe of the game. Once they were eliminated, we started to get requests to put them back in and even add some meaning and story for them. The question was how to return something that was never meant to be there, but the game »created it by itself«. Issues like that are definitely something you do not expect during development. Re-engineering fixed bugs? Preposterous but intriguing at the same time.

Then there were heroes – special persistent NPCs. Many of them are modelled after real player's characters and players enjoyed hunting themselves in the game. Some of them even went a step further and asked us to create a bounty hero based on their wives or husbands. In a way our development was a catalyst to vent some marriage issues ...

The Maze called Early Access

If you would have asked us a year ago whether we should participate in Early Access or not, there is no way we would have been able to give an unanimous answer. It was a strange approach to development, new and unexplored, and we were honestly scared of it. Now, after the entire experience lies behind of us, we can say it was a very positive one. But bare in mind that it will have dramatic impact on your development process. Why? Because Early Access has several specific properties:

1. Upon entering Early Access, you will be offering a »living« product. Kickstarter campaigns offer a complete plan and a promise to create something based on that plan. Early Access allows players to actually enter the game. Every step a developer takes is closely followed and there is little room for mistake. In addition, players will expect regular updates and that will disrupt development plans constantly.
2. Update contents will also become a very blurry concept. No matter what you actually plan to add in a certain update, be prepared to change it numerous times. Sometimes you'll need to include urgent fixes, sometimes it will be a critical feature modification. In any case, first priority will be to have a stable, playable public version during the entire Early Access process. In most cases this will mean that your team will have to have different internal versions on which the main bulk of development is being worked on. This may sound easy, but in fact poses a number of challenges with syncing, updating and closing public versions. During one year of Early Access we even had a few situations where we prepared and launched a wrong version on Steam.
3. Balancing hell: Oh yes, Early Access had a significant impact on the overall balancing process. This happened because thousands of players spent dozens and dozens of hours in game during beta phase. At release, they were very skilled with all game elements

One important feature to keep a game alive is involving the community into development or to allow modding. For»Starpoint Gemini 2« both ways helped to boost sales every once in a while.

and found the game to be too easy. New players on the contrary found it to be too hard. We introduced several difficulty levels but it only helped contain the problem.

4. Communication with the community: This will take a lot of time and effort, so be careful to include this to your timeline. Again, once you see how well people react to open discussion, and how awesome ideas pour in, you'll get hooked and will monitor forums more than anything else. Then there is the matter of bad feedback. Of course not everyone will like the game or everything you do. And we sometimes had a very hard time accepting this part of work. You spend so much time and effort to create something, and then it gets trashed ... The only way to handle this is to keep in mind that most negative reviews aren't evil, but simply a way the community shows you if they like the direction you're going or not. In the end, you can learn from it and make a better game. The most crucial thing is to remember that you are making the game for them, not yourself. The developer's obligation is to explain the vision behind the game through honest discussions. If you succeed in this, the community will be able to better understand what your goal is, and what the best way for them is to participate in the entire project.

Questions that you always start and end with

Ok, so now we have some experience in developing games. You'd say it gets easier with every finished game, and development becomes a well-oiled machine. Well, in a way yes, but at the same time no.

Main thing is this: Once you finally wrap up the game and push it out there, few things can happen. You can make stellar success, in which case you don't need to read this part anymore because your cold Marguerita is just arriving at your Caribbean apartment. Second option is that your game fails for whatever reason and that means game over. Third option is that you are successful enough to keep your team together, but you will continue to struggle with budget and continue to advance in small steps. First two options are relatively rare, while the third one is certainly the most common one.

And that isn't necessarily bad – you get the chance to continue doing what you love, improve yourself and prove that you can do better. Plus, it only postpones your chance for that Marguerita. Not bad at all, right?

What you then need is to stick to a proven development plan and just continue pressing forward. In our experience there are several details that are crucial to the entire roadmap:

1. Create a good, detailed game design document (GDD). Spend as much time as needed on this step. If you make a very detailed list of all the things that team members have to create during the entire development process, it will dramatically cut down the time needed for planning and execution.
2. Organize your team. It makes a big difference whether a game is created by a team of two, five or fifty people. Still, you should plan ahead early if you think to add team members to your project during development.
3. Make the most important technical decision, whether to use an existing game engine or create your own one. This comes down to the clash between speed over control. Existing engines on the one hand speed up the development, but you will have to work within the limitations of that engine. Your own engine will, on the other hand, dramatically slow down your development, but will enable your designers to create whatever they imagine.
4. Our game aimed at a very narrow genre niche. In general that can be a good and bad thing at the same time. Good is that you will compete with fewer titles upon release, and bad is that you will have bigger troubles reaching your audience and explaining customers what exactly you strive to achieve. Both ways can work well, but this is something you need to be aware of in advance.
5. When it comes to optimization, be very careful. We made another mistake here. Do not plan optimization to be the last step in technical development. After the game is already packed with an insane amount of content, it will be very hard to impose some dramatic optimization. And if your frame rate is too low, final stage of development is a very bad moment to fix that. It isn't impossible, but will be much harder than if you keep an eye on the performance for the entire process.

»Starpoint Gemini 2« was developed to be moddable from the very beginning. Thanks to that the community has already created a lot of spaceships from popular Sci-Fi-movies as an editor is implemented in the game, which keeps the game alive over a long period of time.

Game Of Numbers

After you've more or less successfully survived all the steps on a long path of developing a game, we are ready for showdown...

The enormous amount of work during the past years, countless nights spent in the office,

all the people surrounding you ... everything comes down to a single moment: release day. The result of that day will determine the destiny of your team. If things roll in the right direction, everything you've worked for gets a meaning and you can start materializing new projects. If things roll badly, all effort was futile, and there won't be a future to talk about.

The direction where things will go from release on is usually determined within first 24 hours. Worst thing of all is that once you hit the release button, there is no turning back anymore. A chain of events is started and you can only sit back and bite off your nails. From the first second on, with the very first comment, a burden of questions will start spamming your brain: »Were we ready? Is there something more that we could have done? What if ...«

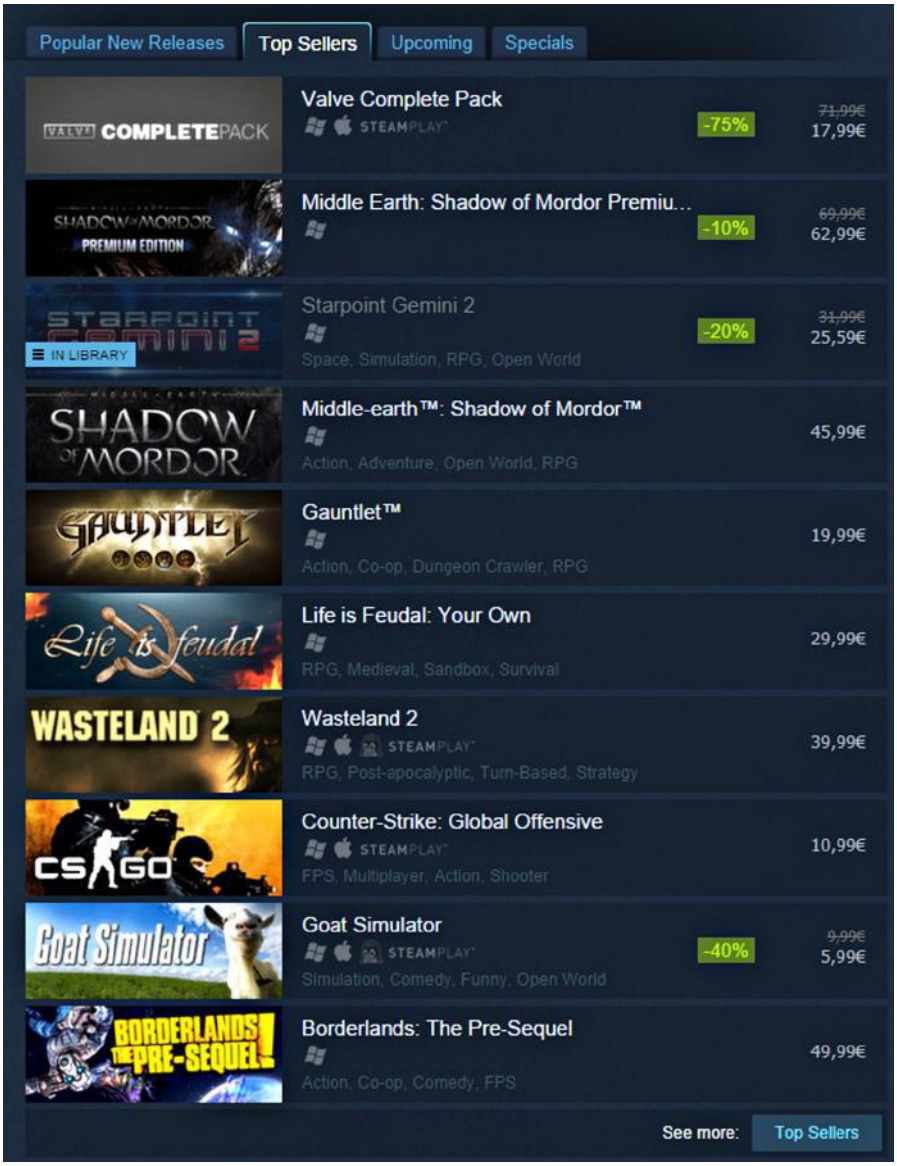
Anyways, let's go forward. Days go by and you realize ... numbers are good! Most players like the game, reactions are positive and sales are stable. You look at the faces of your teammates and see it clearly now: You will continue this amazing journey. You will be making another game, and you will get another chance to grasp the vision of a great gaming experience once more!

So, how do sales figures develop? I can speak only from our experience as I have no idea if this is a general rule or exception. First day is absolutely the best. Your game is new, you get the greatest exposure and visibility and the largest number of new players. Your email inbox will be full of requests for steam keys, both from journalists and fakers. In our case our publisher Iceberg Interactive took a lot of this burden from us there and that helped a lot as we had more time to spare for bug fixing and polishing features.

Following this initial sales boost, it is important to keep the momentum. Talk to your players, explain to them whatever they need to know. Many new players will give the game just one chance. If they get stuck, they have dozens of other games in their libraries to transfer their attention to, and you need to keep them in game long enough for them to see it in all its glory.

From here, things will evolve pretty much on their own. If the game is well accepted, players will spread the word themselves. And if you've been through Early Access, you already know the drill: communicate, assist, and develop further. The only difference is that all this will be scaled up dramatically as lots of new players pour in. However, there is a positive point as well: if you've gathered enough players during Early Access, they will usually be happy to assist newcomers themselves. When your community surpasses 100,000 players it will in many respects become a self-sustainable living entity. From there just listen to what they say, incorporate it in any new plan, and keep your community informed of your future plans.

Regarding sales behavior, for us it worked mostly in waves: You get a large peak at release, and then peaks follow the pattern of released patches, updates, DLCs, steam discounts etc ...



Discounts on Steam can help to boost sales of course. But at the same time it can mean a lot of work as a lot of new players pour into the game, who might have technical issues or who might complain about the game if it is too niche.

We still have a dream ...

To be completely honest, the past six years of making games were the best years of my life. I have never done something so creative and something so liberating. Along the way we came in contact with such awesome people who share the same passion.

In regards to the business side of the job, it is far harder than we had expected. Competition in this industry is merciless, and required funds are something we are still miles away. But we are still learning, and if »Starpoint Gemini« was 40 per cent of our original plan, then »Starpoint Gemini 2« took it to about 70 per cent and we're still improving it daily.

My dream, our dream is to fight for a chance to make one title in such a way that no compromise is necessary, just one chance to reach 100 per cent of the vision that we have in our heads. Our next task is to improve the current game, and make it so good that it becomes our ticket to embark on that next mission. **Mario Mihokovic**

After the second space game has been released the team behind »Starpoint Gemini 2« still dreams of a perfect space opera that they could not yet develop as pictured in their minds.

CREATING A PROPRIETARY ENGINE AS AN INDIE

Zeppelin Studio was founded by some students who wanted to finish the game that they started to work on for university. The plan that sounds already like a tough ride got even more curious as the small Indie team decided to decline well known game engines and work on one of their own.

Michael Benda
is CEO / Co-founder and Project Manager at Zeppelin Studio.

Michael began working on »Schein« in the course of his Master in Game Engineering & Simulation. He is project manager, developer, level designer, animator and self-designated maker of coffee.

Philipp Schaefer
is CTO / Co-founder and Lead Programmer at Zeppelin Studio.

Programmer since childhood, Philipp was the engine lead right from the start of the project. He co-founded Zeppelin Studio and is now responsible for all technical decisions and designs.

The previous part of this article covered the steps that the team at Zeppelin Studio took to grow a university project into a full indie game title: from the design process, to the founding of the company and straight to the release of their pilot project »Schein« on Steam. Our article in the last issue briefly mentioned the team's decision to create their game in an own engine, without going into detail however. Since this was a major aspect of the whole project, the following text focuses on the background and the reasons of building an in-house engine and the most important pros and cons – in short: Why on earth would an indie prefer C++ over Unity?

A forest of engines

The short answer to this question is that there are lots of things you learn and realize while doing so – although most of you will figure out that using a self-written engine as a small start-up team is not the best idea.

But maybe we should start off with a small introduction on how we got to build our own engine in the first place. Our project started at university. »Schein« is a puzzle platformer with a classic orthographic camera. The core mechanic is provided by a portable light that reveals new dimensions – or to keep it in a developer's view – a dynamic light that changes rendering and collision. This is a game mechanic that requires some deeper access to the engine code or would otherwise result in dirty workarounds.

In university we were taught to work with the Gamebryo engine in the first semesters. It has been used in many cross-platform titles including several titles of »The Elder Scrolls« or the »Civilization«-series, but still we found that Gamebryo was a complicated monster that costs you lots of hours before it could be used properly. Or maybe we were all just still to green. Since the Gamebryo engine got officially cancelled in 2012, we were obviously not alone with this problem. The CryEngine was then selected as the new engine for the university lectures. While the CryEngine is good for many things, including cutting edge 3D shooters, it isn't very convenient for prototyping a 2D puzzle platformer as a student. The reason for this is simply that it was developed for AAA 3D games. To be honest we were really surprised when we saw that an indie team was actually building a puzzle platformer with it (»Rolling Sun« by Merhunes).

Finally in early 2011, after having finished the conceptual work, we ended up building two prototypes in two different engines: one in CryEngine and one in Unity3D. This was when we realized that Unity really surpasses many other engines when it comes to fast and intuitive prototyping. While we still heavily struggled to force the CryEngine into the orthographic side-view, the Unity coding was done with ease. It wasn't a beauty of shader coding or graphical design, but it did its purpose: It quickly showed us that »Schein« was fun!

From scratch

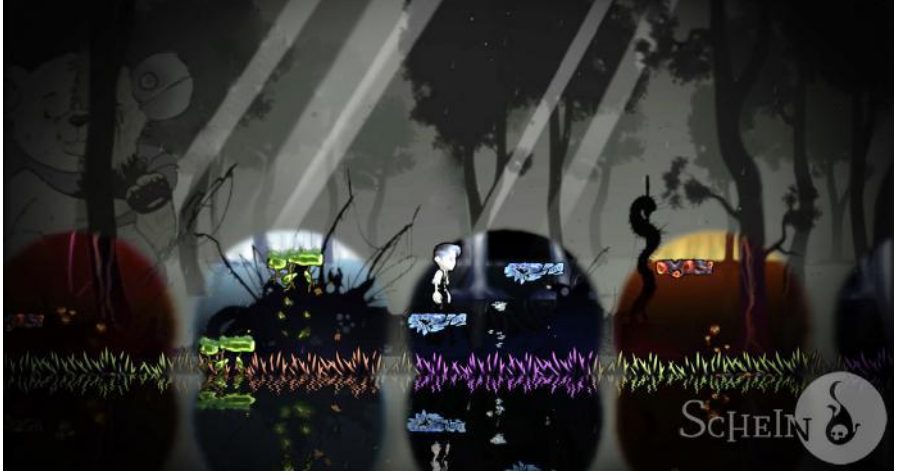
The end of the concept phase also marked the end of the semester and the beginning of the summer holidays. The following semester would start straight with the actual development of the game. At this point however, it was not yet decided which engines we would be allowed to use for the implementation. So

what to do? You are highly motivated and have a full summer holiday in front of you. You study Game Engineering and Simulation Technology and have nailed some things with DirectX and C++ before. Of course you go crazy and start to code your own engine – nobody can deny you this when you are studying programming. So after some brainstorming, which involved some late-night beers and lots of small bits of scratch paper, we started to work on our own game framework. The plan was not as naïve as one might think. We challenged ourselves to create a simple but functioning base for the future 2D platformer. The deal was as follows: If we managed to have it ready by the end of the holiday, we would continue the work until the bitter end. Otherwise the code was to be discarded and we would fall back to third-party engines. It was a great amount of work, but when the next semester was about to start, we had accomplished the task: The engine only had a coarse circle as player model and simple rectangles as platforms, but it represented a functioning base for every 2D platformer out there.

It rapidly evolved from a couple of jumping black and white balls and rectangles to a colored world with the light mechanics of »Schein«, and soon it even featured a newly animated player character. From here it was only a couple of small steps to the fully working game engine. Admittedly some of the steps were bigger than expected, but our team of five processed the tasks one at a time, while simultaneously the actual gameplay was being implemented.

Autodesk's level editor

In the end everything fit together: a functioning collision system, menus and game overlays, an adaptive sound engine, and last but not least, the level editor. In the case of the latter we decided to build upon an established and proven third-party system: Autodesk Maya. So instead of creating our own level editor, which might have doubled the development time of the engine, we started off with a small script that would allow us to build a level out of simple quads in Maya. The engine was then able to load the quads' data and place game objects at their positions. During the development of »Schein« the level editor script evolved as well, and what had started with bright rectangles on a dark background soon obtained an actual likeness of the in-game scene. The engine only added the background and some fancy shaders. This meant that every stone, every tree, every little part of decoration had to be placed by hand. Although this is a good thing, since it makes every part of the game feel unique and handcrafted, it also requires a lot of time. Luckily we were able to rely on a program that offers a well-established set of features and allows a good amount of customization: Autodesk Maya simplified the level design and creation process immensely, and after several months of repositioning platforms and tweaking levels, the game was finally complete.



Zeppelin Studio decided to start working on a proprietary engine to achieve a unique look for their game »Schein« and to have full control of the source code. This way the team could make every single bit of the game fit together perfectly.

The team came to the conclusion that CryEngine isn't very convenient for prototyping a 2D puzzle platformer. Later they were really surprised when Merhunes put its platformer »Rolling Sun« on Steam that is made by using CryEngine.

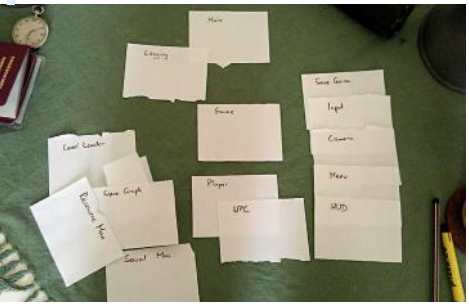
Many pros ...

The whole development process was a thrill ride with innumerable ups and downs. Creating a game that is built on an in-house engine, which is still being developed while the game is being made, is never a simple task. So let us get back to the big question of why we took up this challenge in the first place. The first reasons were rather personal, or specific for this unique case: The University did not define the constraints for the project early enough. Were we allowed to use Unity? Would it have to be the CryEngine, or even Gamebryo? Thus we chose what seemed to us to be the safest bet: No University could say anything against a team building on its own framework. And hey, we were students – we had enough free time and we were supposed to experiment after all.

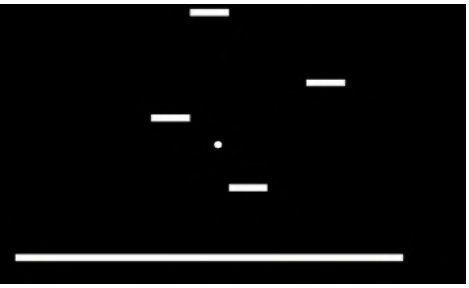
As personal as this initial reason might have been, we soon saw real advantages of our decision. When you start a project from scratch you are responsible for every part of the code. On the one hand this is a great responsibility, but on the other hand, when you are the one deciding over each part, you can make every bit fit into the other seamlessly. Since the engine was being built by a single team from the first iteration on, the separate systems soon start working like a well-adjusted clockwork. This becomes

The series at a glance

- Part 1
  The development of »Schein«
- Part 2
  Creating a proprietary engine as an Indie



This photo shows the result of the first brainstorming session between Michael and Philipp on the engine architecture before they started diving into coding their engine.



The first milestone in the engine development does not look very sophisticated: jumping circles that collide with rectangles, which reminds you of Pong.

even more evident the more unique your game mechanics are. When you use a third-party engine, in many cases an especially creative idea automatically leads to crippling some major part of an engine like – in our case – the collision system. And most of the time this is the point where serious problems arise.

When you walk through a game exhibition hall these days, or take a look around the Steam store, you will recognize lots of games where you instantly sense Unity shouting at you. So many Unity games simply end up having a lot in common visually and are immediately recognizable as such. Not that long ago the same could be said about the Unreal engine and it was not such a big deal. However, this concerned mostly AAA titles that could cope with this easily. But as an indie developer, having your game look or feel like so many others is pure poison to your success. Surely you don't need to write an own engine to create a special sensation, but when you do, you can really be sure that you will create something unique.

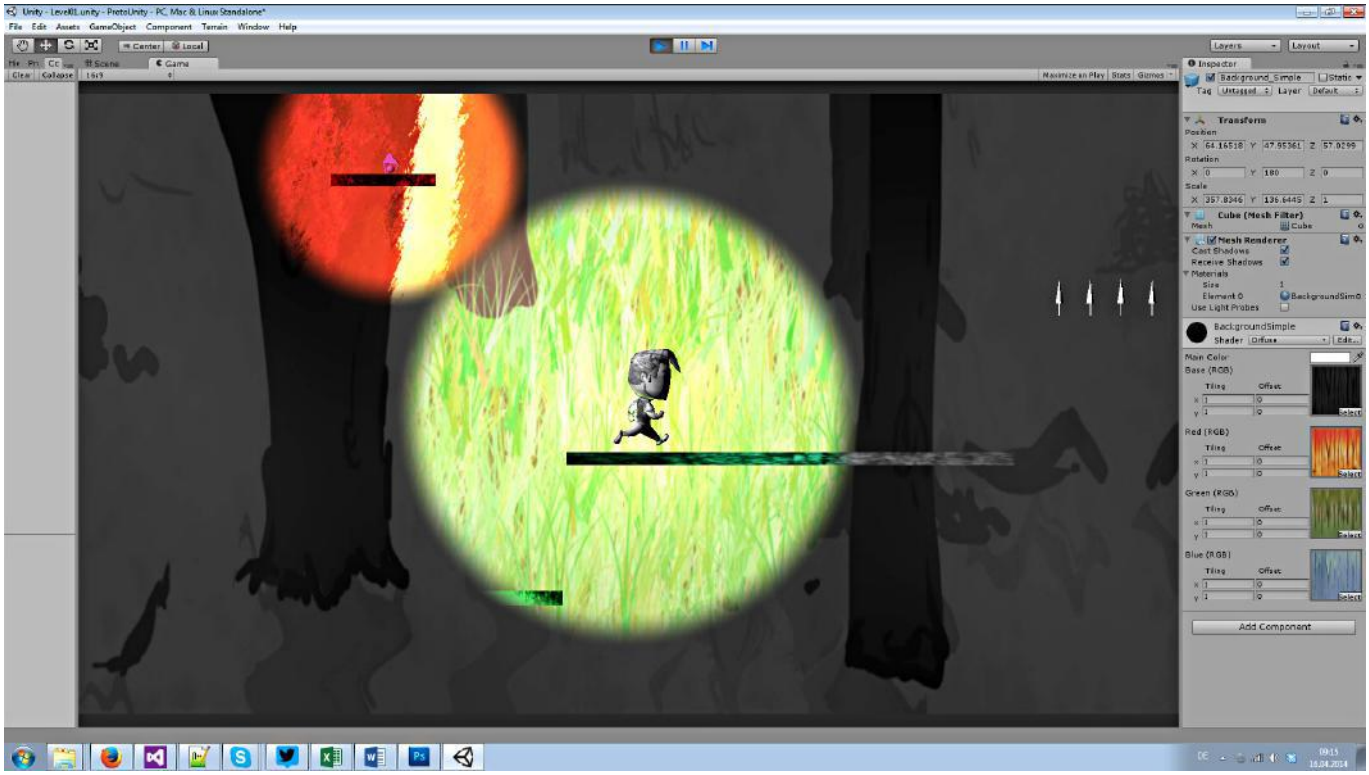
Right now nearly everything you need already exists as a Unity plugin. However, creating all the parts of your engine by yourself gives you much more control over it. Not only in terms of the required outcome of a system, but also in regard to its performance and its use of precious resources. Furthermore, being able to dive deep into the system opens up some nifty and sometimes even dirty paths for optimization.

... and many cons

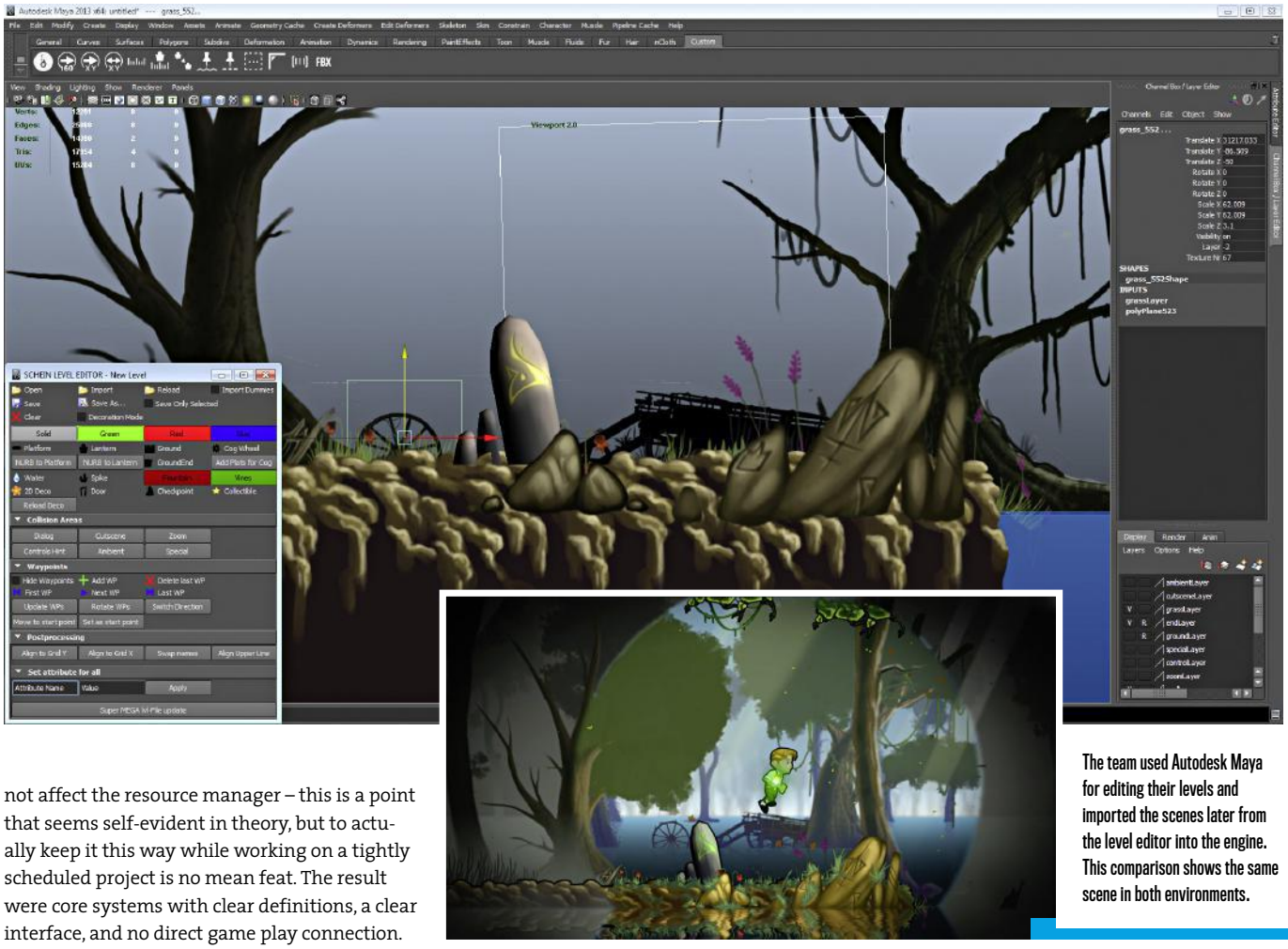
Unfortunately this leads us directly to the disadvantages of our course. Writing everything on your own takes up way too many working hours that could have been spent on the game itself instead. Christer Ericson's book »Real-Time Collision Detection« sure is a blessing and makes the world a better place, but when you have to twist and turn the collision algorithms in your mind for so long, collision becomes the single most annoying part of the engine, where a simple bug can cost you several days of your time. And just when you've fixed it, a new bug appears somewhere else. We suppose that all readers with a slight insight into programming are quite familiar with this problem.

As it usually happens, requirements can change rapidly. When we tested the game and gathered public feedback, we ended up having to adapt our game design and of course, deeper changes in the game had to be reflected in the program's behavior. Very often these changes have to be made right before a strict deadline, so they have to be done as quickly as possible, which often leads to nice looking code being replaced by some quick and dirty code lines. Once the new feature worked however, everyone would forget to smooth things out again and so the code will grow uglier and uglier for an indefinite period of time. Fortunately we made a huge revision of the code close to the end of the project and cleaned up most of the dirty workarounds – better late than never.

One thing that helped a lot, and which is the basis for staying sane throughout the development process, was the modularization of the engine. The different parts were kept as separate from each other as possible. That meant that changes in the resource loader did



A first prototype of »Schein« was developed by using Unity and the team really liked how quickly one could work with this engine. But at the same time they felt that games based on Unity often look too much the same.



The team used Autodesk Maya for editing their levels and imported the scenes later from the level editor into the engine. This comparison shows the same scene in both environments.

not affect the resource manager – this is a point that seems self-evident in theory, but to actually keep it this way while working on a tightly scheduled project is no mean feat. The result were core systems with clear definitions, a clear interface, and no direct game play connection.

This leads us to the final disadvantage, which, now after having released the game, we experience the most. Despite all the modularization and abstraction of many low level systems we managed to achieve, the engine is still implemented as a game framework for Microsoft Windows PCs only. We kept receiving inquiries about our plans to release the game for Linux and Mac OS X or were asked to port it to consoles and mobile. Unfortunately each of these requests implies to rip away the foundation of the engine and this would mean at least another six months of development. While this is possible of course, it still depicts the major problem of having used our own framework instead of existing solutions that offer faster porting possibilities to a vast number of devices.

All in all a good choice

In conclusion we feel that creating one's own engine is a good thing. It takes a huge amount of work and time, but when you are implementing the engine's components one after the other, it can be a benefit for all future games that are built upon the system. Your own framework is sure to give your games a unique look and feel and it can hopefully cope with any weird mechanic they require. Just make sure to add appropriate abstraction layers and to separate the individual components accordingly, so that you don't end up re-coding the whole thing when one module becomes obsolete, or has to be adapted to new platforms. You never know what extents your project may reach some day.

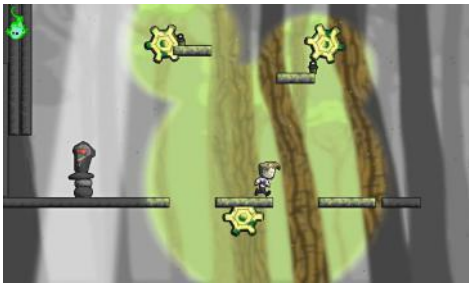
Here is a list of pros and cons that sum up the experience we have gained through creating our own engine:

+ You can make all the core systems fit together, no matter how crazy your game idea is.
+ You make sure to create a unique base for your look and feel.
+ You are in control of the game's computational performance, for better or for worse.
– Building your own engine takes time – be sure to have enough of it.
– From great power comes great responsibility – it will be you who is to blame in the end.
– Porting can be a pain when it comes to changing low level components – make sure to modularize and use abstraction layers.

For our own next project we will have a very different approach when it comes to the choice of the engine. After all we are not students anymore. We will make sure to carefully evaluate a wide range of possibilities. At the moment we tend to think in three categories: If the project is small and requires highly unique engine code we'll stick to our in-house development. If it's a mid-sized game we might use Unity and for some fancy 3D project we would go and check out the Unreal engine. But no matter which choice we end up making, we will make sure that every part and every pixel of our next project retains the unique style of our games.

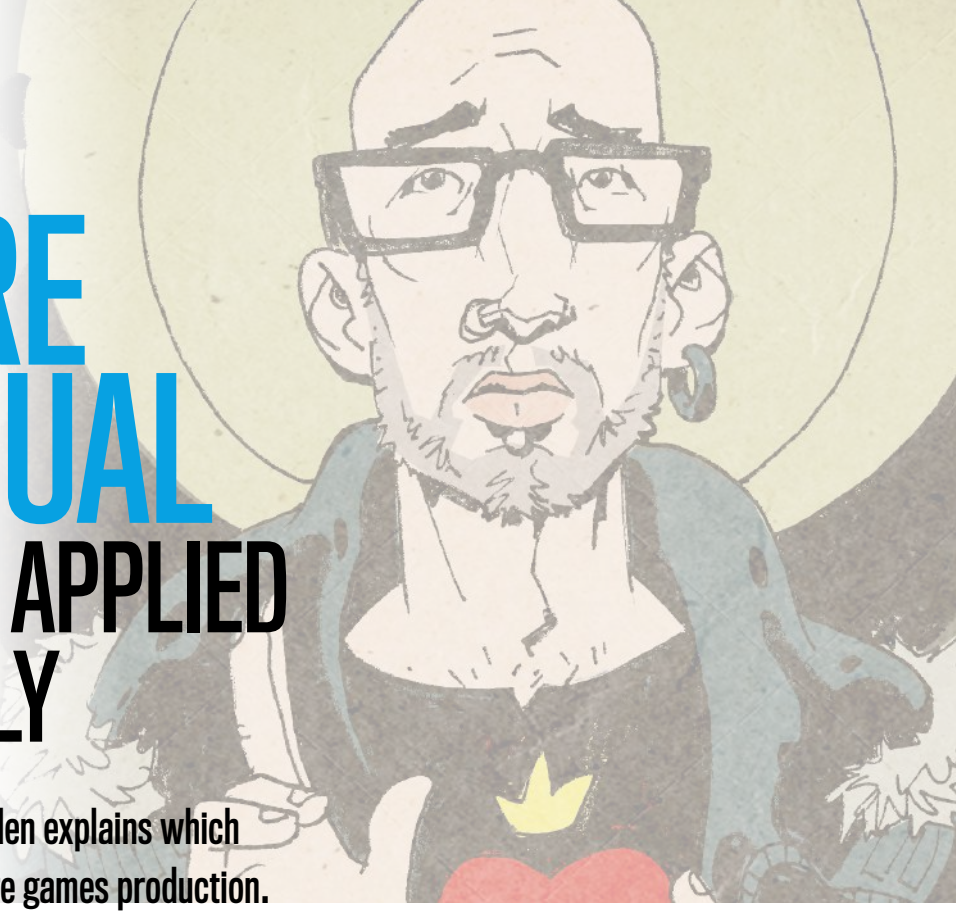
Michael Benda, Philipp Schaefer

The state at the start of the implementation of »Schein«, featuring gravity, collision, sound and a parallax scrolling background. Still there was a long way to go to reach the final look of the game.



When the development of »Schein« ended at University the game (already running on the self-coded engine) still looked a lot different from its final quality.

FROM CORE TO CASUAL
HARD LESSONS APPLIED CASUALLY



Former Rockstar Art Director Ian J. Bowden explains which lessons he learned in twenty years of core games production.

Ian J. Bowden is Art Director at GameDuell.

In 1997 Ian J. Bowden founded Mobius Entertainment where he was Art Director on over a dozen handheld and console games. In 2005 Mobius became Rockstar Leeds and Bowden left his mark on outstanding AAA productions such as »L.A. Noire«, »Red Dead Redemption«, »Max Payne 3« and the »GTA«-series. After two decades in the field of core game production Ian is now Art Director at GameDuell to boost the visual style for their social and mobile titles.

Twenty-two years in games. Two whole decades. It came as a blinding revelation to me when I realized it in 2013. In Autumn 1993, I lucked into my first junior artist job in a tiny studio called Hookstone Ltd. that was situated in the post-apocalyptic wasteland of North Yorkshire, UK. It was the year that we were playing »Sam and Max«, »Doom« and »X-Wing«. Production values were pretty low still, but improving year on year and the games industry was beginning to have pretentions towards the mainstream. Like most of the UK-scene at that time I sort of fell into the business. There was no real plan, no dreams of world domination through the medium of the pixel and the polygon (except, perhaps, in »Sim City 2000«), just a desire to be involved in something more fun than a regular office job.

You see, I had graduated from Leeds University with a degree in English and Art History a year before. With such illustrious qualifications my options were teaching or wearing a garish, itchy nylon uniform and asking if »you want fries with that?« You can probably imagine my delight to have dodged either of those particular bullets and found my way, by luck or fate, into a creative industry for which, back then, there were no real qualifications necessary. Perfect for me since neither English nor Art History are real qualifications.

So what do I have to offer from 20+ years with a track record that covers huge franchises like »Star Wars«, »GTA«, »Red Dead Redemption« and »Barbie« (Yes, Barbie. I did a horse riding game for Mattel starring the immortal plastic bimbo.)? What did 20 years in the games industry make out of me? Now I have male pattern baldness, too much grey hair and a weakness to migraine headaches. And a whole heap of experience.

Mistakes? I made them all ...

Just over a year ago I left the AAA business and found that I was looking for a different challenge which presented itself through an inspirational industry veteran called Todd English, leading the social and mobile games unit of the Berlin-based games company GameDuell. I wasn't sure that I was the best fit for a casual game setting, but Todd, who also just recently joined GameDuell from Popcap where he was Head of Studio for the Asia Pacific Region, convinced me otherwise. I believe the hard learned lessons of those decades of AAA development as well as some of their success factors can be transferred and applied to the casual games production. On the 1st of August 2014, twenty-one years after my first taste of game development, I began to put them into practice in my new studio at GameDuell.

The Eisenhower Plan

I'm sure most people are familiar with the maxim »fail to plan, plan to fail«. I've always hated the smugness of that particular cliché. Say it in a whining nasal voice. Make a smug, weaselly expression while you do. That's how it sounds whenever I read it. I want to punch that phrase in its face. I've always preferred Eisenhower's »plans are nothing. Planning is everything«. A plan is rigid, if something goes wrong, if a goal shifts, the plan is no longer

valid. It is the process of planning that is important, everyone knowing their part in the greater picture, a team dialogue creating a synergistic strategy. I was involved in a game which started as a 3D »Tempest 2000«-clone with a robotic spider as the player character which, over the course of two years, morphed into an epic space opera with a mysterious female bounty hunter protagonist caught in a battle between warring factions of aliens ... and then got unceremoniously cancelled. Unsurprisingly.

But how, I hear you ask, could that happen? Three little words: Lack. Of. Planning. We simply did not understand our goals, or how to achieve them. If, like Ike Eisenhower, we had done our planning, everyone could have moved forward semi-autonomously towards finishing the game that we wanted to create. Instead there was the pernicious spectre of unchecked feature creep. There was no one who said: »But hold on ... what about the spider?«

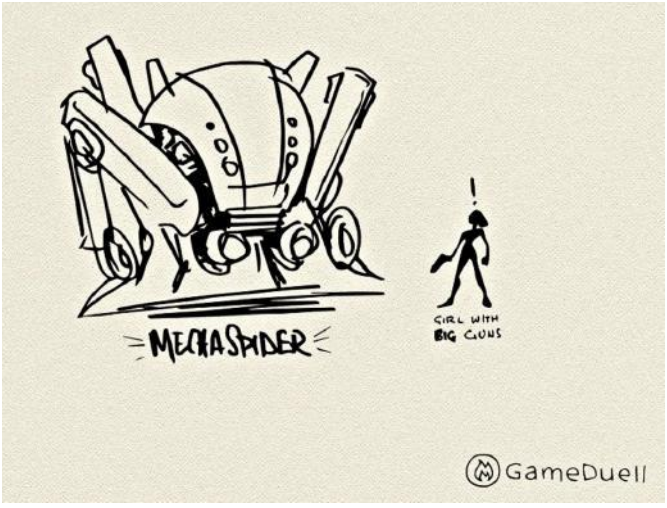
In casual games, working with a small team, effective planning and pre-production is vital and achievable. In all the projects that I'm currently involved in, the GameDuell proprietary development engine and the two games currently in development, we have intentionally reserved pre-production a major chunk of the development timeline. During pre-production all team members are involved. The team is informed about every new idea and how it fits into the whole. Team leads communicate and discuss the entire plan with their team, everything is mapped out and goals are clear. Individual team members can see the deviations to the plan and red flag them. The synergistic creative benefits of the time we consciously take for exploration, research and planning are palpable. Our teams are also aware of the other development projects in the company, which stimulates creative input across actual team boundaries. This way our projects profit to a large extent from the lessons learned in the pre-production and planning phases of the others.

Details: Good games are like great architecture

So, good games are built on the solid foundations of planning. However, Mies Van der Rohe, one of the great pioneers of Modernist Architecture said that either God, or the Devil, »is in the detail«. His body of work not only paid attention to the gestalt, but also to the tiniest details. If a game artist wants to make something truly outstanding, they should learn from Mies van der Rohe and consider everything as gestalt. In open-world games like »GTA«, the reality of the world is sold by the attention to detail. I've carried a sketchbook for many years, recording faces, buildings, the minutiae of life. It is a device to train the eyes to see what makes a world convincing and reproduce these details in another medium. Get the little things wrong and you can undermine the players' suspension of disbelief, break their immersion.

Want to make a believable room? It's not just about clever lighting models or complex shaders. An artist needs to think like an interior designer, an architect, an electrician and a builder. Where would the plug sockets go? Is your room in Europe? In the US? Those electrical outlets will look different in different regions. Are the doors the correct height for a human? How high does a door handle have to be? Basic details to get right. Ignore them and the room will look weird, awkward. The player may not be able to tell you why, but the vague sense of wrongness will make them uncomfortable and mar the experience. But do you want to make a room that feels real? That's where it gets fun. We'll age the room, add scuffs and stains, dust and imperfections. The skirting board that doesn't quite join up at the corner, leaving a gap. The slightly off angle of the light switch. There are no truly straight lines in nature, to the constant dismay of the modern architect. The artist next has to imagine the person who lives in that room. They must inhabit the room in their imagination. Arthur Conan Doyle in »The Sign of Four« creates a

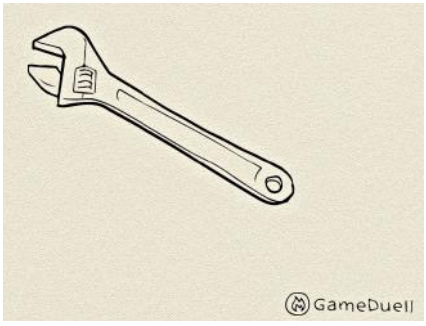
In the early nineties, most developers in UK became part of the games industry driven by the desire for something more fun than a regular office job.



Over the course of two years, a 3D »Tempest 2000«-clone morphed into an epic space opera due to lack of planning.

watch, full of little details that Sherlock Holmes uses to deduce the character, habits and the eventual death of Watson's brother. The details tell a story. Human beings have evolved incredible pattern recognition skills and, as artists, we can leverage this ability to trick the player into reading the narrative of a room. We can tell a compelling story with the most subtle of details, and even the most anodyne story of suburban domesticity is better than no story at all.

Want to make a house? A city? A world? Extrapolate from that. In the casual sphere, the games are simply collections of details to be nailed. The basis of a successful casual game is broad appeal. Look at the »Candy Crush«-franchise, »Angry Birds«, »Bejeweled«, »Plants vs. Zombies«. These games are consistent, instantly readable, never producing cognitive pops in the player's mind. They are truly consistent brands, they have the details nailed down. They may not have the prolonged play times of a »Mass Effect« or »The Last of Us«, but the production values and polish are analogous. A bad choice of typeface, an ill-conceived icon or a confusingly coloured button can make the difference between an engaging and fun experience and spoiling your audience's enjoyment, leaving your app un-tapped and languishing; uninstalled to make room for the next »Flappy Bird«-clone. We make our little worlds for our players to inhabit for a short time, on their commute, while waiting for a friend.



The waste of time and resources of having an artist manually manipulating assets can be avoided with the right tools.

Give me tools!

Coherent worlds have to be created, by artists. Archimedes said: »give me a long enough lever and a place to stand, and I can move the world.« You see, tools are important. Running a small art department producing the dizzying amount of high quality content that is expected of a modern AAA game is near impossible without good custom tools. Give us a good tools programmer, a place to sit (and an unlimited supply of coffee and pizza) and we can make you a world. It is all too familiar to find that an artist gets used to an awkward work around, no matter how much time it costs, manually manipulating assets. This waste of time and resources can be largely reduced when you have a good tools programmer or ideally a technical artist writing a tool to automate the process. Every small piece of mindless drudgery that is excised from the process leaves an artist a little more time for creativity, a little more time to polish.

Even with small teams, high production values are achievable; our audience deserves no less. At GameDuell we are creating a toolset and pipeline that removes the grind from the creation of assets for cross-platform development. Using a proprietary engine, written in HAXE, and after months of research and testing the best commercial tools available the artists are free to create assets which will be converted seamlessly to all target platforms at build time. Obviously, to create the perfect user experience, each version is treated by QA as a discreet entity and the art is modified accordingly, but the simplicity of the process releases the art department to polish the versions to the highest degree.

When I moved to Germany towards the end of the glorious European summer in 2014, I was ambushed in a bar by some industry-types who plied me with alcohol in an attempt to get me drunk enough to spill the »secrets« of the creation of the terrain of the »GTA«-games. It being Berlin, where a bottle of beer is cheaper than one of water, they easily succeeded in their first goal but not the second. Through the pall of smoke in the dimly lit bar I could see the looks of bewilderment and disappointment forming as they posited theories of what landscape generators they believed were used, what procedural terrain techniques had been employed, how the vegetation was generated, how the erosion channels on the mountains simply must have been done with World Machine … Each theory was in turn, denied. The simple truth, the elusive secret, was that every square metre of the games had been built by hand. Built by artists who cared about their craft and had total commitment to making something that was not just »good enough«, but better than everything else. Their time was maximized by optimized builds, integrated source control, good tools that don't create the geometry or textures but free the artists to actually craft them.

The attention to detail is essential to make a game world seem realistic.

But people make the real difference

When we hire new team members, what do we look for? Artistic and technical skill, obviously. Portfolios are the first thing I look at as

an Art Director. To waste those skills in tedious donkey work is not only a bad use of resources but an effective morale sapper. About a decade ago, I presented my first lecture where I likened the games industry to First World War trench fighting and my team to a harried squad of soldiers. Basically some kind of half-baked »Band of Brothers« metaphor. In retrospect, I'm rather embarrassed by the arrogance of the talk. An art department is not an elite fighting unit. Nor is it a machine to make art (although, when it works properly, it can be an awesome art machine.) No, at best it's a family. We spend at least eight hours per day working, talking, laughing, and playing. Add to that the occasional late nights at the office or after-work beers or cinema and we spend vastly more time awake with our colleagues than we do with our actual families during the working week. It is imperative that there is total team cohesion and that begins with the interview process. Cultural fit cannot be underestimated. I would rather hire a good artist who will benefit the team dynamics than a great artist with an ego problem. Unlike oysters, irritants in a creative environment never produce pearls.

Around 1999 a young texture artist came for an interview at the converted church that used to be the Mobius Entertainment office in Leeds. We didn't really need a texture artist, we were hiring for modelers, but this interviewee was so charismatic and fitted so well, that I forbade him to go to his next interview that same afternoon and attempted to hire him on the spot. Within a month, the guy who at his interview had said »I can't model« was building level geometry for the ill-fated »where's the goddamn spider?« game. Fast forward a decade and he was the Lead Artist at the Rockstar Leeds studio, instrumental in the success of the »GTA«-games, later to build the Activision Leeds studio from scratch. He is now Art Director and co-founder of his own studio.

Some of my best hires have come first from gut feelings about team fit teased out in rambling, seemingly-chaotic interviews. A prospective new team member, who settled into the interview, could answer random questions like »what is the correct colour for a pack of cheese and onion crisps?« or »take two animals, splice them together, make a better animal. What is it and why is it better?« and still maintain humour and composure usually got a job. The trick was then to find the place for our new recruit. Good staff, whatever position they were initially hired to fill create a uniquely shaped niche that they can fill and be comfortable in. With the texture artist, there needed to be some encouragement, but he achieved more than even he could have predicted. A lucky QA hire eventually became a senior artist, a junior modeler became the studio's Senior Technical Artist. How could that happen?

An art department and furthermore a games studio is a creative beast by nature. The »machine« analogy is flawed. A machine is simply the sum of its parts. A studio is much more than that. Correctly nurtured and given space team members can grow personally, take ownership, drive projects in unexpected new directions. At Rockstar Leeds, the artists would have field trips. We would bail for the afternoon, have lunch and go to one of the galleries or exhibition spaces in the city, exposing ourselves to new visual experiences and to the wider world of art and creativity, usually ending up with an argument in a pub about how someone's »six year old kid could do better than that«. Now, with my new team at GameDuell, we've taken this to its logical conclusion, a weekly optional creative slot for the artists (or any other interested team member) where there is the opportunity to take part in organized activities from workshops on illustration, through liv e drawing, to a hilariously chaotic large format white-board version of multi-lingual Pictionary.

An art department and furthermore a games studio is a creative beast by nature.

Lessons learned

And so ... after more than two decades of working in the games industry, watching it mature into a medium with real cultural weight, seeing development of new genres and new ways to interact and play and being fortunate enough to have worked on some of the most influential games in the history of the form, I have learned what, precisely?

That planning is important. That details can make or break the experience. That we should invest in tools and in people. Also that Domino's pizza tastes great cold in the morning and too much free espresso stops you from sleeping and makes your jaw ache. Oh, and that there is a correct answer to the spliced animal question. It is »Spider-Chicken«. Obviously. **Ian J. Bowden**

COMRADES IN ARMS
WORLD OF TANKS STRONGHOLDS

Battles are about team cooperation – historically as well as in video games. To get players working together, Wargaming has implemented Strongholds in World of Tanks. Anton Pankov explains how the initial idea transformed from a paper turn-based card game into virtual battles.

Anton Pankov
is Executive Producer
at Wargaming.

Anton manages the organization of »tank« projects life cycles in the CIS region, including »World of Tanks PC«, »World of Tanks Blitz«, and »World of Tanks: Xbox 360 Edition«. He maintains products' quality service and ensures effective communication with the community. Anton is also involved in the overall development, publishing and promotional plans in the region. He started his career at Wargaming as a casual »World of Tanks«-player, worked as a Community Manager and reached his current position in five years. Anton is a strong believer that any product that is created for users belongs to them as well, not just to developers. It means players and community should be at the core of the game development process, not vice versa. Anton has a Master's degree in Archaeology, a Bachelor's degree in Pedagogics and dreams of creating a game that will replace an undergraduate program.

The idea of having clan content in »World of Tanks« emerged long before the release of the game. While working on the concept of the game, we clearly realized that we'd need gameplay functionality for communities and clans. One important lesson that history tells us is that battles are about team cooperation, and our game is above all about working as a team on every level, from eSports pros to amateur players.

Historically, tanks never engaged in battles as single units but as platoons, divisions, etc. and we needed a structure like that in our game. That's why the first thing we thought of before the release of »World of Tanks« was clan functionality. The initial version of it was quite simple: a clan could have no more than 100 members and its primary function was to encourage clan members to play together as a part of platoons (up to 3 members) and companies (up to 15 members).

The Globalization

In 2011, we introduced more wide-scale clan functionality with the Global Map. There's no surprise we chose this path because MMO games are being dominated by clan content rather than solo content. That's because everybody knows about popular clans and wants to become a part of them. Basically, clans are the elite of any MMO project. On top of that, clans attract the most organized players that are prepared to devote their time to the game; they'll make a career out of it. These people are the most dedicated, loyalist and engaged part of our fan base that have been with us since the start of the project.

After releasing the Global Map, the solution of any territorial conflicts was, of course, narrowed down to tank battles. However, we gave our players a lot of room for diplomatic moves, finesse, treaties – in short, we provided decision-making tools that enabled players in some way to create the game of their own. They could create alliances, decide to battle on their own, or take out adversaries.

Due to the Global Map's turn-based nature, it was deployed on an external web resource rather than the game client. In some way, it reminded us of the »Total War« game series. The important part of the Global Map was that the players and clans knew the exact time and date they needed to show up for their battles, as well as the schedule of their weeks-long Campaigns.

These long battles on the Global Map were quite hardcore. In its infancy, the audience was quite niche, but as time passed, we realized that the number of clans had grown exponentially. The Global Map neither by size nor by

functionality could accept all the users who wanted to play on it. Finally, the Global Map turned into a Hall of Fame of sorts with only top clans battling on it. It became evident that we needed to add a mass-content element to the clan content to make it evolve.

We did a lot of research and decided that the optimal decision would be to allow each clan to build its own base with numerous structures that it could further develop and get bonuses from. That's the basic idea that was the foundation of Strongholds.

The Challenge

At the very beginning, Strongholds was a paper turn-based card game. We got together at night, threw the dice, moved the markers and step-by-step tried to discover the gameplay that we finally introduced in 2014.

But transforming this from pen and paper to a full-scale game wasn't easy. We had a lot of challenges on our path to release. First, we were slightly worried that Strongholds may cannibalize the Global Map, as the core audience of both Clan Wars modes, according to our research, was similar.

Taking into account the fact that the Global Map was web-based rather than client-based, the main development challenge was to transfer the clan functionality to the »World of Tanks«-game-client. Most of the time spent on development was dedicated to the integration of clan content into the game client and implementation of the economic strategy algorithms into the game.

The first internal testing (not taking into account the paper prototype we had at the very beginning) was conducted six months before the release. On top of that, we conducted a few closed focus tests among some loyal fans. They tested Strongholds thoroughly and sent us their feedback. Based on it, we incorporated quite a few changes into the gameplay of Strongholds.

For instance, we initially had four zones to be protected during the Defense Period. After we tested this feature, we realized that the clan needs 60 players (15 for each zone) to defend all the zones. Based on player feedback, we reduced the number of zones to just one to attract the interest of smaller clans and a wider audience.

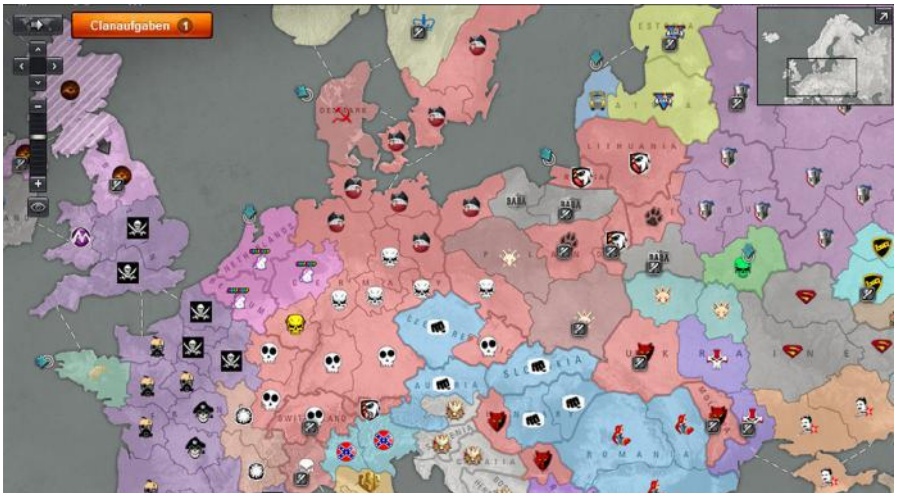
Also, the players helped us to finalize the format of Strongholds' divisions: Medium, Champion and Absolute, with Medium using vehicles up to tier VI and minimum of 6 players, Champion having vehicles up to tier VIII and 8 players minimum, and Absolute Tier X and 12 player minimum respectively.

Another feature that took a long time to develop and test was the War Department responsible for special instructions. Players' feedback helped us make a number of significant decisions regarding this in-game structure. After completing closed focus tests, we conducted a number of »supertests« and then released a basic version of Strongholds as part of Update 9.2 with an upgraded version to follow in 9.4.

The initial version of clan functionality was meant to encourage clan members to play together as a part of platoons and companies.

It took us 18 months from the first idea of Strongholds to implementation and release. During this time, Strongholds went through a countless number of changes and the version introduced was completely different from what we played on a board with dice and paper. The main drivers to evolve the games' content though were players' behavior, requirements and preferences.

We can proudly say that Strongholds meets all the needs the clan community formed over the years of playing »World of Tanks«. We took into account all the weak points of Global Map and provided the opportunity for players to enjoy clan content constantly, not on a Campaign basis only.



The Global Map was launched in 2011 and deployed on an external web resource due to it's turn-based nature. It allowed players to schedule their campaigns, but couldn't accept all the users who wanted to play on it.

The Essence

In short, a Stronghold is a military base of a clan within the game client. A clan Commander may create it and build structures on it that will further bring the clan various bonuses. The special currency called »Industrial Resource« was introduced together with the Strongholds mode to create a complete ecosystem. Clans may earn Industrial Resource in battles.

The Strongholds are military bases within the game client. The battles for these bases are more complicated than battles on neutral ground and result in the defeated clan losing part of its Industrial Resource and damaged structures.

There are two types of battles – Skirmishes and Battles for Strongholds. A Skirmish is a battle between clans on neutral ground, where each of the battling clans can earn a certain amount of Industrial Resource. A Battle for Stronghold is a more complicated type of combat. It's only available to Level V or higher Strongholds. Every clan must switch off the Defense Mode for an hour and be ready to defend the Stronghold. The clan losing the Battle for its Stronghold loses part of its accrued Industrial Resource and also may have some of its structures damaged or demolished.

The gameplay was divided into two parts for a clear reason – to make Strongholds attractive for the widest share of players possible. The trialists and those who prefer less intense combat may casually play Strongholds on Level IV. Those who prefer an intense experience can challenge the enemies in Battles for Strongholds and earn higher bonuses for their clan.

Another principal difference of Strongholds from Global Map that makes the new mode attractive to smaller clans and a wider audience is the number of choices in things like vehicle Tiers and team capacity.

The Results

We introduced a basic version of Strongholds as part of »World of Tanks« Update 9.2 in July 2014. The basic version allowed clan players to build their Strongholds up to Level IV and take part in Skirmishes. The updated version arrived with 9.4 in November 2014 and allowed the clans to update the structures of the Stronghold up to Level X alongside the introduction of Battles for Strongholds.

Update 9.4 proved our theory that the core of the Global Map enthusiasts would soon become the core of the new clan mode. It only took a few days upon the release of new Strongholds functionality for the best clans to upgrade their bases to the highest level.

Luckily, for us, Strongholds neither took anything away from Clan Wars »elite«, nor cannibalized the Global Map. The total number of Strongholds activated are 140,000 and about 80 percent of those are smaller (up to 30 players) clans trying out the new mode.

Strongholds have quickly become a popular mass mode, engaging 2-3 times more players than the Global Map. The latter remains an elite club for the largest and most organized clans. For some, the Global Map is the main focus of top tier »World of Tanks«-players, and Strongholds are a great opportunity for smaller clans to try out playing as a team. We fully encourage our players to join clans and socialize that way because our study shows that clan socialization helps players to evolve more rapidly, feel more comfortable in the game and, eventually, stay in our projects longer.

In Strongholds, we decided to offer our players a third key gameplay element (besides diplomacy and battles) that was lacking before – economical. Having introduced Industrial Resource, we allowed them take their in-game activity to the next level: plan how much of this currency they need to get in battle or how much they have to spend on a certain structure.

As I said, we spend a lot of time preparing the basic Strongholds version. It went through a high number of inner iterations and external user testing, which means that players were well-informed about the game and knew what to expect. Anyway, we were flattered to see that many of our players could not believe their eyes and that Wargaming introduced the economic strategy built into the client of the game initially positioned and promoted as a »slow MMO-shooter«.

We were a bit ambivalent about the initial activity of the players upon the release of Strongholds in July 2014. From one point of view, we released Strongholds in summer to avoid overcrowding the mode at the very start. Looking at it from a different angle, we were slightly disappointed with the numbers. But soon the players' activity in Strongholds began to grow making it no problem at all.

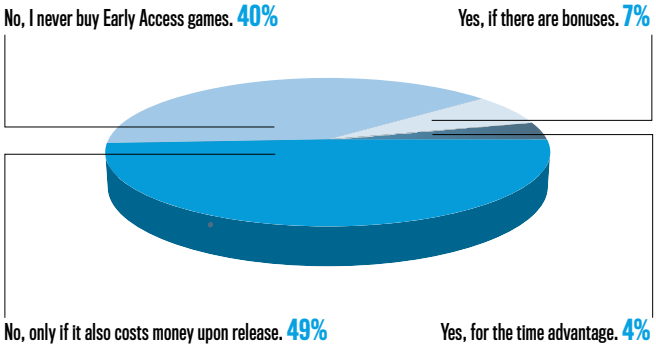
It was quite a surprise for us that Strongholds substantially lowered the activity of platoons, which were initially our most popular team mode. The trend shows that sooner rather than later Strongholds will finally substitute platoon battles.

Anton Pankov

DATAFLOW

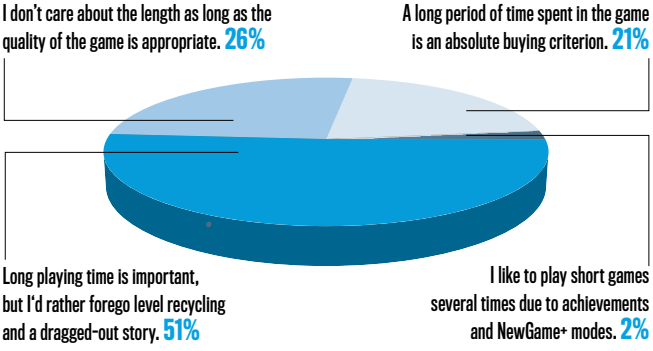
Our consumer magazines GameStar and GamePro are regularly asking their readers and website users about current topics of the games industry. Here's a small excerpt.

Are you willing to pay for an Early Access game that will become free2play upon release?



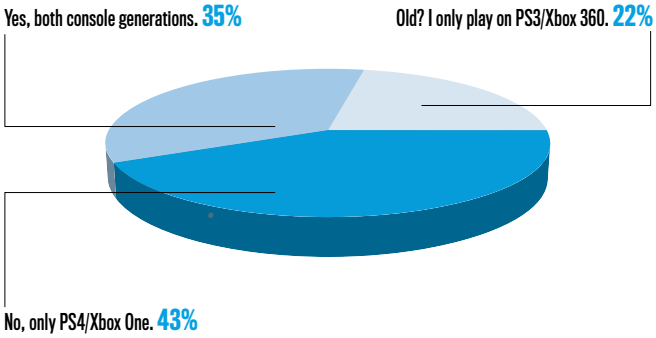
Source: Poll on GameStar.de, 14,287 participants

How important is the length of a game for your purchase decision?



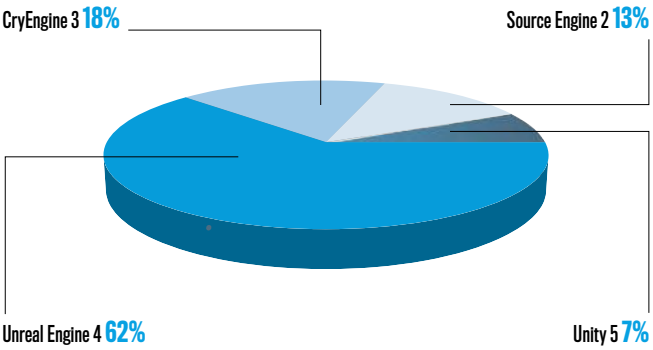
Source: Poll on GameStar.de, 21,853 participants

Are you still using the old consoles PS3 and Xbox 360?



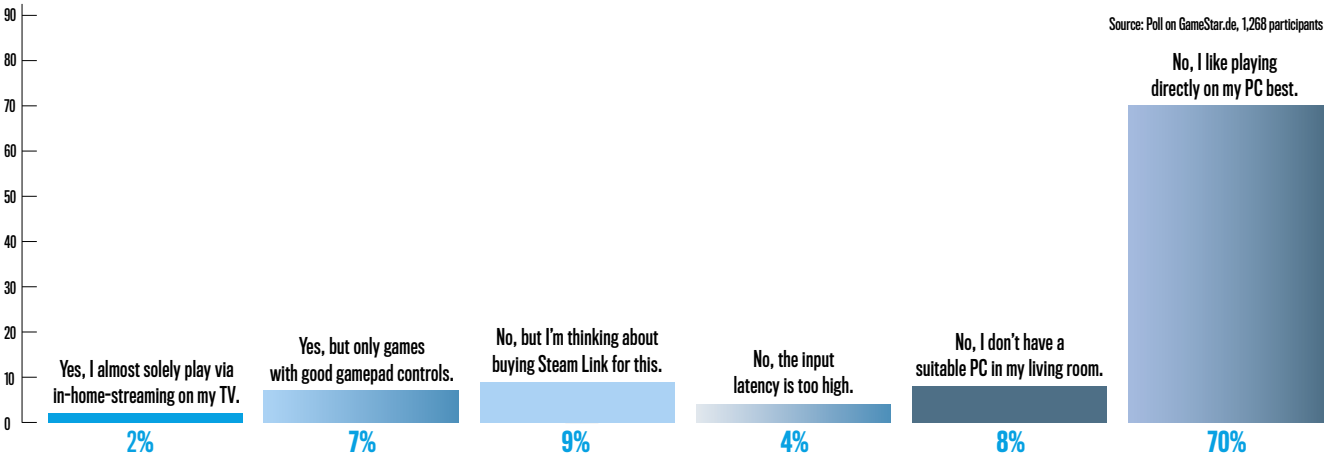
Source: Poll on GamePro.de, 8,504 participants

Which graphics engine demo shown at GDC did you like best?



Source: Poll on GameStar.de, 4,612 participants

Do you use Steam's in-home-streaming to play PC games on your television?



Source: Poll on GameStar.de, 1,268 participants

TRAVELOGUES FROM A VICTORIAN ADVENTURE

The Adventures of Bertram Fiddle was the first game for the team at Rumpus. Alexander Birke recounts the development process and how weird noses, British humor and bad puns come into play.

Alexander Birke was Programmer and Game Designer for »The Adventures of Bertram Fiddle«.

With a leg in both design and programming, Alexander is focused on how advances in both disciplines can lead to more engaging games. From his experience working on three different adventure games and a bunch of other titles he has also come to appreciate technology that lets everyone on a team contribute their fullest to a production. Alexander is currently busy launching his own game company Out of Bounds and also organizes the local Unity meetup in Bristol.

»Has his nose been eaten by a beetle?!?« was my immediate reaction the first time I saw a drawing of Bertram Fiddle. In November 2013 I learned that Rumpus, a Bristol based animation studio, were looking for a developer to help realize their idea of a traditional point and click game about a dark humored murder mystery played out by weird characters with even weirder noses. Being a longtime fan of everything steampunk, and loving the art style I got in contact with them. I ended up spending the next year doing the programming and game design on what would become the first episode of the game. Since this was the first title that Rumpus set out to make, and the scope was really big, there was a lot of challenges we had to overcome. In this post mortem I will dig into what things went right and wrong, which will hopefully be of help to other small teams that want to produce point and click games or other titles that rely on a lot of content. So put on your favorite adventuring monocle and moustache and let's get to work!

The Point&Click-adventure »The Adventures of Bertram Fiddle« stars the titular Victorian explorer and his manservant Gavin The Cyclops.

Things that went jolly good

Classic adventure games rely on two things to create an immersive experience: An intriguing world you want to explore, and compelling characters that you want to talk to. Seb Burnett, the creator of Bertram, first came up with the idea for the character seven years ago and has since spent a lot of time developing the universe, the characters, and its look.

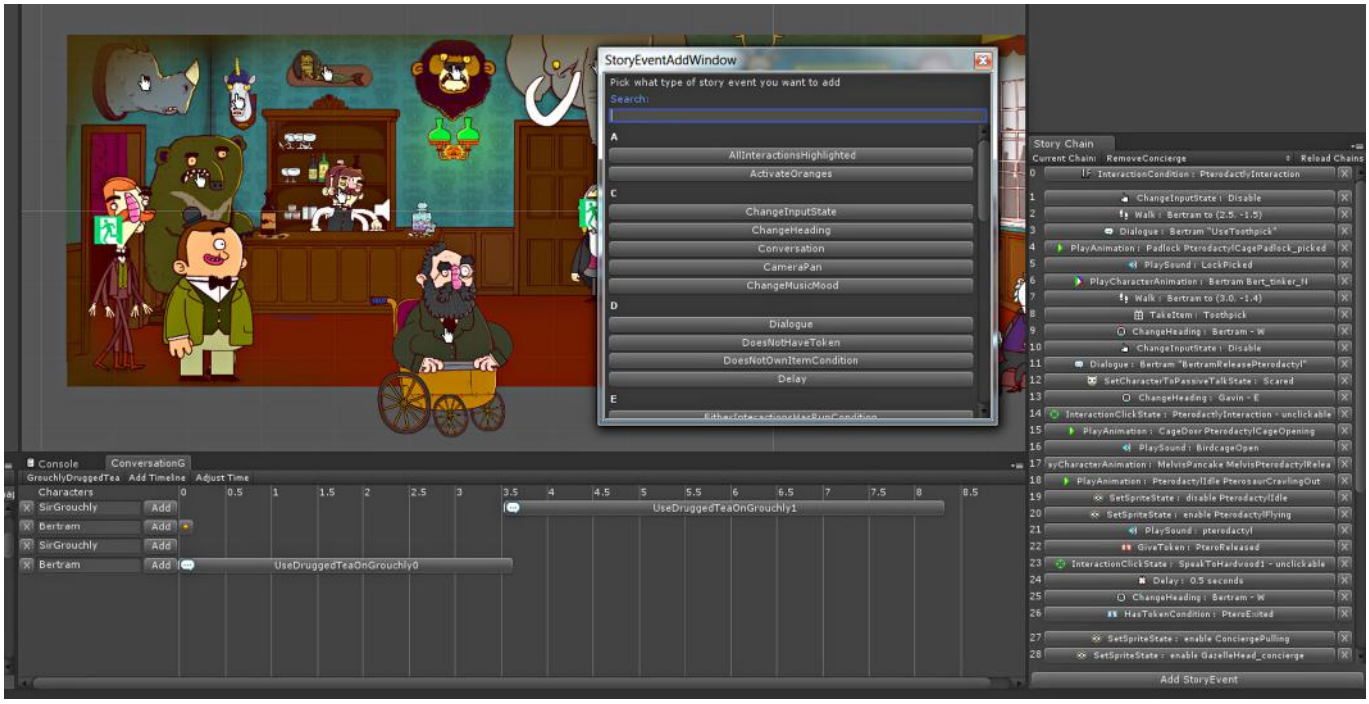
Unique art style and universe

Seb comes from a background in animation and illustration so his influences come from that industry. He also took a lot of inspiration from B movies and Victorian Literature. Having this reference has given Bertram a unique look you have not really seen in games before which I believe makes it stand out from other point and click games. The fact that the universe also had been in development for so long meant that there was less of a need for pre-production of the game since many of the characters and locations had already been thought of beforehand.

Good workflow

Before Bertram, I had already worked on two adventure games. I learned that creating a content driven game with a small team is a daunting task. Whereas big productions can assign more manpower to produce assets, you need to be smart when you only have a limited amount of man hours. What makes matters worse is that where most games can be balanced by tweaking numbers in a spreadsheet, adventures require a serious amount of work to change a puzzle or interaction because you need new assets. How do you go about solving this problem?

One part of the answer is to test early, often, and find faults in the design as soon as possible. The second part of the answer is tools driven development. On »Shadow of Kharon«, the last adventure I worked on, it was the programmers who had to implement the content from the rest of the team. As you might imagine this was very inefficient. If everyone on the team can put in their own content the whole process is much more parallelized and as a bonus you will also have more eyeballs directly on the game which



The core of the gameplay was created with the StoryEvent sequencer and the Conversation Timeline tools.

helps with communication on the team as well. Instead of having to explain how they want something to be inside the game, content creators can simply put it in themselves and get immediate feedback if it works as they want it to.

The core of Bertram's gameplay was created with what we called the StoryEvent sequencer and Conversation Timeline. The sequencer allowed both me and Dan Emmerson, the lead animator on Bertram, to add puzzles and interactions. To start with, the events that could be played back in the sequencer were fairly basic. Such as instructing a character to walk to a point or say a piece of dialogue. But it proved very extendable, so by the end of the project Dan even started to put in his own small easter eggs. A skip button allowed for easy puzzle testing by letting a condition such as using the right item to be skipped. The sequencer would also highlight in red if an event had not been connected properly, which meant it was easy to fix bugs caused by not configuring them properly.

While the sequencer allowed Dan to put in puzzles, the timeline tool allowed me to create cutscenes without having to animate anything. I could simply reuse animations that Dan had already created. Since we used Unity as our game engine it was relatively straightforward to implement these tools, since the Unity editor is very open to adding extra workflow and UI to.

Like digital clockwork

Automation also played a huge part in allowing us to finish the project on time and get a higher quality product than we initially aimed for. For starters, we had not planned to have lipsyncing in the game but just use looping animations of the characters mouths opening and closing no matter what line they were saying. As we started to get dialogue and other animations into the game, it stood out more and more that this looked inferior to the rest of the game. By resurrecting an old open source project called »Papagayo« which was originally intended to be used with Anime Studio, we managed to come up with a workflow that let Dan and me lipsync the whole game, which is about 1,200 lines of dialogue, in a couple of weeks. The new system also allowed us to change a character's expression mid-sentence which made them really spring to life.

Timing is very important for storytelling and animation. Often when you start to put audio into a game it is necessary to revisit the work done earlier because the timing needs to be redone. One solution is to wait until you have the audio but that is often not feasible since you need to test the game before you have the audio available. With some smart programming we were able to bypass this issue. When dialogue was imported into the game, the cutscenes would automatically scale to fit the length of the now spoken dialogue. Some manual tweaking was still necessary, but it saved us a lot of time and was made possible by having our own timeline system we could modify as needed.

Around the world in 80 conferences

Okay it might not have been that many conferences, but we pretty much covered the whole globe in our effort to spread the word. The first event we went to was the Play Blackpool convention in Norbreck Castle where we showcased the game for the first time. It was a small event which was a great way for us to get some experience before attending bigger events. Over the rest of the year we showed the game at Pocket Gamer Connect in Helsinki, Tokyo Game Show, EGX in London and also at Bristol Comic Con.

In general, we got three things out of attending events. One was to get a lot of playtesting done with a broad range of players and get

The game

Developer: Rumpus
Publisher: Self-Published
Release Date: December 11th, 2014 on iOS.
Platforms: iOS, PC & Mac
Number of developers: 3-6 at different stages of production
Length of development: 14 months
Budget: 60,000 GBP
Lines of dialogue: around 1,200
Unique animations: 603
Lines of Code: 8,410
Development tools: Unity3d, MonoDevelop, Git, Flash, Photoshop, Adobe Premiere, UniGayo, our own open source version of Papagayo
Amount of Tea and Biscuits consumed: Enough to feed a lesser spotted glump for a year

Adventure noses, a must have for any self-respecting explorer.

Placeholder art or a concept drawing for »Bertram Vice«?

their immediate reaction. The other was to help build our network. We also think that showing Bertram to representatives from Apple at these events resulted in the game being featured on the front page of the App Store when we released the game. Lastly it also allowed us to learn how to talk about the game to get people and press excited about it. Important knowledge when it came to do the written promotions on our own website, for the app store and when creating the trailer. It also helped us to get in contact with journalists and reviewers who can be hard to reach through email since they are bombarded with press releases on a daily basis.

Good writing and voice acting

In order for an adventure game to click (pun definitely intended) the writing needs to catch the player's interest. Where other genres offer mechanics that let the player have more control over the experience, adventure games rely much more on story. Luckily Seb had a knack for cheeky British humor and bad puns which an adventure game is the perfect vehicle for. Of course you still need good voice actors to make the dialogue come to life. We were lucky that Seb and Joe Wood, the other owner of Rumpus, had a bunch of friends who had provided the voice for Bertram and the other characters for years. Something I also think made the way we did voice acting different from how it is done for most games is how much Seb allowed the voice actors to ad lib a lot of their own lines during recordings. This gave us some of the best jokes in the game and a more natural delivery. In general the game received a lot of praise for the voice acting and we recently picked up a nomination in the Best Narrative nomination at the Casual Connect Indie Prize.

Right amount of playtesting

Inventory based puzzles are tricky to design since each puzzle and the items you use to solve them with can be understood in so many ways. This is especially true for comedic adventure games that rely on the puzzles to provide a sense of the humor. As such a shoe can for example be something you use to plant the seed of the man eating plant you've been carrying around or the peculiar odor it produces could be used to lure a love sick skunk away from its hiding place.

A good adventure game puzzle is often characterized by a bit of lateral thinking but it is therefore also important to playtest them a lot. When we started development we relied heavily on paper prototyping since it allowed us to test out the puzzles without a single line of programming. As good as paper prototyping is though, it has the weakness that it introduces a major variable into the equation: the one conducting the test. When you act as the computer it will inherently introduce variations in what you say or how you move the paper around for each person you test with.

As more of the tools became ready to use we started to skip paper prototyping and implement a rough version of the puzzles and story with placeholder art in engine instead. In general we were good at testing ideas out in this way before we spent too much time producing assets for them. At the end of production we also became better at using our design document to find faults in the puzzle design before we even got to test them which was a big plus. You should not be afraid to optimize how you work as the game progresses.

Things that went down the chimney

At the start of the project iOS was selected as the lead platform for the game. This was pretty much because the iPad 2 was the device with the lowest specs we wanted to support and barrier to entry is easy since everyone can submit to the App Store as long as they are a registered developer. However, midway through production we decided that it would be better to release on PC first. Firstly we expected our core audience there, PC still has the biggest community of adventure gamers and premium games are still the norm on the platform as well. We also wanted to avoid the stigma games that first come out on mobile carry with them for some of the PC audience.

Jumping back and forth on lead platform

We first tried through a contract to get onto Steam but were told we needed to go through Greenlight instead. Reading about other developers experience interacting with Steam representatives, we might have been lucky if we had talked to someone else from Valve. We decided to try our luck with a Greenlight campaign. We launched it at the same time we showcased the game at EGX and initially had a lot of traction but it slowed down drastically after the game was pushed out of the Greenlight front page. In hindsight we should have launched the campaign sooner. In the end because of a deadline imposed by our funders we were forced to jump back to iOS as the lead platform instead.

Looking back at it I think our decisions were made on a sound basis but what we should not have done was put time into the PC version before we were sure the game was going to release on it. At least a couple of weeks was spent working on the UI for PC which could have been used on something else. Thankfully the game was greenlit in February this year and the desktop version is now available on the AdventureGamers.com's online store, so it is not like the effort was wasted.

Awful audio workflow

The only core team member who did not work in our office was our composer and SFX guy Cameron Reynolds. Because we could not afford to give him his own Unity license it was up to Dan and me to implement all the audio he sent to us over Dropbox. Unity is great in many ways

but audio has always been one of the weaker sides of the engine. I had created a bunch of components to make it easier for us to trigger audio on animations, randomize the audio and crossfade between the music but it was still a lot of work to get all the audio in. During the whole process Cam had to act through us as intermediaries which greatly slowed the process down. If our audio workflow had been better Cam would have been able to implement and play around more. In the end we had a marathon audio session over a weekend where he came to our office and we mastered the game in one go with me acting as a human interface to Unity.

Another great limitation that arose during this session was how Unity 4 represents volume. Actual volume and perceived volume are not linear but exponential which is why the logarithmic dB scale is audio professionals' favorite way of representing volume. Unfortunately, Unity 4 does not use a dB scale which meant it took us a long time to find the correct levels. Thankfully that seems to have been solved in Unity 5 now with the new audio mixer.

A final pain in the audio department was the lengthy task of adding all the individual audio files for the dialogue into the game. When we got the raw audio from the recording studio Seb had to manually cut out the best takes in Adobe Premiere and export them with the right names. When you have more than 1,200 lines of dialogue that quickly becomes a chore and we often had audio files not being named correctly. We did create a system that would import the audio from dropbox automatically and tell us what lines were missing in the game, which helped a great deal but this was still a surprisingly large task to get right.

No localisation at launch

Separating subtitles from the rest of the game was one of the first things we implemented so it would be easy to have more than one language. It also had the added benefit of making it easy for Seb to change the dialogue without having to edit the game in engine. Unfortunately, the most use we had of the localization system was that we had a friend translate the first part of the game to Japanese when we showcased the game at Tokyo Game Show. If the game had launched with more than one language it could have helped to make a bigger splash. We are currently getting Russian, Spanish and German translations made, but should have done this before so they had been ready for the launch.

Beta testing is Alpha & Omega

While we had enough playtesting done, we should have put more effort into having the game beta tested. For beta testing we used TestFlight which worked out pretty well for us, although we were quite late to recruit testers which meant we did not have a big pool of devices to test on. Right up to submitting the game we were also busy putting in content which meant there was less time for bug fixing. As a result the game was released with at least one bug that could stop progression, something that pains me as the programmer on the project. The quality of responses we got from beta testers also varied a lot which meant it could be hard to track bugs down. In the future I would also hire a QA tester to tear the game apart. To use such a tester effectively I would make sure the game is essentially done a couple of months before release so all development time can be put towards making it robust. That ties neatly together with the next point.

Not enough of the right kind of PR

Evaluating PR is kind of tricky since you are trying to predict how something might have turned out that is heavily affected by external factors. However, in retrospect a couple of things we should have spent more time on was building a bigger online presence and in general get the word out to more people. Bertram has always had a strong presence on Twitter and Facebook but those are best at keeping your audience engaged while you make the game.

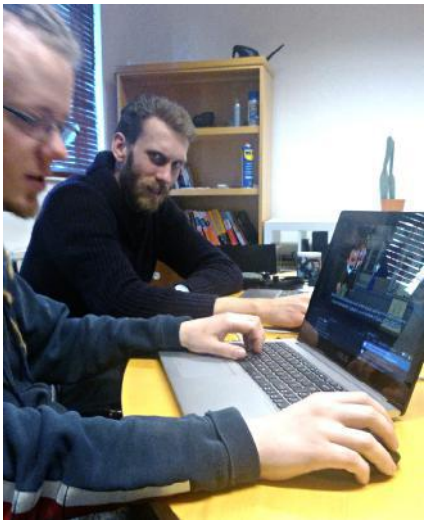
The last conference we showcased at was AdventureX in London. By far the smallest event we attended, it was also the one that catered most to our target audience since it is only for adventure games. A lot of the attendees really liked the game and were really surprised that they had not heard about it before. They were even more surprised to hear we were releasing the game on the App Store in less than a week of the conference! To me this indicates we had not done a good enough job of connecting with the most loyal fans in the adventure game community who could have helped us to spread the word of the game as we were developing it.

Something else we learned at the end of development is that when you release on the App Store it is only after the app has been approved that you can send out review codes. Because we were busy putting in content we only just completed the game before we had to submit it to hit our target release date. As a result we could only send out review codes about a week before the game would launch, not ideal if you want as many sites as possible to cover your game and which probably lead to the game not getting as much coverage as we had hoped.

End of the Journey

In the end we managed to produce and release the first episode in a little over a year. The game was featured on the front page of the App Store in America, UK and several different countries. We recently got greenlit on Steam after having been there for about 6 months and the game has also been picked up by a publisher who will handle distribution in Japan and China. If I should point to the biggest mistake we made it was to not focus enough on PR. It is an experience we will carry with us to the games we make in the future.

Alexander Birke



Me acting as human mouse for Cameron Reynolds, our composer.

The team behind Bertram: Paula Poveda, Alexander Birke, Leah Panigada, Seb Burnett, Dan Emmerson and Joe Wood.

THE ART OF COMMERCIAL GAME DEVELOPMENT

Jeroen Mol and Rick van Beem give their vision on how Lunagames is able to create games for the mass and reach a wide audience, while cutting down risk on app development. Their latest multiplatform title Luna League Soccer acts as an example.

Jeroen Mol
is Game Development Manager at Lunagames.

Jeroen has a Master degree in Art Management and combines that with a strong passion for everything digital. Jeroen has developed and released more than 25 games and 10 apps, mostly for mobile devices. Luna League Soccer was one of the games he helped to create. Currently he is working as Project Manager for Ex Machina and was former Game Development Manager for all outsourcing projects at Lunagames.

Rick van Beem
is Marketing Manager at Lunagames.

Rick is an industry veteran with over eight years of videogame marketing experience and has worked for companies such as Playlogic Entertainment, Gamania Digital Entertainment, OneBigGame and GameVillage. Currently he is Global Marketing Manager at Lunagames International and writes for Dutch tech website Dutch Cowboys.
@IanFleming77

Lunagames has a unique approach when creating apps. Instead of painstakingly polishing a product before release, it launches games and apps relatively early to see initial pick-up, community response and therefore receives valuable feedback on the viability of the product. Lunagames only spends around 1-4 months developing a game, depending on base complexity. After the first build is out on the different app stores, winning apps are quickly updated and enhanced with more features. Less performing apps take a back seat on the priority list or are left to die.

Luna League Soccer was a game developed in just four months using a talented team of one developer and two artists. This game, like any other Lunagames title, has been developed around three basic principles. The core gameplay needs to be based on a proven game concept, all features need to be reusable and the game needs run on multiple platforms and devices.

Core gameplay based on proven concept

Lunagames normally does not create revolutionary new game concepts. The time involved to develop and test new game mechanics does not fit within our narrow development window. We prefers to build upon existing genres and games, which will in turn minimize risk and will ensure that the general public is comfortable in playing these titles. More effort is put in adding a theme and marketing to the mix that should give a title instant appeal.

For Luna League Soccer, we used core soccer rules and gameplay mechanics and combined these with fantasy power-ups and special abilities. This idea is not new. Games like »Mario Strikers« for the NGC or »World Cup Soccer« for the NES already played around with this idea. Initially, all development was focused on getting a minimal playable version that would be fun and which could be expanded upon with new features and modes.

Reusable features

We at Lunagames believe you don't need to invent the wheel twice, and that is something we keep in mind when designing games. Certain elements such as assets can be reused and gameplay can be wrapped around other themes. This will save valuable time on future releases and will allow us to improve efficiency and output. We also often shop in the Unity Store to see if systems or assets we are looking for are already available before we try and create stuff from scratch.

For Luna League Soccer, we were able to reuse the AI, achievement and team upgrade system, which we had created for one of our other games. These were subsequently tweaked and extended with new features. We just needed to find the right look and feel of the game, for


Luna League Soccer (left image) combines core soccer rules and gameplay mechanics with fantasy power-ups, an idea that already existed in games like »Super Mario Strikers« for the Nintendo GameCube (right image).

which we invested quite some time and effort in externally developed assets. Particle effects, UI elements and an iApp purchase system were bought in the Unity store.

Multi-Platform Support

We like to work on multiple platforms for reaching full potential and maximum reach. All of our titles need to at least work on all current mobile hardware and preferably backwards compatibility with older hardware models. We aim to support the low-end (512 MB) device range as well as working well on the latest flagship phones. Using an engine like unity enables the developer to easily port games to target platforms, but these versions still require some tweaking to support platform specific features.

During Luna League Soccer's development cycle we often tested milestone builds to ensure we were still on track with our added features and multiplatform support. The toughest part was getting the ten different player models (1200 polygons each) working well on low-end. By repurposing the models as sprites we were able to pull it off.

Development Setup

Lunagames has a studio with twelve employees. Apart from the main development studio, the company works with a group of outsourcing partners that are based Eastern Europe, Asia and Africa.

Game development in the Netherlands is relatively expensive and by outsourcing (parts of) the development, you are able to reduce app development cost by 30-90 per cent, depending on the outsourcing location and stability level. From our point of view you have 3 stability levels with outsourcing partners:

- High cost/low risk: Large outsourcing partners (+30 employees). They have enough resources to scale up or replace members when needed. You have one contact person, which could manage multiple teams.
- Medium cost/medium risk: Development teams/studios (max. five team members). It is hard to scale up or replace members in the team. You will have one main contact point that will manage that local team.
- Low cost/high risk: Local freelancers. You need to find and manage other people to scale up. If you have proper control over the source-code and it's well documented, you could move the project from one freelancer to another. You need to manage the pool of freelancers working on the project.

Keep in mind that the Management cost will always increase when outsourcing. I would not recommend outsourcing on your first project.

For Luna League Soccer we decided to partially outsource the project to a group of international freelancers. The project was managed via the waterfall model, with a fixed weekly team meeting via Skype. All team members shared their progress, issues and planning regarding the project.

The team worked towards a fixed milestone (three weeks), after which they needed to upload all files, including source files, to the FTP of the Luna League Soccer project. We were able to produce the game in four months, three weeks more than was initially planned.

The toughest part during development was to get the ten different player models to work well on low-end devices.

Showing the game at events like Casual Connect helped to gain exposure for Luna League Soccer.

Marketing

After the first version was done, the marketing team took over.

As most of our revenue is based on in-app advertisement, as such, we cannot partake in the spiraling business of user acquisition. The costs are crazy right now and it is nearly impossible to break-even. We had to go for entirely different approach and tactics. Lunagames typical marketing approach can be broken down into four components. Some tips:

1. ASO (App Store Optimization)

Lunagames has always put a lot of effort in so-called app store optimization. Each store works differently and by optimizing your store's content one can increase visibility through search and/or visual attractiveness. Make sure you have compelling screenshots and preferably a video that shows off the functionality of your app. Put time and effort in keyword research so that users are able to find you in the store. Ensure you have a catchy icon that is instantly appealing, as this is your first point of contact with the end-consumer. Test different icon designs, to see which works best.

2. Community management

Leverage your existing community, even if it is a small one. There is a reason this group of people is following you on social media or rates your product in the store, they care! Don't shy away when seeing bad user-reviews. Instead engage, gather feedback and try to iterate based upon player feedback. Even if you are not updating or fixing, we have seen that just communicating with the community can make a huge difference on ratings and growth. Use your community to test early builds in advance, announce your latest products on Facebook and twitter to increase your reach.

3. PR

PR is perhaps the most difficult one on the list as there is so much competition out there and editors are buried in press releases and pitches. Secondly, good PR doesn't seem to bring a lot of downloads on itself, even if you get covered by high-traffic websites. Still it is very important as the App store editorial teams are closely watching publications like Touch Arcade and Pocketgamer to get hints and tips on what products to feature in the app store. So exposure is important! When pitching make sure it is short and to the point and instantly interesting. Back up your pitch with a short gameplay video or trailer to spark interest. Show your game at events like PAX, Casual Connect and gamescom. Make sure you do this ahead of launch!

4. External distribution platforms

At Lunagames we cover the Apple App Store, Google Play Store and Windows (Phone) Store ourselves, but there are a lot of options out there such as Steam, alternative android stores and of course console. Depending on what type of content you have it is always interesting to test the performance on these channels.

We have also released Luna League Soccer on NVIDIA Shield and plan to do a PC port in the near future. By offering wide range multi-platform support, one can easily tap into new markets and find fresh audiences. We often see cross-pollination between platforms and simultaneous growth.

Jeroen Mol, Rick van Beem

The company

Lunagames is a Dutch game development studio and publisher, which was founded in 2005 by Richard Hazenberg. The studio has twelve employees and the company has been focusing on casual smartphone games. Within three years, the company has built a portfolio consisting of more than 80 games and apps, most of which have been created with the Unity 3D engine

Lunagames

MAKING GAMES
THE NEXT ISSUE

04|2015 MAKINGGAMES WILL BE AVAILABLE ON JUNE 19TH 2015

The Age of Steam

Anybody who nowadays wants to publish a game on PC more or less can't evade Steam, as the service offers the perfect tools for everybody be it an Indie-studio or AAA-publisher. Our experts will elaborate how Steam Greenlight and Early Access can be successfully used, what you should be aware of when planning a Steam Sale and whether they are a curse or a blessing for the industry and how marketing on Steam works in general. In addition in detail breakdowns will show how sale numbers on Steam change over the lifecycle of a game.

Other topics

Top Eleven

Nordeus explains how they keep players invested in their five year old football manager by releasing big content updates.

The Trip to Panama

Mixtvision and Mimimi Productions created a game based on the famous children's story by Janosch and talk about how to entertain kids and their parents at the same time.

Middle-earth: Shadow of Mordor

Monolith looks back at the development of their open-world game on what went right and what went wrong.

IMPRINT

Editor-in-chief
Heiko Klinge
(Address see Publisher)

Managing Editor
Sebastian Weber

Editors
Yassin Chakhchoukh
Patricia Geiger
Markus Schwerdtel

Art Director
Sigrun Rüb (ltd.)

Layout
Manfred Aumaier
Alexander Wagner
Eva Zechmeister
Anita Blockinger
(Freelancer)

Editing (Freelancer)
Marion Schneider

Translation
Bettina Wilding

Contributors
Michael Benda, Alexander Birke, Ian J. Bowden, Xavier Damon, Christian Feldbacher, Michael Geidel, Jan Klose, Thorsten Lange, Corrado Longoni, Jeroen Mol, Anton Pankov, Mark Robinson, Olivier Rozay, Philipp Schaefer, Rick van Beem, Andy Velasquez, Annekathrin Wetzel, Matthias Zorn

We thank our interview partners:
Mathieu Ferland, Stefan Layer

Office Management:
Isa Stamp, Anita Thiel, Annie Weissenberger

CONTACT US

Advertising Sales:
Dustin Reichl,
dreichl@idg.de
Address Changes and Subscriptions:
kundenservice@makinggames.de
Article Suggestions and Comments:
editor@makinggames.de
Recruiting- & Projektanfragen:
projects@makinggames.de
Address changes and Subscriptions:
kundenservice@makinggames.de
Company register changes:
info@makinggames.de
Subject: Company register
Ad approval for print issue:
Manfred Aumaier, +49 89 / 360 86-602
anzeigendispoprint@gamestar.de
Ad approval Online:
banner@idg.de
Media rates:
www.idg-eas.de

CUSTOMER SUPPORT

Subscription and single issue orders:
Making Games Kundenservice
ZENIT Pressevertrieb GmbH
Postfach 81 05 80, 70522 Stuttgart, Germany
Tel.: +49 711 / 7252-275
Fax: +49 711 / 7252-377
Austria: Tel.: 01 / 219 55 60
Switzerland: Tel.: 071 / 314 06-15
E-Mail: kundenservice@makinggames.de
Web: www.makinggames.biz/shop
Prices (print):
Single issue: 6,90 Euro
Subscription: 35,40 Euro
(includes 6 issues)
Please contact us for foreign prices and bulk discounts
Payment option:
Postbank Stuttgart, BLZ 600 100 70
Konto-Nr.: 311 704
How to reach the team of Making Games:
IDG Entertainment Media GmbH
Making Games
Lyonel-Feininger-Straße 26
80807 München, Germany
Tel.: +49 89 / 360 86-660
Fax: +49 89 / 360 86-652

DISTRIBUTION TRADE CIRCULATION

MZV GmbH & Co. KG
Ohmstraße 1, 85716 Unterschleißheim, Germany
Telefon: +49 89 / 319 06-0, Fax: -113
E-Mail: info@mzv.de
Web: www.mzv.de

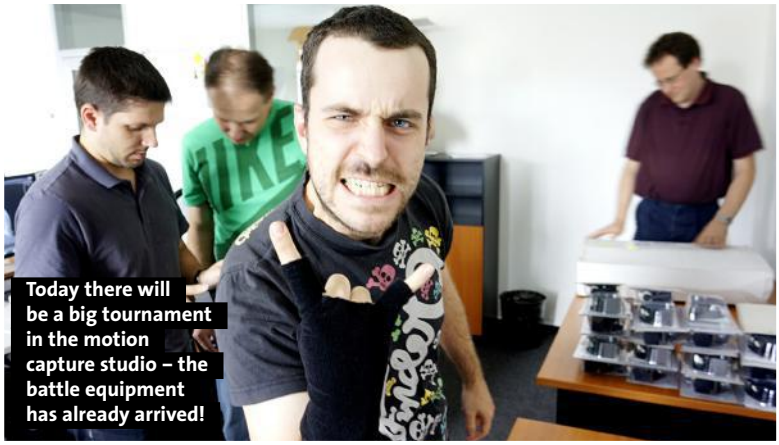
PUBLISHER

IDG Entertainment Media GmbH
Lyonel-Feininger-Straße 26
80807 Munich, Germany
Phone: +49 89 / 360 86-0, Fax: +49 89 / 360 86-501
www.idgmedia.de
Founder: Patrick J. McGovern (1937 - 2014)
Authorized Representative: York von Heimburg
Publishing Director: Frank Maier
Director Sales: Ralf Sattelberger (-730)
Head of CRM/Marketing: Matthias Weber (-154)
Board:
York von Heimburg, Keith Arnot, David Hill
Production Management: Jutta Eckebrecht
Print:
SDV Direct World GmbH
01159 Dresden, Germany

A day at ... WARHORSE STUDIOS

Welcome to Warhorse Studios – and welcome to 1403 Bohemia!!!

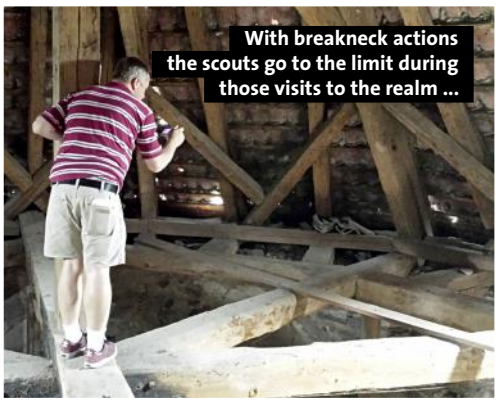

The designers at Warhorse take care of the daily topics and curiosities and at the same time create the game world of »Kingdom Come: Deliverance«.


Today there will be a big tournament in the motion capture studio – the battle equipment has already arrived!


The arena is ready, and the first warriors are warming up in our MoCap-studio.

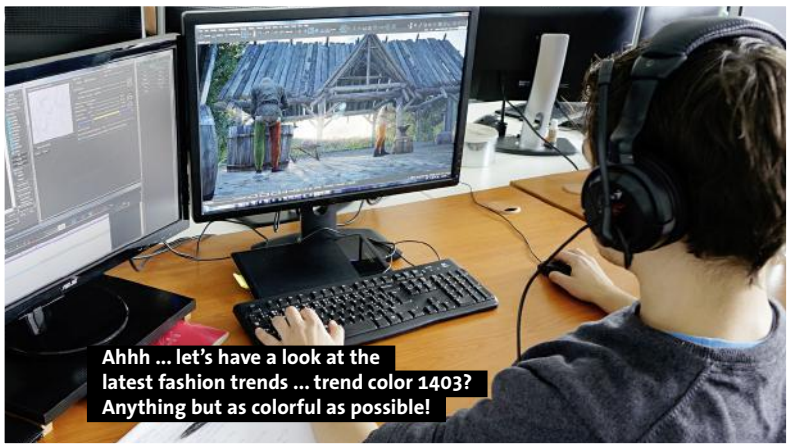

Taaaadaaaaa ... and the winner is ... MARTIN! Here is the reward for your victory: a crown.


After the tournament follows a quick interview with the King. Mr. Vávra. What are you most proud of in your Kingdom?


With breakneck actions the scouts go to the limit during those visits to the realm ...


... so that the cartographers can recreate the beautiful world of »Kingdom Come: Deliverance« in all its details.

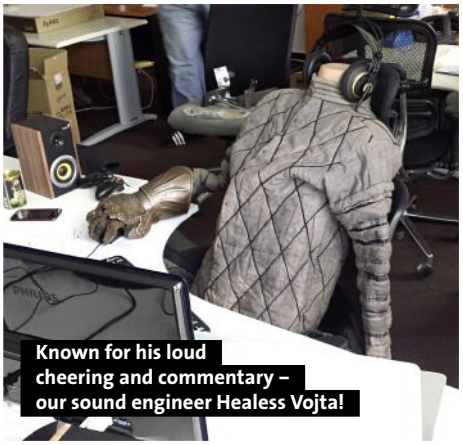
The medieval-themed roleplaying game Kingdom Come: Deliverance started on Kickstarter and by now got more than two million Dollars in funding from backers. The team from Warhorse Studios and its founder Daniel Vávra show how they spend all that money on the following pages.



Ahhh ... let's have a look at the latest fashion trends ... trend color 1403? Anything but as colorful as possible!

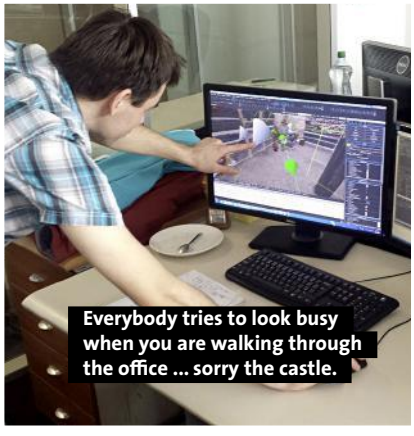
And here we have some ideas for evening activities – how about a jar of mead in the »Beautiful Maiden Tavern«? Interested?

Latest orders hang on our papers wall. Besides the contestant stats you can see the entire plan for »Kingdom Come: Deliverance«!



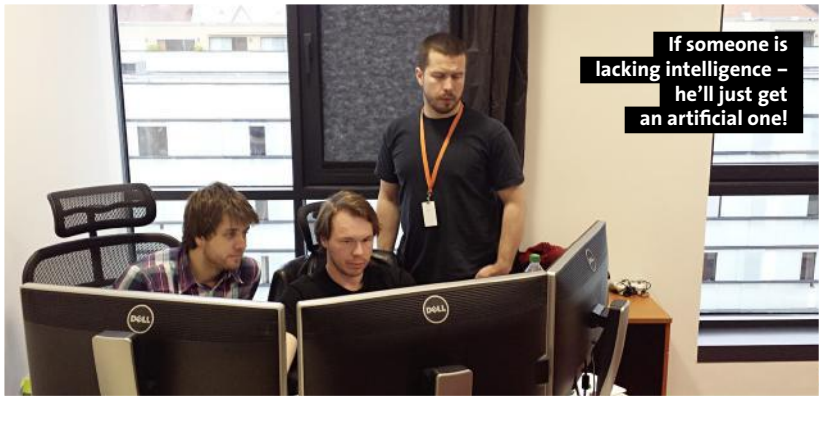
Known for his loud cheering and commentary – our sound engineer Healess Vojta!

Catering is also taken care of – David is mixing some of the good stuff at our alchemy table!

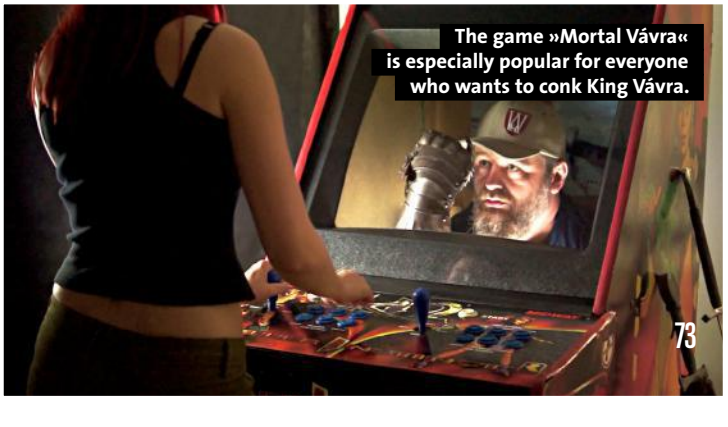


Everybody tries to look busy when you are walking through the office ... sorry the castle.

Visits to the Realm are most important – that's how the world of »Kingdom Come: Deliverance« creates an authentic feel.



If someone is lacking intelligence – he'll just get an artificial one!



The game »Mortal Vávra« is especially popular for everyone who wants to conk King Vávra.

CLEMENS
Programmer
Intern

KONSTANTIN
Principal Programmer
started as Intern

YAGER

More than 130 people from
20 different countries
working together in Berlin,
right on the banks
of the Spree river.

Creating awesome games
since 1999.

Join Our Team!

WWW.YAGER.DE
/CAREER


GAMEFORGE
START YOUR CAREER,
BE PART OF
THE TEAM!
CORPORATE.GAMEFORGE.COM/JOBS