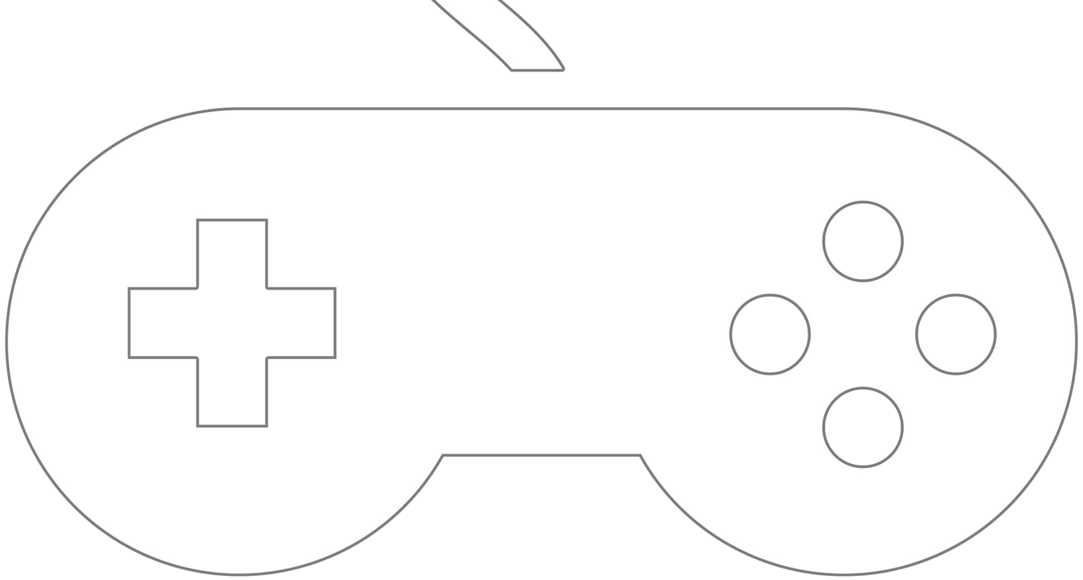

GAMIFICATION

But how and why?

AHOOT MEDIA

Dania GAMES

INSTITUT FOR DANSK SPILUDVIKLING

DEN EUROPÆISKE UNION
Den Europæiske Fond for Regionaludvikling
Vi investerer i din fremtid

Booklet available on idsu.dk

Discovery

Onboarding

Usage

Re-usage

Onboarding

Flow

Adapted from the work of the notoriously hard to pronounce psychologist Csíkszentmihályi, Flow is the principle that we perform best at any activity when it is neither too hard nor too easy. This is difficult to achieve as a designer, but you can approach it by making sure that the user always have very clear goals, designing a progression along goals of increasing complexity, having clear and immediate feedback and take away distractions.

Ask yourself how many concrete goals your user needs at any given time. When should a certain feature be presented? How are you telling the user that they are doing something right? Do you have to have all features visible at the same time? Can some steps be rearranged to create a better user-journey?

INTRINSIC

ONBOARDING

Tutorial

Instead of telling the player what to do, have them do it and reward them for good results. While a tutorial for a coder or in the beginning of using a piece of software could usually simply be a step-by-step execution of a task, tutorials in games are more often than not fun little levels, using only part of the full mechanics and giving the player who plays through them an advantage in the rest of the game in the form a little extra health or a couple of items. What is essential is giving the player the opportunity to choose how to solve problems and let their progress carry on afterwards.

Where can you give a new user some room for making choices? How can you structure the tutorial so it will be useful for the user afterwards, and not just as a separate exercise?

INTRINSIC

ONBOARDING

Progress bars

There are few things as de-motivating as having no idea whether you are hours, days or weeks away from completing a task. Games recognise this and spend a lot of time trying to quantify progress. This is done through characters becoming more powerful and gaining levels, where the player starts out at level 1 and then gradually as they play the game progress towards level 2 and 3 and eventually 30. Many games illustrate the possibility of this progress through a skill or advancement tree. You might train you hero in dodging because you can see that this leads you to be able to duck and roll on level 5.

Use progress bars to clearly indicate how far the user has come along a task or a learning path, and indicate along the progress bar what they will be able to do or have achieved at certain stages.

EXTRINSIC

ONBOARDING

Larger narrative

Day by day, data entry by data entry, every project can look boring, drab and overwhelming at the same time. The same can be true of some of the enormous video games out there requiring 60, 70 or even 100 hours of gameplay to complete. What keeps the players invested through such a long experience? A larger narrative that provides an end goal. The animated film Tangled begins with the words "This is the story of how I died", which really does tell you that there's drama coming, even when you're watching the heroine assault other characters with a frying pan willy-nilly.

What story can you begin with when you first meet your users? How will they end up feeling when they understand and can use the software? What can the software accomplish in their lives, in their own actual stories?

EXTRINSIC

ONBOARDING

Empowered progress

You know the expression "pushed out the door"? When someone is about to go on a journey, and they need that extra little nudge in the right direction? This is the mechanic of endowed progress. When you set a goal for a user, don't have them start from scratch. Give them a little help along the way. If they have to fill in forty data points to progress in the tutorial, fill in the first couple of them for them first. This lets the player get a taste of success, motivating them to go the rest of the way themselves.

Where can you help your users along? Which parts of your software seem the most out of reach to a new user?

EXTRINSIC

ONBOARDING

Usage

Game feel

Games are not limited only to rules and goals and actions. They are also about how they feel. Game designers work with this aspect by doing extensive user testing. But instead of focusing only on whether users can accomplish tasks in the game, game designers look at the intangible, tactile sensation experienced when interacting with game.

How do you users interact with your software? Do they want to use your software again - just because it felt nice to doodle around in it? Observe them and ask them and polish that game feel. How can you add visuals and or sounds to amplify the users experience?

INTRINSIC

USAGE

Pliability

When an interaction is tightly coupled and highly responsive - it almost feels like you can form the object with your hands. Games with high pliability often have few to none overlaying interface elements, but the interface exists within the game world itself. In “Dead Space” Isaac’s health and stasis are represented on the characters back, build into his suit - a very intuitive place to look for that type of data, in this type of sci-fi game experience.   Pliability evokes a tactile experience, and a sensation that something is natural.

How can you create a direct link between the system and the way you control it? Can you remove some overlaying interface elements and integrate them directly into the software? Can you add more physicality to the interface, so it feels like you manipulate it directly (etc. like a hologram)?

INTRINSIC

ONBOARDING

Customization

A way to ensure player investment - and thus a feeling of relatedness to the game - is, believe it or not, to let the player choose a hat. Not for themselves, but for their character. Making choices about how a character looks, dresses or lives makes the player feel their character is distinct from the characters of all other players. The smart way to do this is to set up a set of variables - hair length, hair colour, facial hair, tattoos - which allows for millions of combinations, making any combination feel unique to the player, no matter whether someone else has the exact same customization.

In software this can be achieved for example through the use of different colour schemes, the ability to re-map shortcuts or choosing between different ways to visualise data. Where in your design is there something the user could be allowed to change?

INTRINSIC

USAGE

Achievement badges

When you get to a milestone, it's often a good idea to celebrate it. This gives a sense of accomplishment and you can look back at the celebration as a reward for all your hard work. Perhaps you even produce a little memento, like a commemorative t-shirt after a project well done. In games, this is usually called Achievements, since that was the name for them on the Xbox 360 where the concept was introduced. If you do something either particularly impressive or at least correct in a game, you are awarded with a badge that has a name related to the task. "Button masher" if you press alle the buttons down at once, or "Pigeon Hunter" if you find all 35 pigeons hidden around the city. Smart game designers use this to show off certain parts of the game the players might not get to. Looking for 35 pigeons all over the city makes you explore the city, probably discovering something you wouldn't have seen otherwise.

Which parts of your software are users never getting around to? How can you design achievements to guide them there?

EXTRINSIC

USAGE

In-game economy

Many games have the players collecting coins or flowers or berries or other resources. In the game these can then be exchanged for things that help the player, such as a better sword, extra lives or opening up a new area. This is useful if you want to have the user explore many different parts of your software to "gather resources" but if you do not have a particular place you want to direct them to. It can also be used to make repetitive task seem rewarding. Let's say you let the player earn points for each new data entry and then spend the points on new avatar pictures on the company message board. The user will then be able to set the goal of gathering a certain amount of points so they can purchase a certain award.

What repetitive or menial tasks of your users are necessary for your software to work? What might your users be interested in buying inside your software?

EXTRINSIC

USAGE

Quest lines

When players navigate an epic story in a sprawling world, they can get very overwhelmed unless they are presented with well-structured problems and little stories. One of the game designer tricks for this is presenting quest lines. This is usually structured with a Quest Giver that presents the player with a Task. Accomplishing this task will usually lead to a complication that the player then has to overcome. For example, the mayor of a town may hire the player to combat a group of grisly goblins, but arriving at the goblin camp, the player discovers they have also abducted a stable boy from the nearby manor. If the player can manage to save the stable boy, they can then head to the mansion for an additional reward before heading back to the mayor for the promised reward. This lets what might be seen as a boring set of bland tasks be connected as a story, which the player can relate to and tell others about.

What set of connected tasks seem bland in your software? How might you have a quest giver in your software? What could a reward be?

EXTRINSIC

USAGE

Scarcity

How do you know what you want in a game? One way is through scarcity. In Minecraft, trees are everywhere and wood is a very common resource. Thus, tools made from wood are seen as commonplace and not highly valued by players. In contrast, diamond is only found by tunneling deep beneath the earth and using especially well-made pickaxes to extract them. Thus, diamonds are more scarce and feel much more valuable to the player. This is difficult to replicate in software, because you rarely want to limit the amount of times a user can use a particular feature, for example. But maybe there is something in the interface design that can be made to stand out through making the element scarcer than others? Or you might be using the in-game economy mechanic and can use scarcity to direct user to a certain purchase?

Ask yourself what in your software is precious and whether you can signal this being precious to your users by making it less available.

EXTRINSIC

USAGE

Loss aversion

Rather than merely winning, players are often even more jealously guarding their already achieved accomplishments. Every player hates having to play an entire level again or losing progress in some other form.

Does the user ever risk making a mistake in your software? How can you make sure you tell the user they can avoid this? Alternatively: if you want the user to do something specific, how can you gently threaten them with losing some progress if they ignore your instructions?

EXTRINSIC

USAGE

Triggering

Players often have rather short attention spans, but are excellent at indicating, what they want, even if they can't get at it. For example, a player might need 10 diamonds to buy a certain in-game item, but has only 9. If the player frustratedly keeps pressing the diamonds icon, this could trigger the game presenting the player with an option to buy or earn more diamonds through a real-world action (paying actual currency or watching an ad).

Even if your software doesn't use in-app purchases, you could do well to try to identify what users do when they are frustrated. Where do they click when they can't find a certain feature? How do you trigger an explanation that can lead them to the right one?

EXTRINSIC

USAGE

Micro/macro/meta

Most games - indeed most tasks - are repetitions of the same things you do over and over again. However, to create variation, many games structure these in three types of loops. Micro, macro and meta. In the micro loop is the moment to moment gameplay, like attacking and killing monsters and picking up the treasure they leave behind. In the macro loop, meanwhile, the hero may periodically, after an amount of turns in the micro loop, return to town with the treasure and buy better gear, to more effectively do the micro loop. Finally, in the meta loop, the player will want to clear out all the monsters in the area, so they can move on to the next area and the next part of the game's story. What's great about this mechanic is that it lets th game designer bridge the gap between intrinsic and extrinsic, making enjoyable in themselves interactions play into rewards usable in other interactions.

How is the moment to moment interaction with your software? How does that lead to a larger progression? What is the overall purpose with the interaction and progression?

EXTRINSIC

USAGE

Re-usage

Collaboration

Sometimes players work together toward common goals rather than competing against each other. This is usually a challenge to encourage as a game designer, since you need to figure out how the players can help each other, how they should agree on goals and how to keep both players engaged at the same time. However, in work life most of our tasks are done in some form of collaboration with others, and this is a rich area for gamification designers to explore.

Ask yourself how your software allows users to help each other, what tasks in the software might be enhanced by more than one user working on it at the same time, and if you can integrate something a user can do while waiting for the other user to finish their part (perhaps a small game integrated into your software?

INTRINSIC

RE-USAGE

Traces

If you play a racing game, you can usually turn on what is called a ghost car. This is a ghostly representation of a car driving through the course, illustrating how fast someone else has completed the course. In the lonesome castles explored by players in the Souls-games, the player will never meet another player, bot can find notes left by them, warning of dangers ahead or setting up traps. This heightens relatedness, making the games seem more lived in, without having messy real-time interactions.

How can your users leave traces of themselves in your software for other users to find? What would be a cool trace to find from an earlier user?

INTRINSIC

RE-USAGE

Endowment effect

In for a penny, in for a pound, the expression goes, and this is true for games as well. Games are designed to have constant progression, making the player feel that they have invested a large amount of time already and this makes the game precious to them. This is true for many activities: the more we do them, the more we want to do them.

How do you keep users interacting with your software for as long as possible? What will make the users keep the software running even when they do not actually use it?

INTRINSIC

RE-USAGE

Competition

All games are a struggle, and most games are struggles against other players. Competition keeps us sharp and engaged and is an easy an clear way of presenting a goal to a user. Find out what can be measured and provide a reasonable metric of the users' progress, and then establish a way for the users to benchmark against each other. This can be as smple as a high score list or as complex as a multi tier league system. What's important is that the competition rewards doing things that are actually useful and that it is clear to the users where they and their rivals are placed in their contest.

EXTRINSIC

RE-USAGE