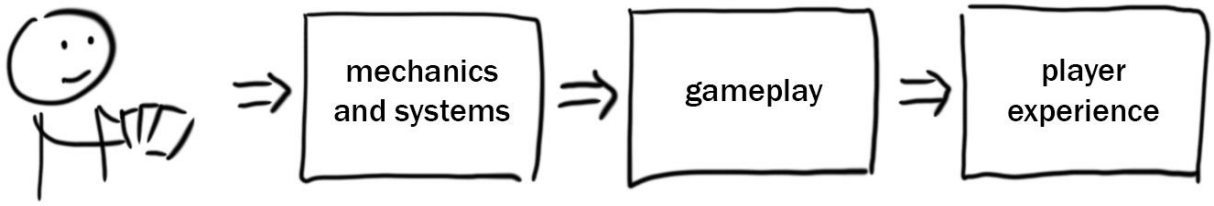
Gameplay

In the previous chapters we covered mechanics, the building blocks of gameplay, and assembling them into systems so that together they form chains, loops, and interlocked mechanisms. In this section we look at how these elements together produce gameplay.



Gameplay is the *dynamic activity* that emerges out of player's interaction with mechanics and systems, and participating in this interaction creates a certain *experience* for the player. Gameplay is what happens when we participate in an interaction loop with the game's mechanics and systems, and other players in games that support that.

How can we analyze gameplay, and what kinds of tools do we have at our disposal to evaluate and improve it? In this chapter we look at how we can systematically think about gameplay.

Motivating example: The Sims

Let's say we are playing a session of *The Sims.* This is Alice and Bob's house, and Bob came back from work to see that Alice is throwing a party. Bob is exhausted, and his energy is close to crashing. But at the same time, he feels unhappy because his social score is low, and meeting someone new would really cheer him up.

We need to decide what he should do. Meeting someone new would be good, but he is also very tired. Is it worth it, to risk passing out from tiredness and oversleeping for work the next day? On the other hand, meeting new people will make him very happy, and therefore more positive and efficient, a sure way to get that promotion that he has been working on. His life ambition is to become a CEO one day, and a promotion would get him much closer. So, we make a decision: Bob will have to buck up, go out there, and socialize. It's for his own good. Sleep can wait for another time.

Just that one decision point asked us to consider our choices and goals on several levels:

1. Short term: need to maintain our sim's immediate well-being and urgent needs.
2. Medium term: need to make sure today and tomorrow are taken care of.
3. Long term: we need to work towards our larger goals of becoming a CEO.

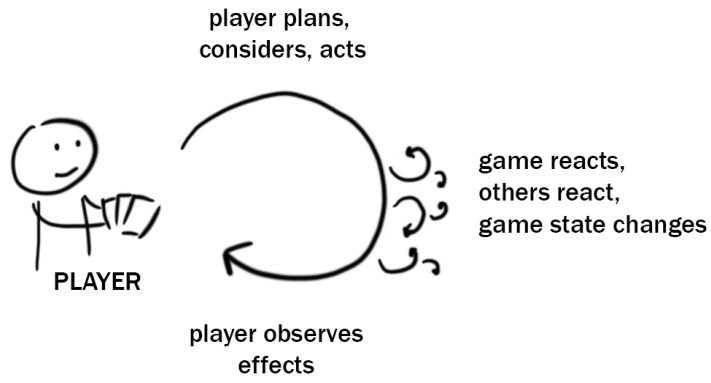
As we play, we must consider goals and activities on different time scales at the same time. This is not accidental: games are designed to deliberately present challenges on multiple levels simultaneously, and the various levels support each other: without short term challenges, there would be nothing interesting to do, while without long-term goals, there would be no reason for the player to keep playing over a longer period.

Working towards goals on different time scales typically means engaging with *different systems* in the game. In *The Sims:*

1. Short term planning is based on the "wants and needs" system, which gives sims different needs like "energy" with different levels of urgency, and a variety of ways to improve them, but usually at the cost of making other ones worse. For example, a hungry sim will need to find food very quickly, otherwise bad things will happen.
2. Medium term planning is based on the "jobs" system and the game economy, where sims can find jobs with different requirements, salaries, and work hours, so that they can make the money to afford the things they need to succeed in the long term. For example, buying a more expensive bed gives a sim more energy in the morning, so they don't need to rest as much and have more time to spend on productive activities (or partying and enjoying their simulated life).
3. Long term planning is based on the "life goals" system where a sim's long-term goal might be to become a CEO or a famous musician. Setting them up for success requires planning in advance and often spending money and time on things that have very delayed payoffs, such as taking evening classes or investing time in skill practice.

Gameplay loops

We have seen some activities in which the player engages repeatedly. We will call these *gameplay loops* since they are cyclical: the player keeps making decisions, acting on them, and then coming back to the same decision points.



In our *Sims* example, our smallest gameplay loop was *action selection:* pick the next action to take (go to sleep, go socialize, go make dinner), wait for it to happen, see the effects (more energy, happier sim), and repeat. This happens very frequently: every few seconds.

Then there are larger gameplay loops present as well at the same time. For example, every morning in the game, we need to make sure the sim is well rested, going to work, and making money. This requires different kinds of decisions, maybe every few minutes of real time. Finally, on a multi-day basis, we need to make sure they are improving their skills and advancing their career. These kinds of larger decisions do not come as frequently, maybe every ten or twenty minutes, but they also require attention.

Loop frequencies

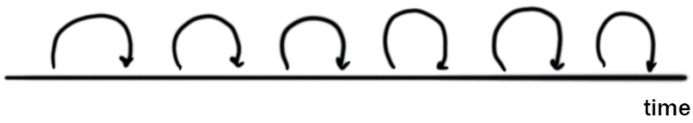
Gameplay loops have different *frequencies*: a fast loop turns over every few seconds or minutes of game time, while a very slow loop (like working towards a career goal) only requires infrequent decisions.

For other examples of gameplay loops, we can look briefly at a dungeon crawler game like Diablo:

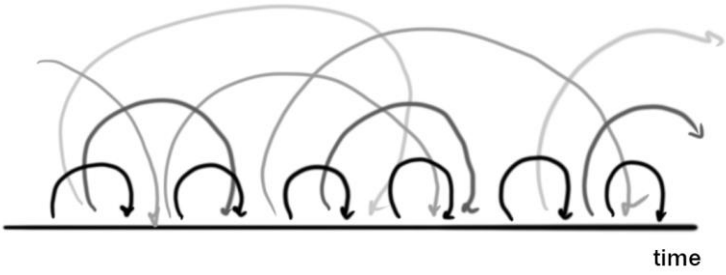
- Every few seconds (aka *micro* level): move, attack, defend, cast spell, equip, take, drop
- Every minute or two: find enemies, battle enemies, collect loot, locate treasure
- Every 5-10 minutes: venture out from camp, clear out a section, return and sell loot, heal
- Every 15-30 minutes: clear out the entire area, change up crew, level up characters
- Every few hours: explore an entire region of the game world, advance the story

Loops with different frequencies are active *at the same time.* The player's attention will be most focused on actions on the micro level, focusing on the immediate situation. But these actions also have macro consequences, so players must keep all levels in mind as they play, so they can slowly advance long-term loops while they interact with the shorter ones.

As designers, we must plan for all loops to be active, and design the game with multiple frequencies of decision-making in mind.



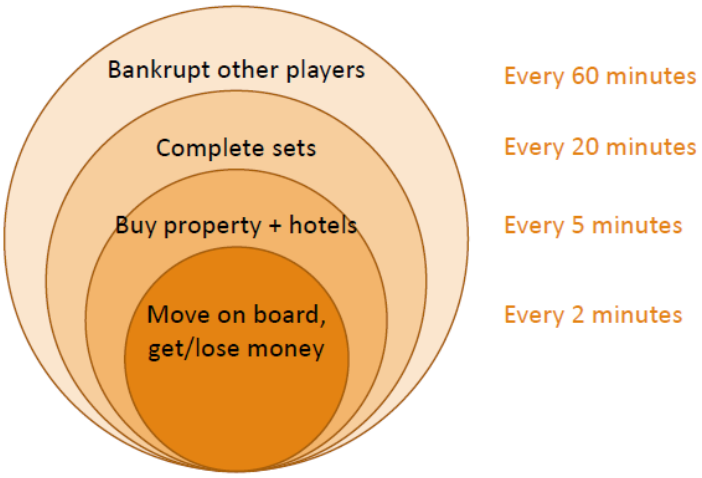
Fast loops only



Loop layering

Onion diagrams

A popular way to diagram gameplay loops visually is using *onion diagrams*. Here's an example onion diagram from a (simplified) description of Monopoly, which will also illustrate the reason for this name:



Monopoly

The fastest, smallest loop start out in the center, then ones with longer and longer periods get displayed around it. We can imagine that as the player iterates through the smallest loop over and over, they will also progress through the outer layers as well, but less quickly and only partially.

Here are a few more examples, also simplified:

Diablo

The Sims

Onion diagrams are mainly a communication tool - they make it very easy to document and communicate to others what kinds of gameplay loops exist in the design.

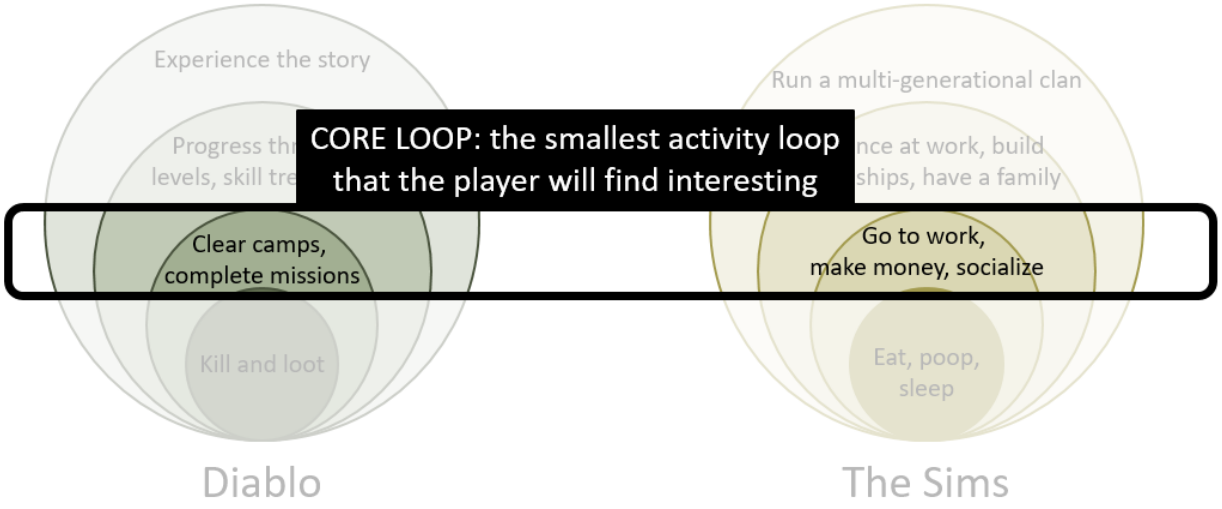
Since a game will also want to have a good distribution of loops at various frequencies, an onion diagram will also easily show if we have a gap someplace that we need to fill in with interesting activity.

The core loop

Designing interlocked activities is complicated, so designers typically start by focusing on the *core loop* first and evolve the design from there. This is the "minimum viable" level of activity, the smallest kind of a loop that's going to be *meaningful* and *enjoyable* to the player, and give them a reason to keep playing.

In some game types, the smallest loop or *micro loop* is also the core loop, because micro-level activities already provide challenge that keeps the player interested. This is more common in action games, for example in a driving game just mastering physical movement and keeping the car on the road, while passing the others to get into the lead, might be challenging enough.

In other games, however, the core loop will need to be slightly more complex than just micro actions. This is particularly the case in more systems-heavy games, where micro actions are not sufficiently challenging and not enough to motivate the player to keep playing. In a game like *The Sims*, just keeping track of physical needs might not be challenging, and the core loop is higher up, on the level of "go to work, make money, come back, buy some stuff for the house, repeat". In *Diablo*, it might be "go out and fight, collect loot, come back to camp, sell it, buy or upgrade gear, repeat".



The core loop requires a lot of attention, because the player will be interacting with it *all the time*, so we need to make sure they enjoy it, even as their attention shifts towards harder problems with a longer time horizon. This problem is amplified by players' varying skill levels, where advanced players may not be as drawn by the core loop as novice players.

Layering

The core loop is also a convenient starting point for building a larger game on top of it. A common design approach is as follows:

- Start with a micro loop that involves doing small things in the game. This typically won't be very interesting *yet,* but will be interactive, and lets the designer test out the basics.
- Explore and grow the micro loop, to find the core loop. This is the point where the game starts being enjoyable to play over the short term, on the level of minutes. Once identified, we spend some time iterating on the core loop to improve it, and to explore ideas about how to extend it.
- At this point we can start layering on longer term loops. If we have not designed those yet, exploring the core loop should give us ideas about larger loops that make the player enjoy interested over a longer time. These larger loops should *support* smaller ones, by interacting with them or their outputs.

In *The Sims*, for example, we described an engaging core loop of making and spending money, while managing the sim's physical needs. But this loop by itself has relatively short-term appeal. Adding larger loops that challenge it, such as adding family members or having a career to advance in, will change up the parameters of how to play the core loop, and keep it fresh and interesting, in addition to adding long-term challenges.

Advancing in one loop does not have to advance other loops, and it might be more interesting if the various loops are slightly at odds. Some game types seem to delight in making the layers work against each other, which gives the player a considerable challenge.

Factorio, for example, is a resource production and logistics game, where the player manages complex production chains, and spatial layout of the "factory" is of utmost importance. However, the player's objectives in early game will push them towards layouts that become a serious problem later in the game, and need to be worked around, or undone and rebuilt. Figuring out how to balance short term early game goals with future long-term objectives becomes a major source of fun, and often frustration.

Loops and systems

Gameplay loops come from engaging with mechanics and systems, as well as other players. Different frequencies are often supported by different game systems that work together, as already highlighted at the beginning of this section in *The Sims* example.

For another example, we come back to dungeon crawlers. We have already mentioned the loop of leaving camp, clearing out an area, and coming back with loot. Then each time we come back to base camp, we have a variety of goals we can try to advance: economic development (selling and trading loot), developing our characters (leveling them up or training), crafting (such as upgrading weapons), and so on. These in turn serve to advance different longer-term goals as well: wealth creation, or character and story advancement, or ensuring that we have much better weapons for later battles.

Just like we listed out activities for each loop, we can turn it around and match loops to systems and mechanics, for example:

- Every minute or two: find enemies, battle enemies, collect loot, locate treasure
  Systems: exploration, combat, items, stats, currencies, inventory
- Every 5-10 minutes: venture out from camp, clear out a section, return and sell loot, heal
  Systems: exploration, items, inventory, economy, crafting, upgrades, stats, skills
- Every 15-30 minutes: clear out the entire area, change up crew, level up characters
  Systems: inventory, crafting, character specialization, collectibles
- Every few hours: explore an entire region of the game world, advance the story
  Systems: character specialization, crew management, campaign / story arc

We can see that *various loops and frequencies often involve specific systems* (as opposed to having systems that work across *all* frequencies). This is because different systems have different behaviors, and are tuned in different ways, so it makes sense for them to hook up to specific loops. In fact, it would be devilishly difficult to devise a single system that would present the player with interesting challenges at all the various frequencies from micro to long-term.

Secondly, *loops often share their systems with their "neighbors"*. For example: inventory and stats participate in several loops, across different frequencies. This is desirable, because it means that a single action that advances one loop can also affect (advance or revert) progress in other loops - which makes those actions more strategically interesting.

Player motivation

We have looked at how mechanics and systems produce gameplay, by giving the player a variety of challenges and activities that arise out of interaction with them. The player interacts with them repeatedly, forming loops of repeated activity with different frequencies. By layering challenges with different time horizons together, we give the player a stream of gameplay which challenges them in the short as well as long term.

However, all this is premised on the assumption that the player will *want to* engage in the loops to begin with. If the loops are not interesting to the player, they will not want to play the game, and they will not enjoy our elegantly designed machine.

So, what is it that motivates players to play our game?

In the chapter on player experience (TODO REF CH2) we already talked about large-scale player motivations: the desire for particular types of experience, such as challenge, strategy, or community. We also talked about how personality types might affect what players enjoy, and how to take all of this knowledge into account in overall game design.

In this section, we turn our attention to small-scale motivation: what is it that motivates players to play the game on a minute-by-minute, or hour-by-hour basis?

We can address this by looking at some psychological principles for how different structures of activities affect behavior, which are shared broadly across people of various different personality types and individual motivations. This part will discuss those broad motivation theories, and their use in game design.

Intrinsic and extrinsic motivation

First, let's talk about two different types of motivation:

- *Intrinsic motivation:* when the player is inherently interested in the activity and its outcome. There is something about the activity that the player finds appealing. For example, dancing in *DDR (Dance Dance Revolution)*, or learning to fly around the world in *Microsoft Flight Simulator,* may be intrinsically enjoyable.

- *Extrinsic motivation:* when the player is driven to the outcome of an activity, but not necessarily to the activity itself. Grinding to get great items, or competing in an uninteresting competition just to get the top prize, may be examples of being motivated extrinsically.

Both intrinsic and extrinsic motivations are important, and both need to be addressed by game design. We describe them in greater detail in the following sections.

The line between them is fuzzy: something like playing a competitive multiplayer game usually mixes goal-driven motivation (getting a top position on the leaderboard) with some intrinsic pleasure of the game itself. Also, both intrinsic and extrinsic motivations are going to be specific to particular players - looking from the outside, we cannot tell how much someone is playing to get a reward, versus playing for the enjoyment of it, or some combination of both.

A game cannot shape a player's intrinsic motivation, but it should *support it* whenever possible. We cannot make the player enjoy dancing in a virtual competition in *DDR*, this has to come from within. But if they do, or they think they do, we can build a scaffolding that will make them grow in skill and enjoyment.

On the other hand, extrinsic motivations are easier to manage artificially. Getting more points, getting a high score or a cash reward, winning in a competition against others - these kinds of motivations are popular across many types of games. They speak to a more basic human desire to get better, to compare ourselves against others, and to see reward for hard work.

Ideally, games combine intrinsic and extrinsic motivations. A game that provides extrinsic rewards but is not itself interesting is going to turn into a chore. On the other hand, an activity that is enjoyable but does not provide feedback about performance or some extrinsic rewards, might feel too much like a hobby or a toy and not sufficiently "game-like".

Games do best if they can both fulfill player's intrinsic motivation for a specific kind of activity, with the extrinsic "carrot", rewarding them for getting better and better at what they already enjoy intrinsically.

Intrinsic motivation: flow and mastery

What makes players interested in a game on the small time scale, why do they enjoy engaging the game repeatedly, minute after minute and hour after hour? Everybody has their own reasons and motivations, but there are some commonalities that are broadly shared.

In this section we focus on the idea that gaining mastery of a skill or a domain is inherently interesting to players, and serves as a powerful motivation behind our desire to play. Our reaction to challenges is not the same across all players – it is based very much on where we are mentally when we encounter the challenge, whether we are able to meet the challenge, or if we are improving and learning from it. But even so, we as humans find learning and mastering challenges to be intrinsically interesting, if it matches our interests and skill levels.

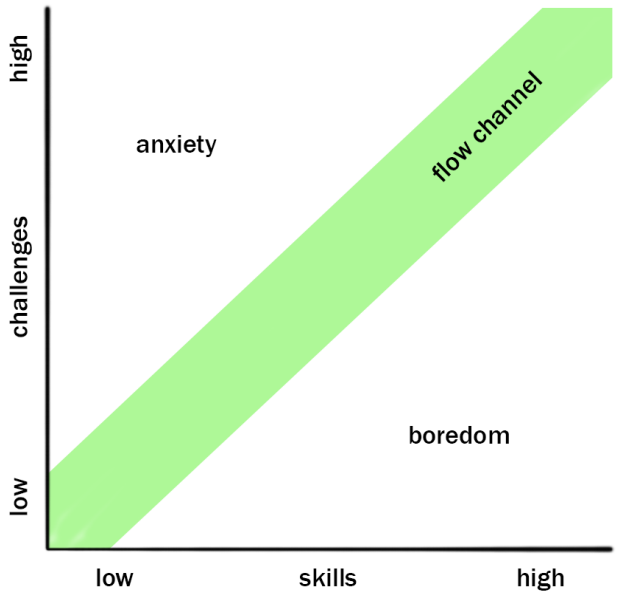
Surprisingly, this can be true regardless of whether the challenges are useful life skills, or "useless" past-times such as videogames. The connection between ability and mastery on one hand, and games on the other, has been noted for a long time (TODO REF Caillois 1958).

Flow theory

The concept of *flow* is central to understanding the joy of mastery. Commonly it is described as the feeling of "getting into the zone" when doing something. Anyone who has gotten lost in a game or an activity for hours at a time, and not realized how much time had passed, has experienced that intense, euphoric feeling.

People who get "into the zone" often report similar psychological effects: losing track of time being the most common one, as well as hyperfocus on the single task at hand, and losing track of oneself and one's presence in the world. When we are in the zone, it is just us interacting with the activity, without anything getting in the way. This happens not only in mental challenges, but also with physical challenges such as with athletes pushing themselves beyond their skill level.

Introduced in psychological studies by Csikszentmihalyi (TODO REF), the *theory of flow* posits that getting absorbed in activity like that is tied directly to the person's skill, and how well the challenge of the activity matches this skill. If the activity is too easy the person will be bored, and when it is too hard they will get frustrated or anxious. But if the subject has focused on the task and challenged well for their skill level, they will enter an optimal kind of state where they are fully engaged with the activity.


high
challenges
low
anxiety
flow channel
boredom
low
skills
high

Csikszentmihalyi et al postulate three core conditions of getting into the flow state:

1. A clear set of goals, which channel attention towards specific purposes
2. Balance between perceived challenges and user's perceived skills, in order to absorb one's attention into the task
3. Clear and immediate feedback, which tells the person how to adjust their actions

They also mention, in passing, one more optional element that supports getting into the flow state:

4. Congruence between task-specific goals and longer-term, more abstract goals

For games, this state of flow is highly desirable, and games often actively try to trigger it. As we have seen, games employ a variety of techniques to satisfy conditions 1, 3, and 4. Game content and the variety of systems provide a large variety of clear and actionable goals for the player to choose from, at different scales and frequencies which are set up to support each other. Also, feedback on how well the player did is plentiful, including using progression mechanics to guide the player along.

But one thing that games cannot easily do by themselves is balancing the challenge to match the player's skill. This is something that is still a difficult design challenge. Most often, we resolve it by having the player pick their "difficulty level", but that is often unsatisfactory (the player does not know or care to adjust the difficulty level correctly, and will not experience the right level of challenge).

In multiplayer games, the problem is slightly easier: we may try to match the player automatically against an opponent of similar skill, using scores or Elo ranking as proxy for skill measurement. This works well over a longer period of time, but is harder to tune correctly for new players. Many multiplayer games suffer from a "learning cliff" (as opposed to a learning curve) where brand new players are not correctly matched, and need to accept a period of frustrating losses as a part of the learning experience.

In the end, reaching the feeling of flow is central to successful gameplay. It requires the whole variety of game building blocks to work together: mechanics and systems that provide challenges for the player, systems and content that gives them goals, and finally the enjoyment of interacting with all of these elements and getting a good experience out of the challenge, exploration, and experimentation of interacting with a game.

Learning and challenge escalation

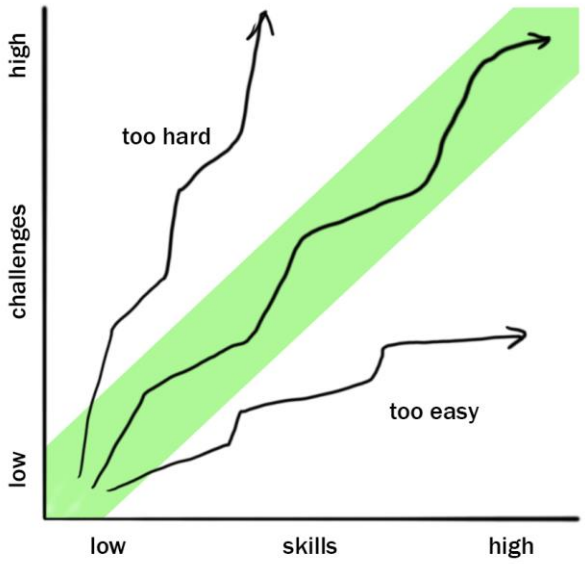
Koster's "theory of fun" (TODO REF) presents a related but slightly different argument. He posits that the enjoyment or "fun" of playing a game comes from learning, from acquiring mastery of activities in the game, and especially the skill to *predict outcomes* within the game. For example, this might mean learning how to navigate a particularly tricky level, or figuring out what is an optimal team make-up for a given type of a challenge. Additionally, this learning must happen in a safe environment, where stakes are low, but challenges are still interesting, so that the player has room to experiment and have fun while learning.

This model comes from a different perspective than flow theory, from the intuition of game design practitioners rather than from psychological studies, but it reaches similar conclusions: that enjoyment arises out of mastery, out of understanding how something works, and from giving the player space to figure it out.

If learning how to predict or "figure out" the game is part of the fun, what happens once the player figures it out? One implicit consequence is that if a game has static or limited challenges, the gameplay will inevitably become boring once the player masters it and is able to predict how the game will behave.

In the context of the previous discussion of loops and activities, this motivates *why* we need multiple gameplay loops at different frequencies, as well as content arcs that modulate those loops. Once the player learns small loops quickly, they will no longer hold any secrets. To keep the player engaged, we need to make small loops challenging (as with difficult action games), stack on larger loops that are more difficult (as with systems-heavy games), or repeat challenges but modify them along with the player's growing mastery (such as by introducing new and different content, or in multiplayer games, by matching them up against more advanced opponents).

The major difficulty here is that the player's skill keeps increasing, so we need to keep challenge escalation in sync with player's changing skill. If the challenge jumps too quickly, this may end up being frustrating, and while some players cherish this kind of a feeling, others hate it. Conversely, if the challenge increases slower than skill, the game may be too relaxed and ultimately boring, and again, some players enjoy relaxing activities that are below their skill level while others are infuriated by them. And sometimes the same player might have both reactions to different kinds of activities within the same game, depending on their mood or goals (whether they want to be challenged, or relaxed).



Dominant strategies and "solving" the game

As mentioned above, enjoyment of a game sits in the peculiar valley between full unpredictability and full predictability. If the game is completely unpredictable, like roulette or other games of chance, that reduces the enjoyment of gameplay (instead, players play for the exogenous metagame elements (TODO REF CH6) more than gameplay itself).

Conversely, as the game becomes predictable, it also loses appeal, for the opposite reason: if we already know how it will turn out, why even play?

To ward off predictability, designers pay special attention to *dominant strategies* in games, that is, strategies which clearly bring out better results than others. A simple example of a dominant strategy in *Tic Tac Toe* is to start the game by putting your cross in the center of the board, rather than somewhere else, which guarantees that any opponent moves can be countered, and each game can be won or tied, but not lost.

Dominant strategies can also be subtler. For example, when the game *Civilization V* first came out, different civilizations had different unique buffs, but the French had a particularly powerful buff that allowed them faster border expansion compared to others. Playing as the French, and concentrating on specific upgrades early on, one could easily outgrow other empires in mid-game, which then led to easy victory. (As expected, this unbalanced tuning got viciously nerfed in a subsequent expansion pack.)

Dominant strategies affect enjoyment of the game, but they are difficult to find analytically. Typically, finding them involves a lot of playtesting before release, as well as adding analytics to games after release, and monitoring for specific patterns of play that cause unexpected scores or win ratios (TODO REF CH4).

Loops and challenges

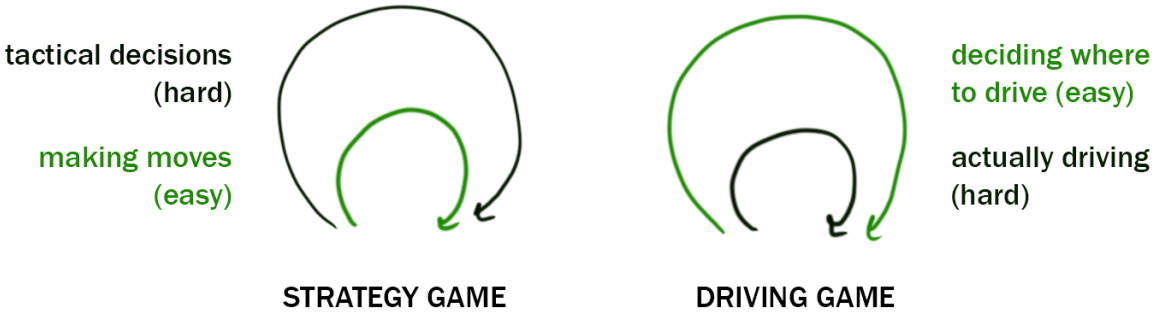
Do smaller loops always offer smaller challenges? As we have already hinted, this is not necessarily the case. We can consider a few examples with extra details:

1. Action games. Participating in the micro loop is already difficult: driving the car while passing and not crashing, shooting at alien space ships while avoiding their missiles – these take a lot of practice to get good at them. Arcade games, from *Frogger* or *Arkanoid* to the more recent *Flappy Bird*, made punishingly difficult micro loops their hallmark. In comparison, larger loops are easier (or non-existent) and participating in them is a more abstract, aspirational goal for more advanced players.

2. Strategy games or board games. The more abstract the game, the easier the micro loop becomes, as moving pieces on the game board can be tedious or require some calculations, but is not typically meant to be challenging. The larger challenge is in the medium and longer term strategy, of knowing what should be moved, where to allocate resources, and how to play the long game.

3. Games are likely to mix the two approaches. For example, squad-based tactical FPS games mix challenging micro loops that demand physical dexterity and fast reaction times, with longer term loops based on the tactical situation of the entire squad, as well as a solid amount of metagame (TODO REF).

We can illustrate this situation using onion diagrams as well, from (TODO KOSTER REF2):



Extrinsic motivation: work and rewards

When the player is driven by an outcome of an activity rather than the activity itself, such as rewards or acclaim, we can say that they are motivated extrinsically. These kinds of motivations are also common in games, and supported by the family of progression mechanics as well as other game design elements.

Here we introduce some elements of why players find extrinsic rewards motivating, and what games do to support them, and to tie extrinsic and intrinsic motivations together.

Progression and rewards

It is very common for games to include achievements and leaderboards, and many platforms (including consoles or the Steam desktop platform) support them natively, and make it easy for the player to see their virtual trophy wall or score board, and maybe even show it off to other players.

The family of *progression mechanics* includes these kinds of rewards. These mechanics encompass the different ways for giving the player feedback on their progress. As discussed previously (TODO REF CH3) these are elements such as:

- Score or XP (increases as the player accomplishes various goals)
- Levels (earned as the player reaches specific milestones)
- Achievements (earned for reaching specific uncommon goals)
- Leaderboards (shows how players' scores, levels, etc. stack up against other players)
- Campaign or story (unlocks content as the player progresses through the game)

Progression mechanics can provide a variety of extrinsic goals for players. They are very commonly used in games because they provide that extra bit of motivation for players. As the player's intrinsic motivation tends to wax and wane over time, the additional extrinsic "carrot" helps to tide them over. Additionally, the rewards themselves can be enjoyable on a meta-level, such as getting a high score can bring acclaim and jealously among friends.

Other mechanics can also serve a double duty, as rewards in addition to their regular role. For example: amassing cash, gold, resources, or units, can provide feedback about one's progress and feel like a reward, separately from how those resources can be used later in the game.

However, it is important to note that extrinsic rewards by themselves are not sufficient to make a game enjoyable. They provide motivation, but the game activities have to be enjoyable for the player as well.

As a historical side note: quite a few historical definitions of "what is a game", such as the classic Crawford essay from 1990 (TODO REF), have rested on the assumption that games *must* provide extrinsic rewards, such as scores or victories. While contemporary game design no longer considers this a necessary condition, it highlights the historical importance that extrinsic rewards have played in the development of the medium.

Reward schedules

Assuming that we grant extrinsic rewards, we should consider *when* to do so: what is the best way to grant them to the player, at what kind of pace, or based on how much effort. What will be the most appealing to players?

To answer questions like these, we can turn to psychological studies of rewards, and of how people assign value to objects or activities in general, not just in context of games. The best-known and highly studied family of approaches comes from the field of *reinforcement learning* in psychology, which is the study of how different animals and humans learn to act over time when their actions are predictably rewarded (reinforced) or punished.

For example, in some classic experiments, handlers would teach a pigeon to peck at a marked target spot and get food as a reward, and then they would vary how often the rewards get dropped, to see if that affects how the pigeon "works" for the reward. The term *reward schedule* describes a particular plan for how rewards get generated over time, based on some chosen principles (for a simple example: one unit of food every five pecks, or one unit of food every minute).

It is commonly known that animals can usually be trained to do simple things for food, but one of the discoveries was that specific kinds of reward schedules would result in very different training results, some better than others. Another interesting discovery was that, by spacing out the rewards, some animals could be trained to do increasing amounts of work in exchange for their rewards (up to a point where they lose interest).

Since its heyday half a century ago, reinforcement learning has been *thoroughly* criticized in psychology and education, as a massively reductionist and inadequate view of human learning or motivation, which is a very correct accusation – there is much more to human behavior than reinforcement learning.

However, the studies also discovered that in some cases this *does* model some simple human motivations, in some contexts, which makes them *useful enough.*

In game design, reinforcement and reward schedules are very commonly used, although not without controversy. They are criticized for encouraging players to focus on extrinsic rewards over their intrinsic motivations, or for being overtly manipulative. Some vocal critics also raise a moral panic, accusing them of exploiting human weaknesses for corporate profit (TODO REF Juul https://www.jesperjuul.net/ludologist/2010/08/31/the-video-games-of-video-games/).

However, virtually all video games use extrinsic rewards, and most commonly they use variable ratio schedules (described below), because they work very well at motivating the player. Regardless of the intentions of their designers, it behooves us to understand how they work.

Types of schedules

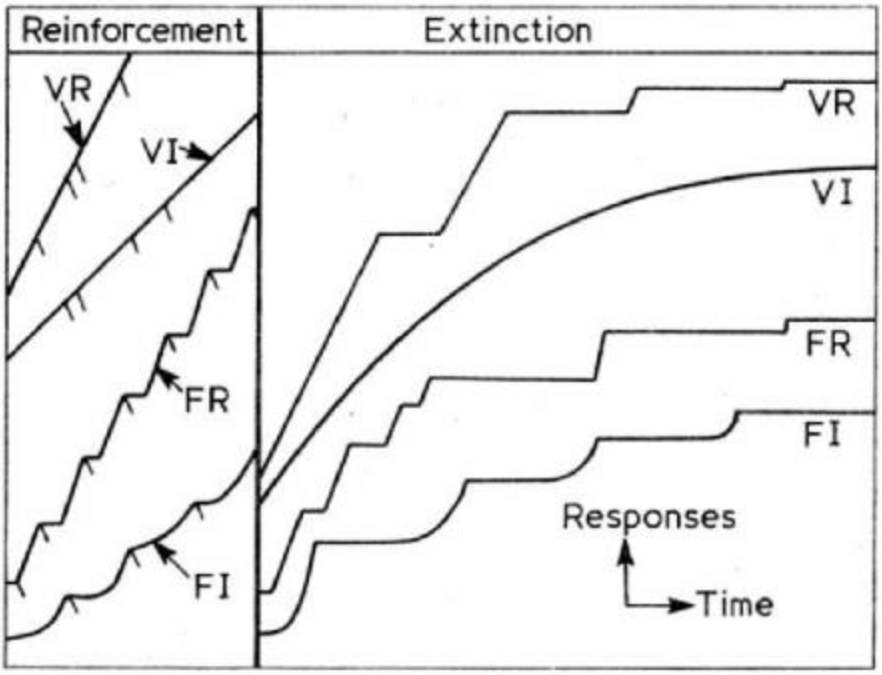
Many kinds of reward schedules have been studied, but the basic schedules we will discuss here are as follows:

- Continuous: reward the subject for each action directly
- Fixed ratio: reward the subject every N actions (eg. every 5 pecks)
- Fixed interval: reward the subject every N seconds while they're performing actions
- Variable ratio: reward the subject every randomized number of actions
- Variable interval: reward the subject at randomized points in time, while performing actions

These schedules can be compared along two dimensions:

- Reinforcement: how many actions are taken over time (how hard the subject is working)
- Extinction: how long do subjects keep working until they lose interest

The results are startlingly consistent across different experiments and subject types. Results of how trained animal subjects perform can be summarized in a chart as follows:



(TODO REF: Walker, S.F. (1976) Learning and Reinforcement, London, Methuen. Pp. 81. http://s-f-walker.org.uk/pubs/lr/lrframechaps.html)

To describe in more detail:

- *Variable ratio*: has the highest reinforcement and slowest extinction rates. In other words: if the rewards are proportional to amount of work, but somewhat randomized, the subjects work the hardest and longest to get them.
- *Variable interval:* tying rewards to randomized time intervals, rather than amount of work, results in slow extinction, but also less work per unit of time.
- *Fixed ratio*: with rewards spaced out predictably, work levels remain high, but extinction is faster. Also, it is more likely there will be pauses in activity right after a reward.
- *Fixed interval*: giving out rewards on a steady schedule while working shows faster extinction and lower reinforcement. Also, work slows down after a reward, but then speeds up as expected reward time approaches.
- *Continuous schedule* (not shown on chart) has the fastest extinction and lowest reinforcement of the set.

While these results are best known from animal studies, they have been repeated in various ways across a variety of test subjects, including humans. Something about the anticipation of an unpredictable future reward has clear effect on human and animal subjects alike.

Game examples

To see how reward schedules can be successfully used in games, let's go back to our ongoing example of dungeon crawlers – and specifically, games from the Diablo series.

The core loop is: go out and kill monsters, collect loot, come back to camp and sell loot. Loot is collected after combat: monsters randomly drop loot and gold when killed. Levels also contain scattered treasure chests with random loot. Additionally, there are different types of loot: tougher monsters drop better loot on average, and both monsters and chests also occasionally drop rare, collectible items.

If we reword combat as "work" (ie. successfully finding and killing monsters over time), we can see that loot is a reward for this work. But what kind of a reward schedule is it on?

1. Monsters take a randomized amount of work to kill, and on success they drop a randomized amount of reward, which is proportional to how tough the monster was. This is a variable ratio reward schedule, rewarding the player for their actions with some built-in randomness.
2. Sometimes monsters also drop rare collectibles, which have much higher value, more like rare payouts from slot machines. This is a secondary reward schedule layered on the first one. It is also a variable ratio schedule, like the first one, but with much higher payouts for much higher amounts of work.

But combat is not the only way to get treasure: you can also explore to find treasure chests:

1. Various locations hide treasure chests that drop different types of loot than monsters do. This requires finding those treasure chests, which in turn requires exploration and risk-taking (like fighting monsters along the way), and it might also require extra skills or tools to open them. Just like with monsters, the drops follow a variable ratio schedule, but in exchange for a different type of activity - also, the work-to-reward ratio will be different as well.
2. Similarly to monsters, some chests randomly contain very high value drops, so they also have a secondary variable ratio schedule layered on top, one that is slower but with much higher rewards.

So in summary, this dungeon crawler loop employs four different reward schedules:

1. Rewarding combat
    a. High frequency variable ratio
    b. Low frequency variable ratio
2. Rewarding exploration and risk-taking (different work and rewards from #1)
    a. High frequency variable ratio
    b. Low frequency variable ratio

Interestingly, combat loops and exploration loops have their own, multiple reward schedules, and they are tuned differently to keep things interesting and unpredictable. This is a great example of chaining multiple, concurrent types of schedules to reward the layer for both varying their activity, and also for advancing their skills in all of them.

Changing workload

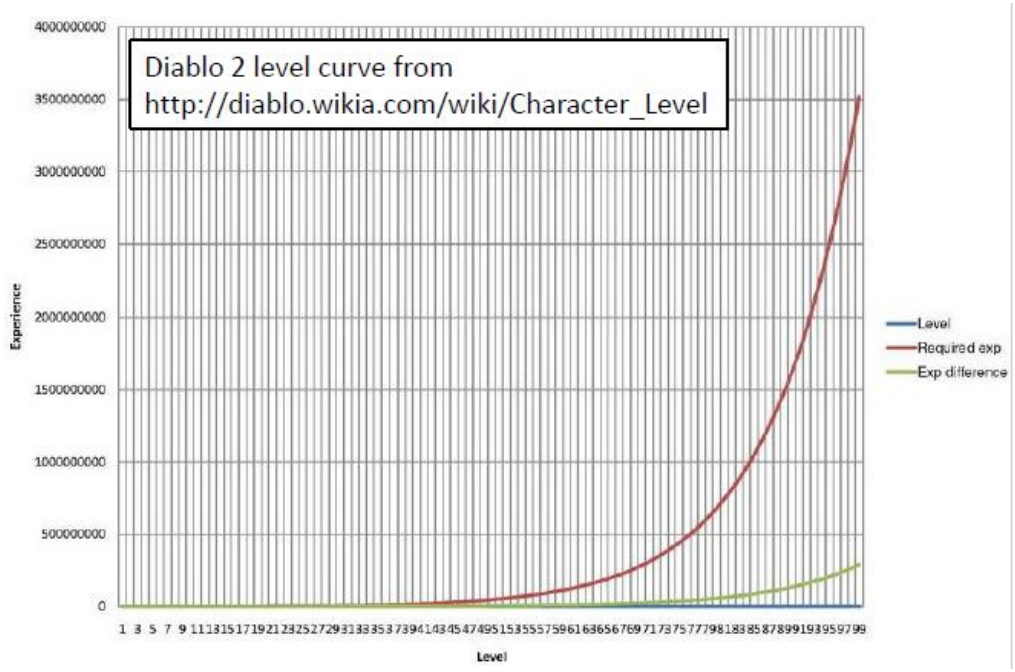
Another aspect of reinforced behavior is that, once learned, it can become basis for new kinds of behaviors: either more complex work, or a higher workload, for the same kind of a reward.

This effect is intuitively known by teachers and trainers: teach the student something simple, and reward them consistently, and once that is mastered, you can start increasing how much they have to work for the reward (for example, to build up endurance), or you can use that as a building block to make the work more complex (for example, by adding more complications).

This is commonly done in games as well. Perhaps most typical example are XP / level curves:

- Game actions (combat, exploration, etc) generate consistent amounts of XP
- XP gain is proportional to the difficulty of the action, which is in turn capped by the player's level
- Reaching specific XP milestones lets the player level up and get some rewards
- However, those XP milestones get spaced out further and further as the players progress

If we drew this as a curve from level to XP required, it might look like this:



In a level curve like this, every time you reach a new level, the XP required to reach the next level also increases. This has an interesting effect on the player: they have to either work harder to get the next reward, or they have to work smarter to get more XP with the same amount of work, for example by crafting better weapons.

In some cases, this might make them want to play the game more, while in others the increasing amount of work will demotivate them - depending entirely on the combination of how the rewards were tuned, and the player's own motivations.

Related topic: Gamification

Gamification is closely related to extrinsic rewards: it is the application of progression mechanics to domains outside of games. For example, online discussion boards like Stack Overflow reward participants for their contributions with points and badges, and eventually level-based ability unlocks (that is, the user needs a high enough score to be able to post new questions).

Since gamification provides extrinsic rewards, it can definitely serve as a "carrot" to drive desirable behavior, but with caveats. First, the reward needs to be meaningful and valuable to the participant. Leveling up to unlock an ability is more meaningful, than just leveling up and getting nothing more than a higher number in the player's profile. Secondly, the activity itself needs to be interesting as well. Just getting points without enjoying the activity behind it is neither interesting nor meaningful. A chore is still a chore even if you add points to it, and a leaderboard for who gets the most points is still a leaderboard for chores. Rewards do not drive motivation just by themselves.

Gameplay loop design heuristics

Let's say, we have an idea for a game, and we even worked out roughly what the player is going to be doing and what kinds of experience they will be having. Now we want to figure out how to design the gameplay loops that will drive it. How do we get started?

In the chapter on systems (TODO REF CH4) we talked about using user stories to guide system design. This technique of starting with *user stories* is broadly useful, and we can use it to guide gameplay loop design as well.

From user stories to gameplay loops

Consider our previous example involving a tower defense game. We can start by asking: what will the player be doing, over and over again? Can we find a core loop just from a narrative description?

We can take a stab at it as follows:

> Player builds out defenses, then waves of creeps spawn and attack the player's base. Killed creeps drop coins, which can be used to repair the defenses and build better ones, before the next, more powerful wave of creeps attacks.

This level of description is good: the player's actions are going to be building, defending, and then buying/upgrading/repairing. This kind of a loop sounds interesting already, and it could be a viable core loop (and playtesting will confirm whether it is engaging enough).

From this description we can also imagine smaller loops, for example:

> During defending, creeps drop coins. Player must pay attention and collect them as they drop, otherwise they will roll off the game board.

This describes a very fast, reflex-oriented action loop. Is it interesting and enjoyable? That is for the designer to decide, from experience or playtesting, and potentially replace it with a different one if this one doesn't work well.

Similarly, we can imagine larger loops, for example:

> Every wave of creeps grows more and more powerful. But the player can spend large numbers of coins on spells, which change what creeps get spawned – for example, there could be spells to make them spawn weaker, or slower, or with a lower attack rate.

Now this describes a potential long-term loop, which lets the player strategize: if they built up a lot of attack towers, for example, they might want to invest in a spell that makes the creeps weaker, to make their tower specialization even more useful. And since the coin cost is high, this is a long-term plan that needs to be worked towards, and prioritized vis-à-vis spending coins on other things such as more towers or necessary repairs.

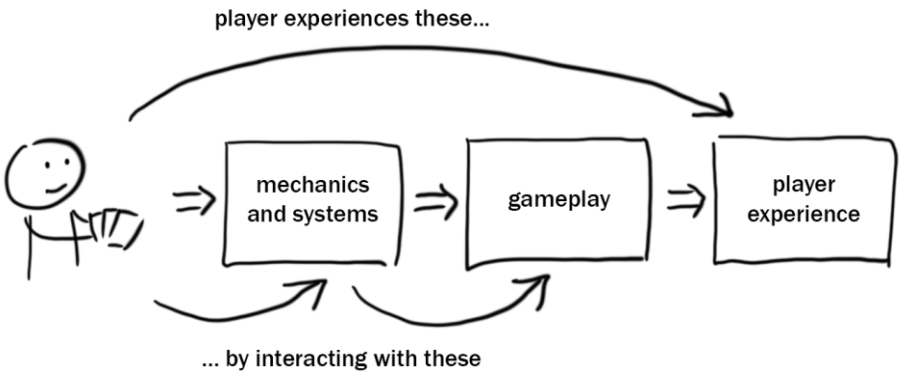
Prototyping and playtesting

Once gameplay loops are designed "on paper", it is very important to implement them as soon as possible and actually see how they feel in real life. The implementation can be *in-game,* or as a *standalone prototype,* depending on whether the loops interact with other systems or can be separated out.

The reason for immediate implementation is that gameplay loops often feel different live than how they read on paper. Narrative description can easily bias the reader to imagine a different experience, a better experience, than the one actually produced by the implementation. And so, it is important to figure out the actual experience of gameplay quickly, before we start layering more loops on top. Skilled designers can sometimes infer some of this live behavior just from experience, but even so, there is no substitute for actual working implementation.

From gameplay to player experience

With this discussion of gameplay loops, we are finally ready to close the loop on player experience. Recalling our diagram:



We started the discussion by introducing player experience: the "fun", the enjoyment of playing the game: maybe being challenged by a tricky puzzle, or being dropped in a fantastic virtual world, or the tranquility of taking care of happy animals on a peaceful virtual farm.

Now we can see how it is tied to gameplay. The experience is related to the player being interested in the game, and interacting with the gameplay loops, progressing and advancing through it, having a variety of interesting goals and activities, good feedback, and of course, intrinsic interest in the activity.

And if all the stars line up right, they will enter that delightful state of flow, where time warps and the world falls away, and the enjoyment of the activity takes over.

Further reading

Gameplay loops:

Gameplay loops are well known in game design, perhaps the earliest preserved mention of them is in Will Wright's 2003 GDC talk "Dynamics for Designers" (https://www.youtube.com/watch?v=JBcfiiulw-8).

They are more recently discussed in detail in "Loops and Arcs" by Dan Cook (http://www.lostgarden.com/2012/04/loops-and-arcs.html)

See also *Advanced Game Design* by Sellers (TODO REF).

Mastery and Flow:

Short overview: Csikszentmihalyi, M.; Abuhamdeh, S. & Nakamura, J. (2005), "Flow", in Elliot, A., Handbook of Competence and Motivation, New York: The Guilford Press, pp. 598–698 (TODO REF)

Popular science overview: *Flow: The Psychology of Optimal Experience* by Mihalyi Csikszentmihalyi (TODO REF)

Learning as Fun:

Theory of Fun by Raph Koster (TODO REF)

"Theory of Fun: 10 Years Later" by Raph Koster (https://www.raphkoster.com/gaming/gdco12/Koster_Raph_Theory_Fun_10.pdf)