Game Design Theory
Part 3: Systems and Gameplay

Dr. Robert Zubek, SomaSim LLC

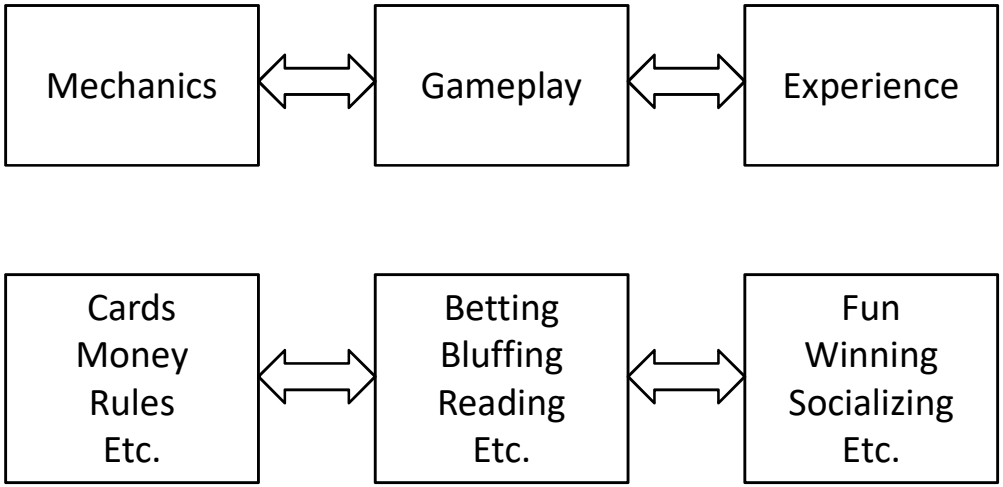
EECS-397/497: Game Development Studio
Winter Quarter 2018
Northwestern University

Let’s recap our model so far

Mechanics
- Game elements, rules, code
- Player’s inputs and outputs

Gameplay
- How the game unfolds over time
- Activity / behavior/ patterns of play

Experience
- The feels / the fun



Player experience (recap)

Player's feeling of "fun"

Different players want different kinds of experiences

Designers can't create it directly –

- need to implement it via mechanics and gameplay that bring it about

Mechanics (recap)

The basic building elements of games

- Objects / nouns
- Actions / verbs
- Rules / grammar

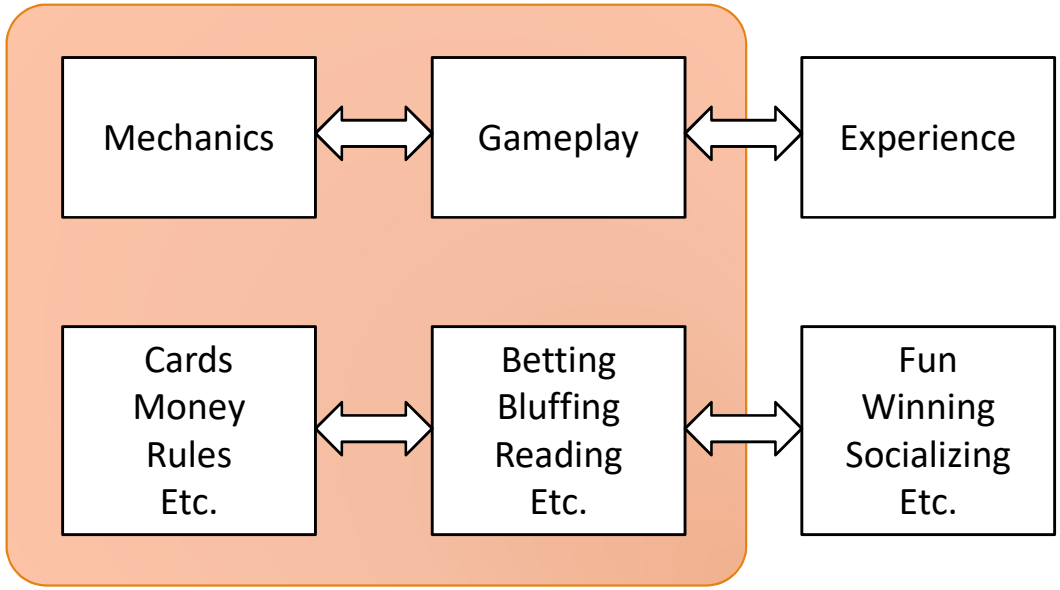
This tells us how we can build something

- But not what we should build, or how it will work

Roadmap

Today

- Systems
- Gameplay



Gameplay

How the game unfolds over time

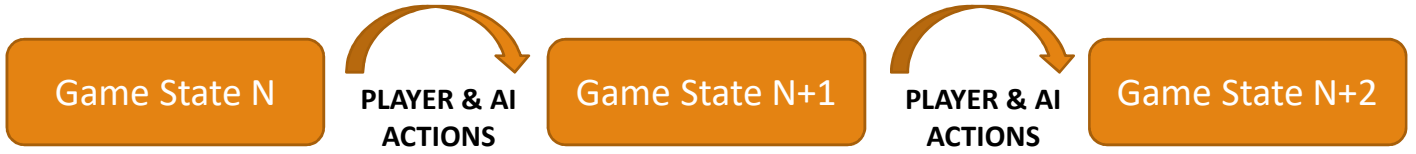
- Arises from players figuring out how to win, given the rules
- Arises from game state being automatically transformed via game rules or code

We take a ***cybernetic*** approach:

- The **player(s)** and the **game** form a **feedback system**

Gameplay

Game as a state machine:



Described as a system (or collection of systems) that evolve over time

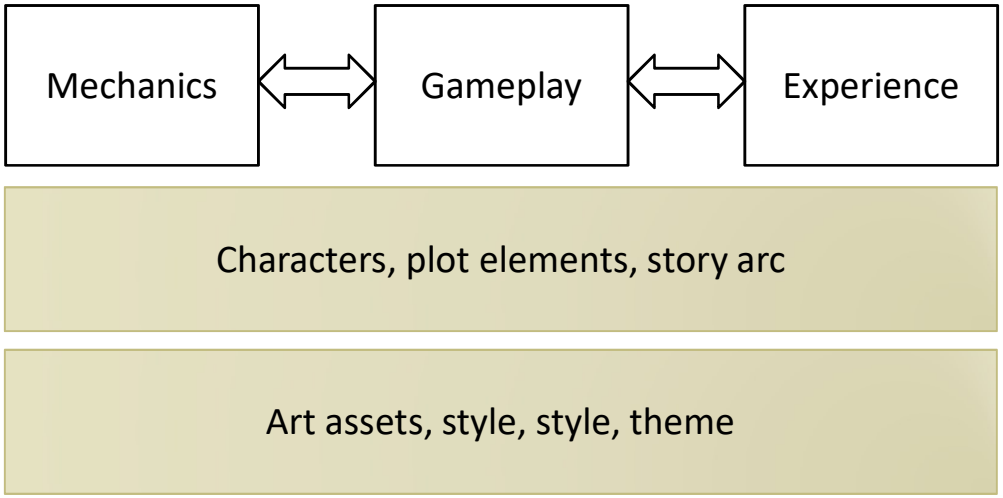
- Game state + rules + player input → new game state
- Positive / negative feedback causing divergent / convergent system behavior
- “Onion diagrams” help us visualize loops at different frequencies working together

Gameplay …is not the only thing

Note: this model only talks about things connected to gameplay

Other aspects also influence player's experience of the game:

- Art style, setting, visuals
- Story, characters, plot
- Etc.



Some examples…

Example from DIABLO III

Venture into the wilderness

- Run into and kill hordes of monsters
- Monsters drop loot and items
- Collect into inventory and later head back to camp

Back in camp

- Sell inventory for gold
- Buy/fix weapons and armor, learn new skills
- *Maybe* use experience points and level up characters

Venture back out

Example from DIABLO III

Large variety of mechanics:

- character specialization
- units
- melee combat
- ranged combat
- stats
- weapons
- equipping
- spellcasting
- upgrades
- loot drops
- carrying capacity
- currencies

Easier to talk about larger building blocks

Combat
- → Interactions between weapons, items, characters, enemies, etc. to determine damage and wins

Inventory
- → Pick up loot after a fight, some items can be used (eg. weapons, armor) others can be brought back to camp
- → Inventory and items have limitations

Economy
- → Gold is a currency
- → Sell loot or items to get gold
- → Use gold to buy items to use, buy spells, crafting materials, etc.

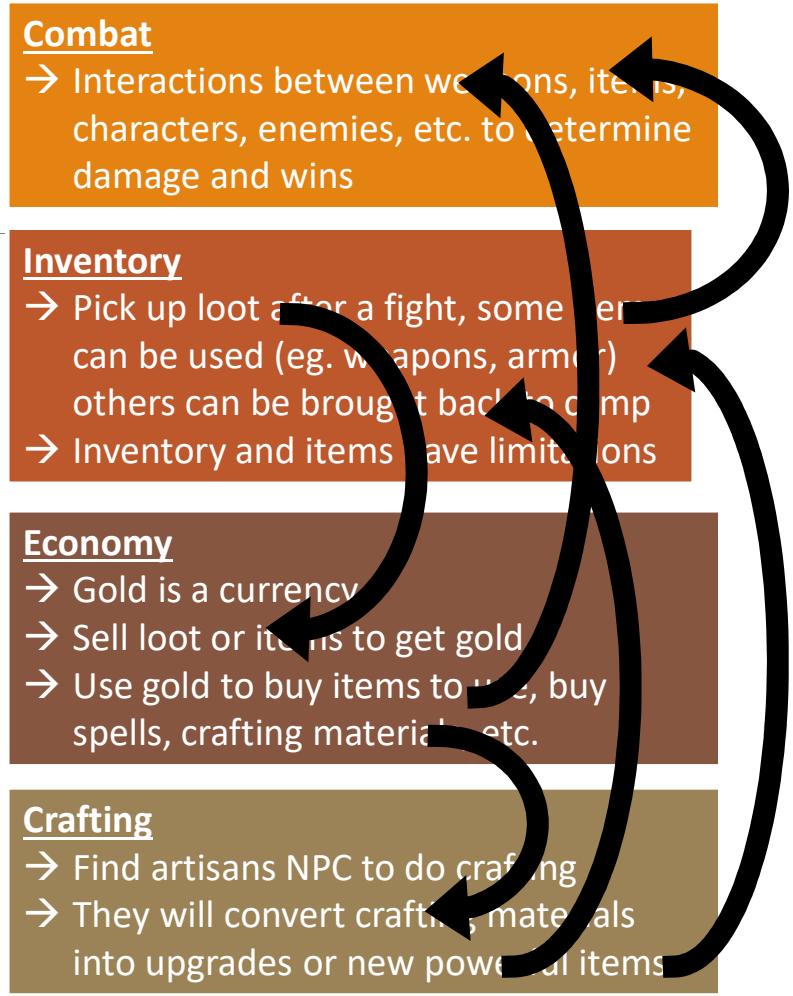
Crafting
- → Find artisans NPC to do crafting
- → They will convert crafting materials into upgrades or new powerful items

Systems

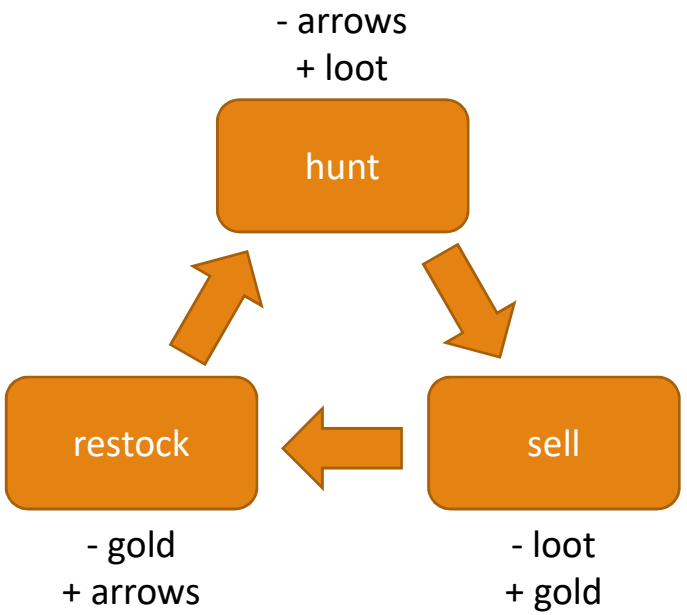
Systems are collections of mechanics that interact together

We want to design them to be
- interesting
- sustainable (don't go off the rails)
- interlocking

But it's easier to analyze each system if we assume some encapsulation :)



Resource loop



Describes how resources can be converted

- (not for free: each “move” requires work)

Example: resource loops in RPGs

Killing monsters gives you
- Gold
- Experience

Gold gives you
- Weapons
- Ammo
- Armor
- Health/mana/whatever

Weapons, etc. let you kill more monsters

(this example courtesy of Ian Horswill)

Example: resource loops in RPGs

Monsters have cash value

But cash has “weapon value”

- So monsters have weapon value
- Cash has monster value
- In some sense they’re all interconvertible

In tuning you game, you need to get the “exchange rates” just right

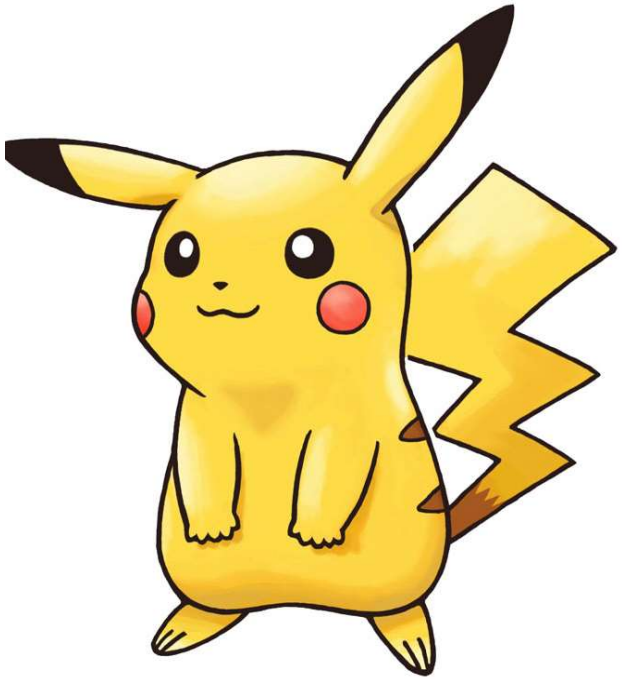
- Or the feedback loops behave badly

(this example courtesy of Ian Horswill)

Example: Pokemon Hunter

Suppose

- Pikachu takes 10 bullets to kill on average
- And a Pikachu pelt is work 50 GP
- And bullets are 10GP each



What does this tell you?

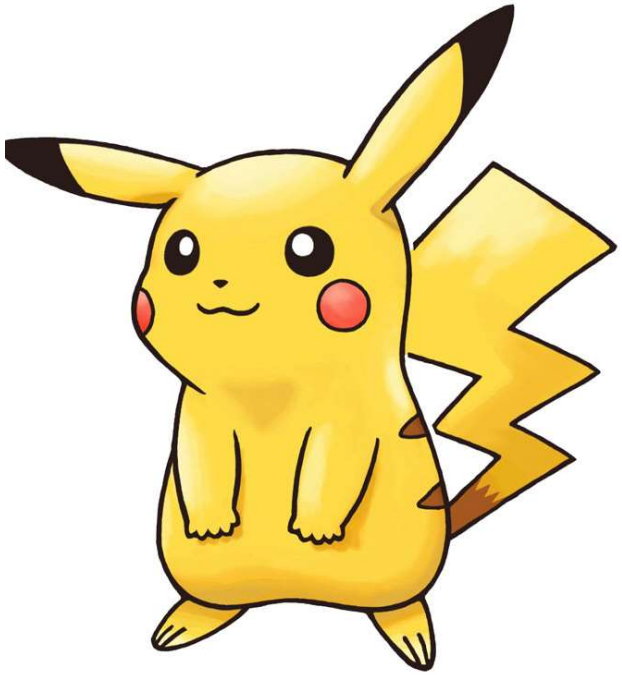
(this example courtesy of Ian Horswill)

Don’t mess with the Pikachu

It’s not worth it

You’ll use 100GP worth of bullets

To get back 50GP



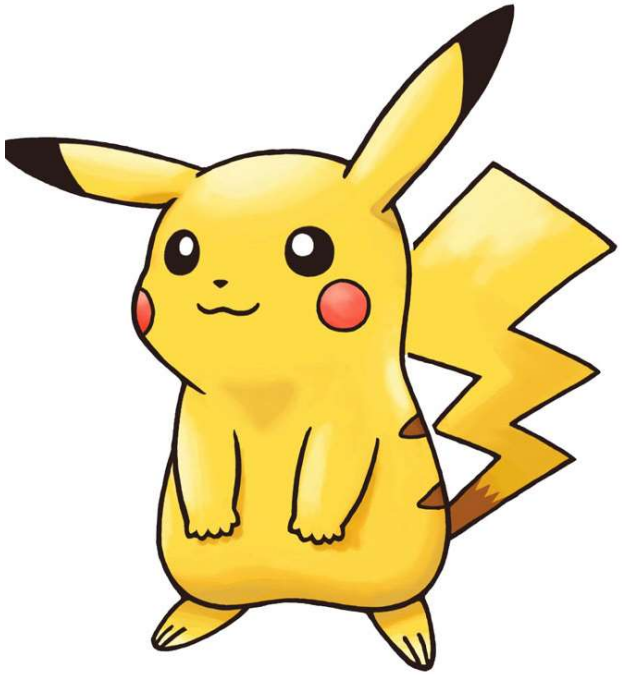
(this example courtesy of Ian Horswill)

Simple example

Suppose

- Pikachu takes **1 bullet** to kill on average
- And a Pikachu pelt is work 50 GP
- And bullets are 10GP each

What does this tell you?



(this example courtesy of Ian Horswill)

Massive Pikachu depopulation

Easy to kill

Pays well

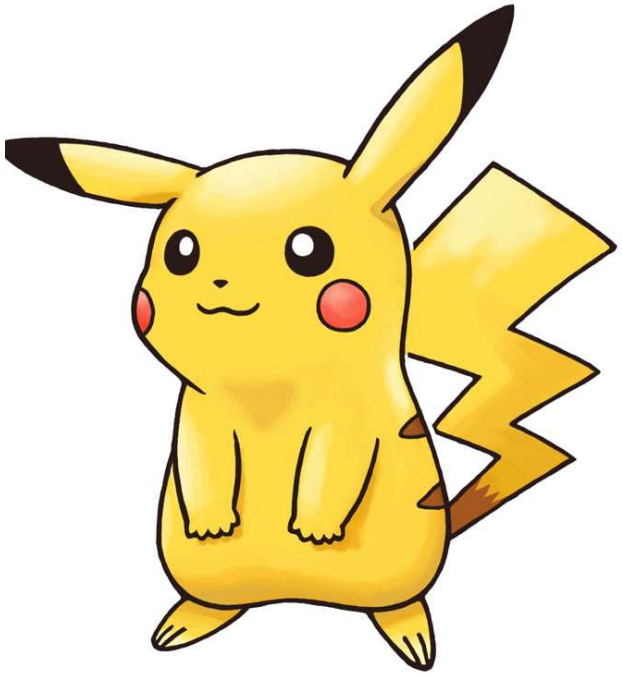
If

- There's an infinite population of them

Then

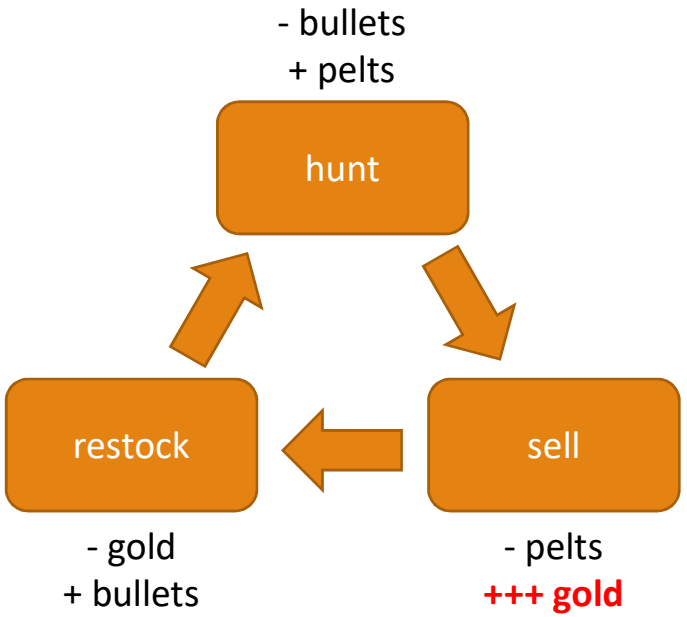
- you've given people an infinite source of money they can mine forever

Money is free!



(this example courtesy of Ian Horswill)

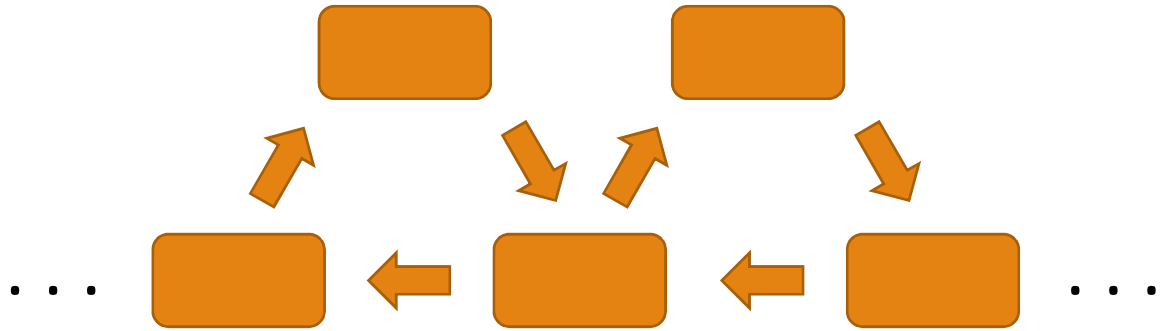
Resource loop



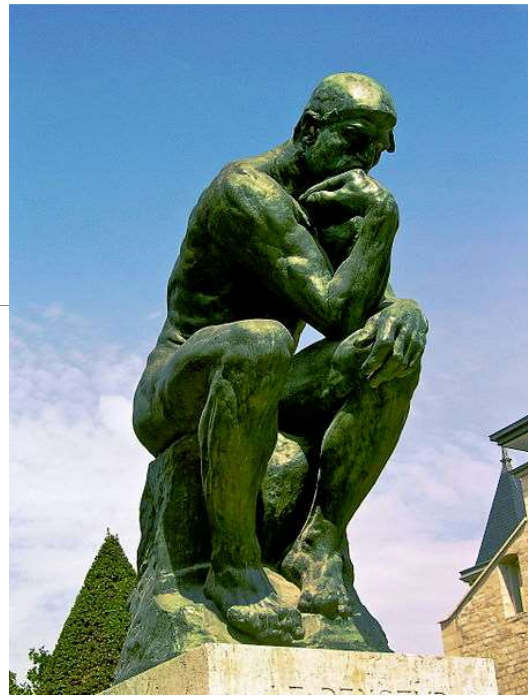
← Need to watch out for bad tuning

Resource conversion chains

These resource conversions can chain together to form interesting behaviors…



System analysis



1. Imagine how players will interact with the mechanics
2. Imagine how the interaction will evolve over time
   - → Is it enjoyable?
   - → Is it interesting?
   - → Does it have defects that will affect player enjoyment?
     (eg. goes off the rails, exhibits “degenerate strategies”, etc)

→ **SYSTEM BEHAVIOR IS REALLY HARD TO PREDICT WITHOUT PLAYTESTING ☺**

Example: “beer distributor” game

Classic supply chain management game taught in business schools :)

We’ll play this in class

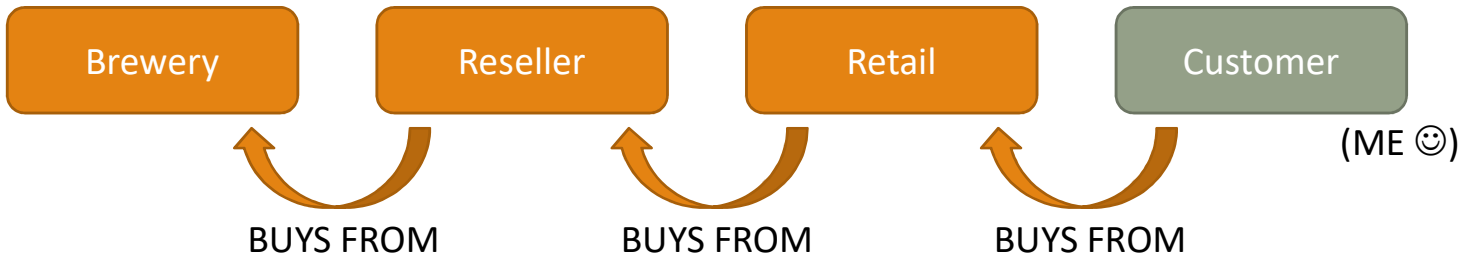
Divide into three groups:

(ME ☺)

Example: “beer distributor” game

Each group has an inventory of beer, and can buy more from the previous group

- At the **beginning of each turn**, each group decides how much to buy and **sends a “buyer”**
  - Manufacturer just makes beer ex nihilo during that time :)
- At the **end of the turn**, buyers come back with beer
- Then each group updates inventory, updates cash on hand, and we repeat



Let's play

Economics:

- You gain $$$ from each keg you sell to your customer
- You spend -$$$ to buy a keg from your vendor
- You lose -$2 each turn from each keg in inventory (inventory penalty)
- You lose -$5 for each keg order that couldn't be fulfilled (backorder penalty)

← Each group has a different sell price but the profits are the same

We'll play for 6 turns

- Start with $300 and 3 kegs in inventory
- Keep track of your inventory and budget in the chart (handout)
- **No chatting** between teams! :)

Let's play

Economics:

- You spend **-$$$** to buy a keg from your vendor
- You gain **$$$** from each keg you sell

- You lose **-$2** each turn from each keg in inventory (inventory penalty)
- You lose **-$5** for each keg order that couldn't be fulfilled (backorder penalty)

No chatting between teams! :)

	Turn 1				
Starting inventory	3				
Starting cash					$ 300
Purchased kegs	2	@	$ - 30	=	- $ 60
Sold kegs	-3	@	$ 40	=	+ $ 120
Ending inventory	2				
Inventory penalty	2	@	$ - 2	=	- $ 4
Backorder penalty	0	@	$ - 5	=	- $ 0
Ending cash					$ 356

Results

Feedback loops

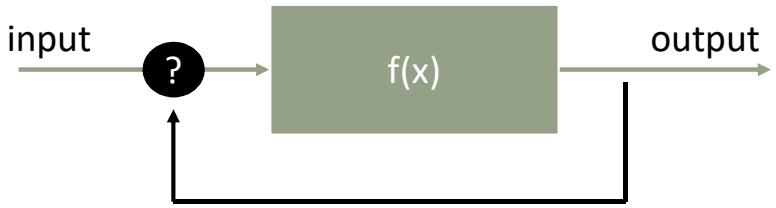
So far we talked about resource conversions that don't change

But what if things changed based on what you did before?

- For example, dynamic pricing

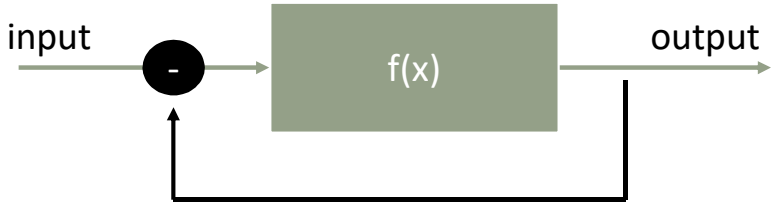
"Feedback loop" is a loop where output at time *t*
affects input at some future time *t+1*

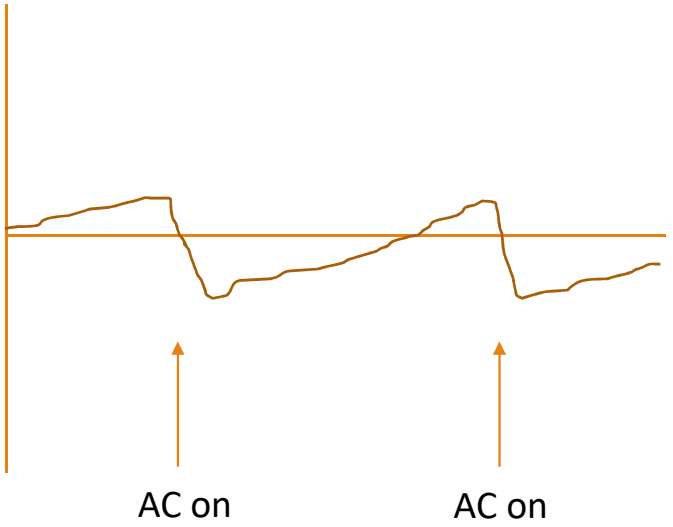
Feedback loops



Thermostat

Negative feedback loop





Human controller

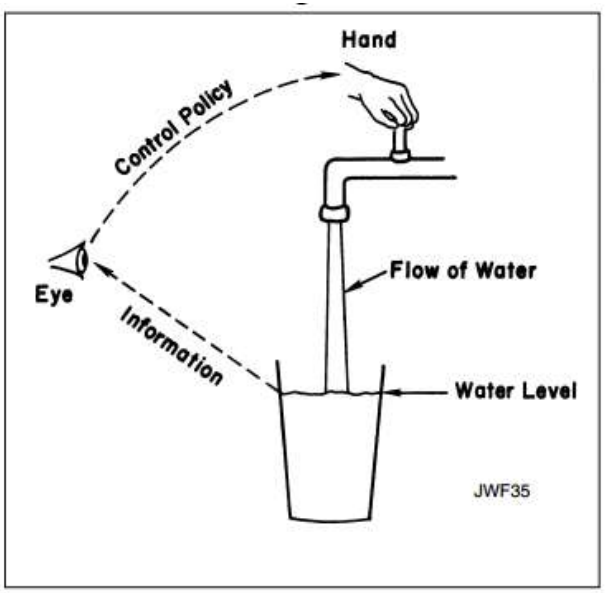
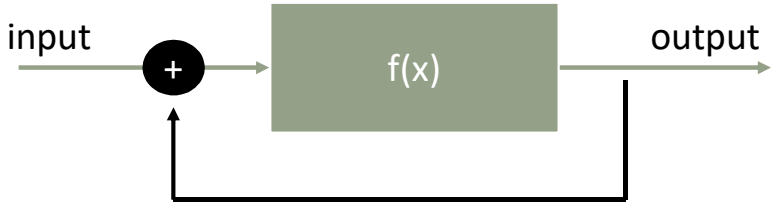
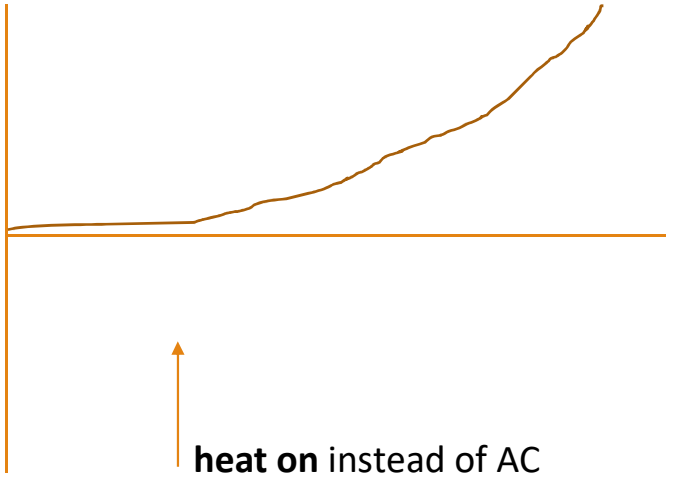


Image: Jay Forrester, “Some Basic Concepts in System Dynamics” http://static.clexchange.org/ftp/documents/system-dynamics/SD2009-02SomeBasicConcepts.pdf

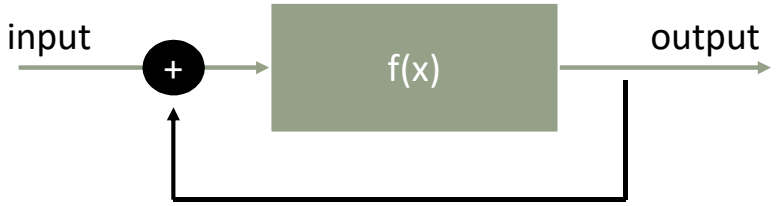
Broken thermostat

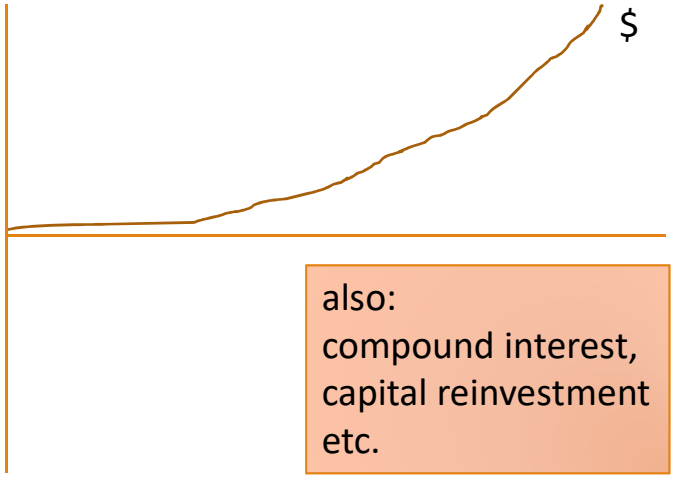
Positive feedback loop





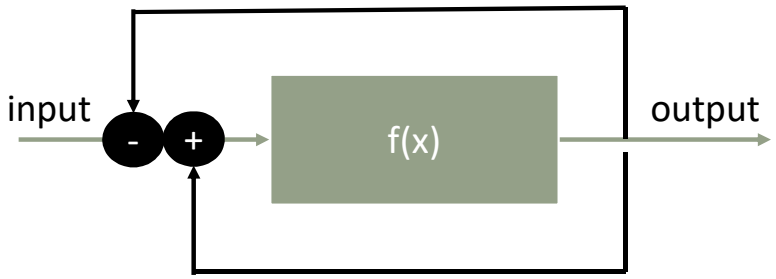
Positive feedback loop



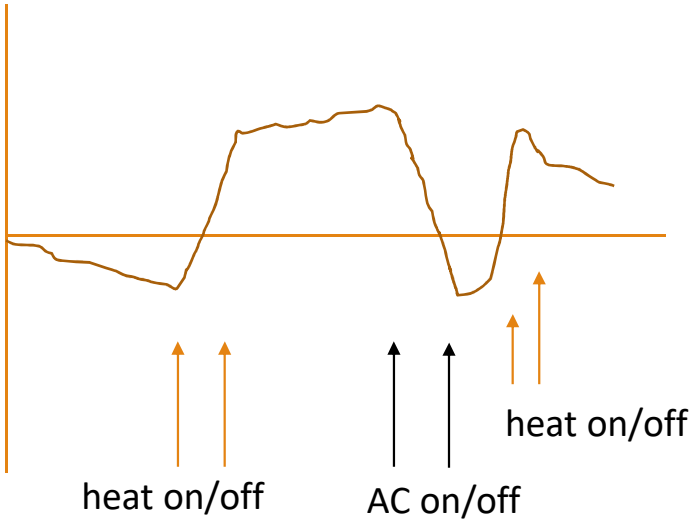


also:
compound interest,
capital reinvestment
etc.

Double thermostat



Starts getting complex
(and slightly ridiculous)

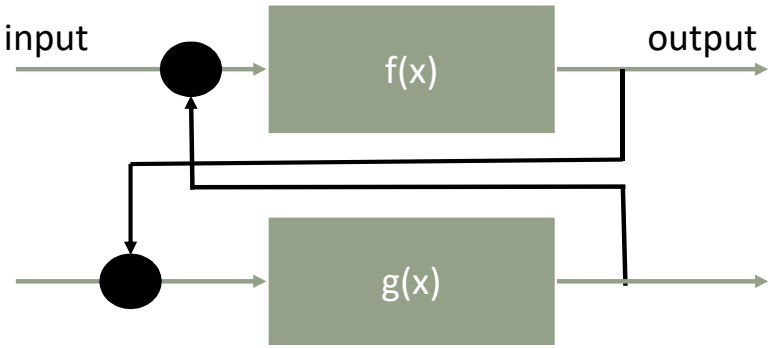


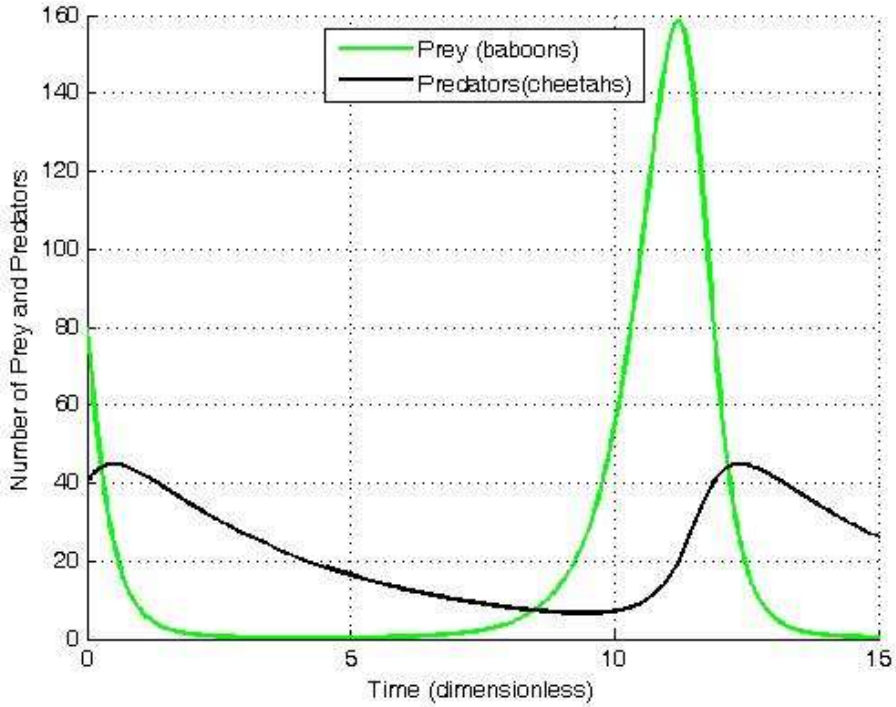
Multiple interdependent loops

Many other examples in nature

Eg. predator / prey models

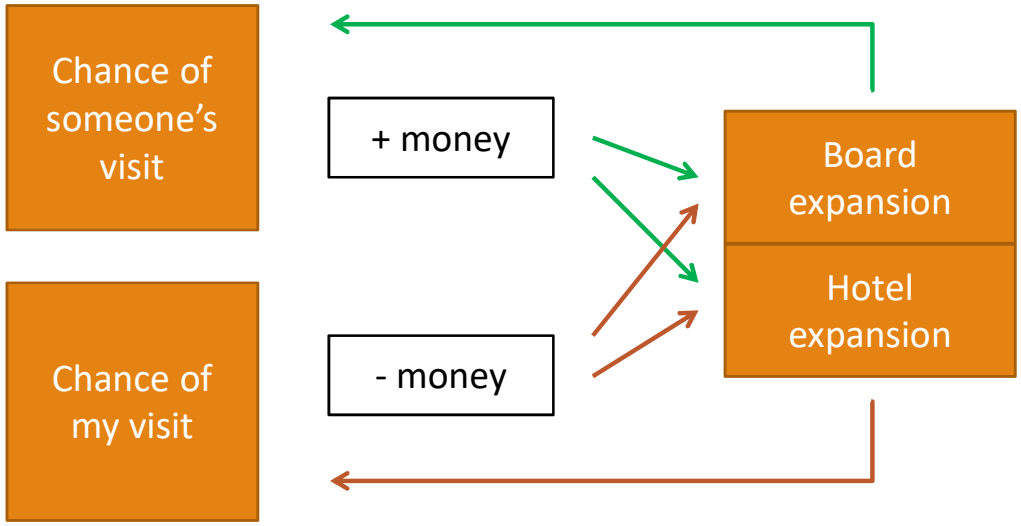
Eg. economics: manufacturing and consumption






MONOPOLY

Simplified!


Chance of someone's visit
Chance of my visit
+ money
- money
Board expansion
Hotel expansion

Rules of thumb

Positive Feedback – Monopoly, StarCraft, Civilization, etc.

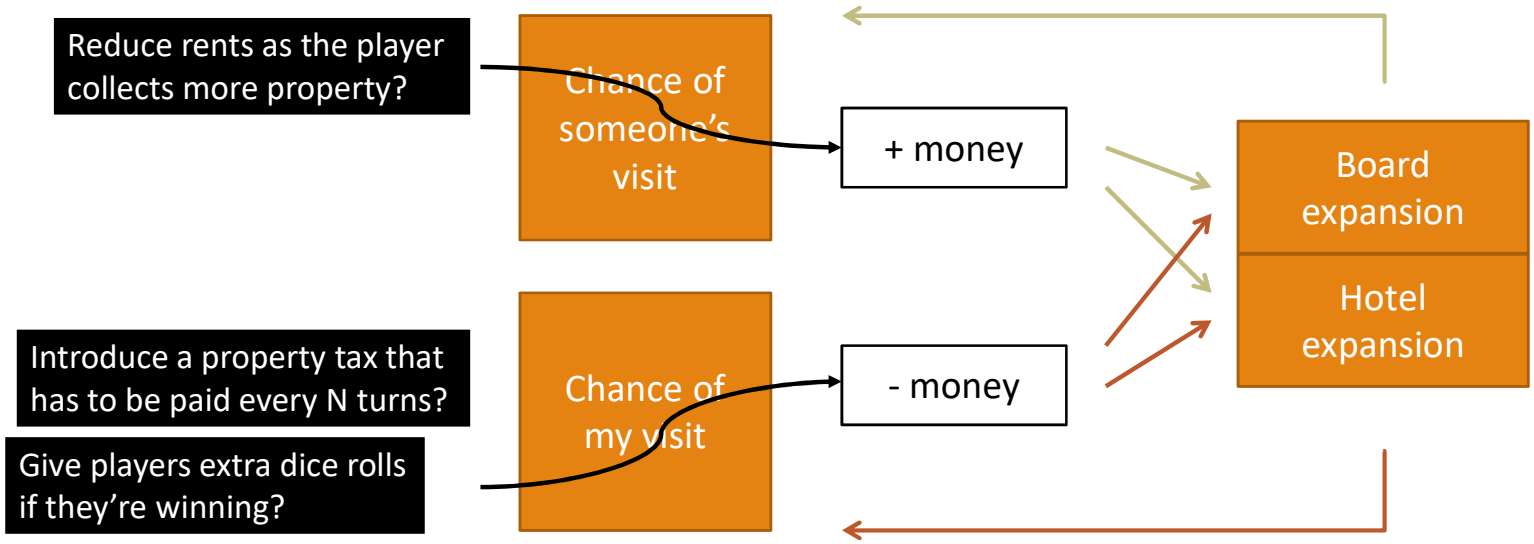
- Early success counts most
- Early winners get entrenched

Negative Feedback – Mario Kart “rubber banding”

- Late successes count most
- Early wins not rewarded (“wasted effort”)

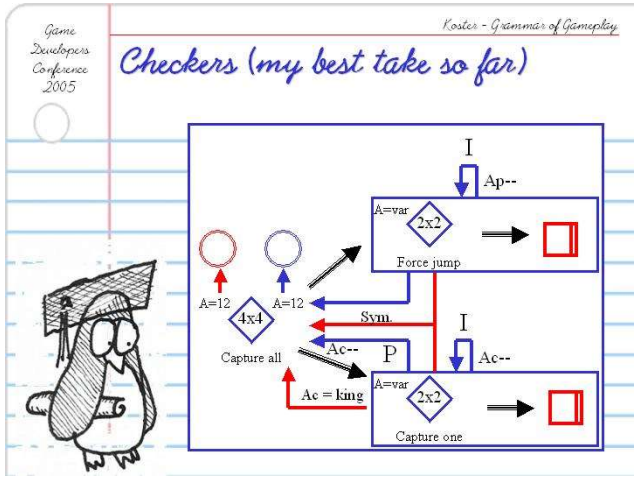
Too much of either feels unfair / not challenging. Need a mix of both.

Can we reverse these feedback loops?

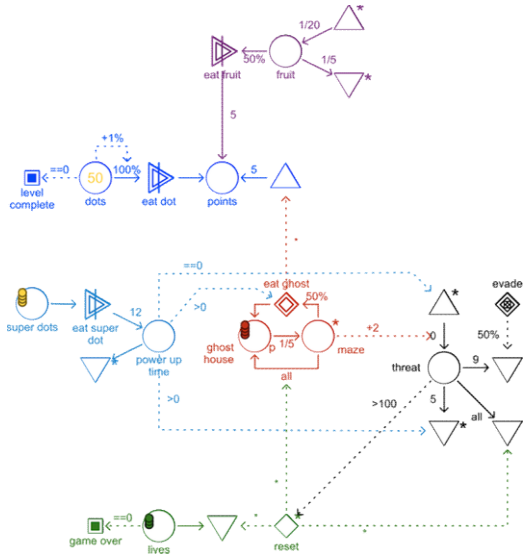


Detailed analysis gets complicated quickly

… so nobody does it



Model of checkers
From Raph Koster, GDC 2005 talk
http://theoryoffun.com/grammar/gdc2005.htm



Model of Pac-Man
From *Machinations* by Dormans and Adams
http://www.gamasutra.com/view/feature/176033

Resource loops vs activity loops

We talked about resource loops – exchanging items, etc

But we should also look at repetitive activity

- Diablo: hunt, collect loot, go back, sell loot, buy stuff, … and repeat
- Sims: wake up, get ready, go to work, come back,
  eat, practice skills, sleep, … and repeat

Players do similar things over and over

→ “Activity loops”

Scale problems

Games usually contain **simultaneous activity loops** that work on **different time scales**

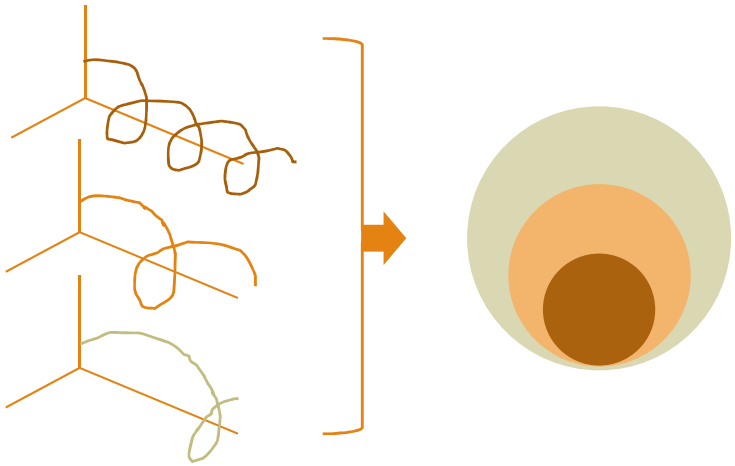
Why time scales? Because we need to make sure our player always has something to do

- Minute to minute
- Hour to hour
- Come back next day, and day after

How to analyze those?

Solution: move up in abstraction

"Onion diagrams": show loops with different frequencies on one diagram



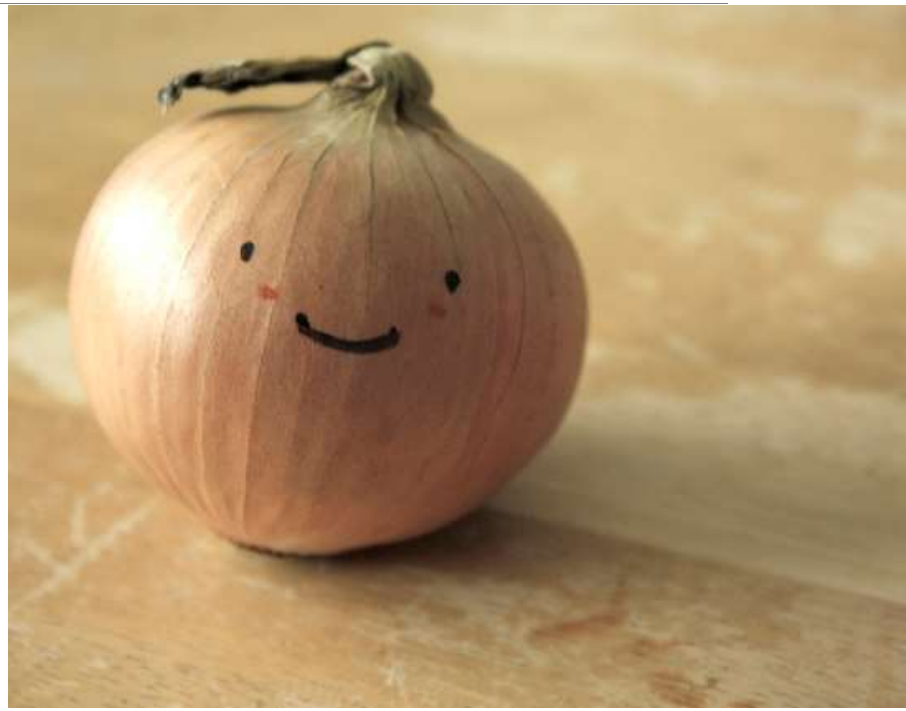
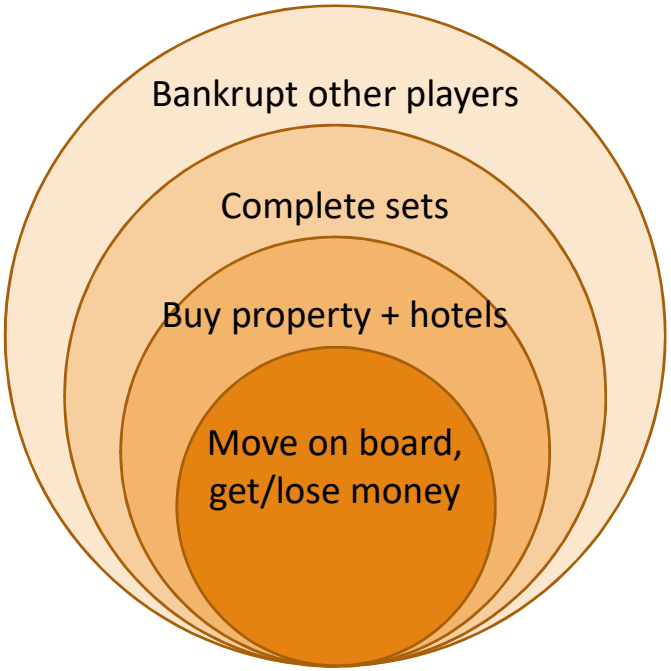


Image credit: https://www.flickr.com/photos/deliriant_o

Onion diagram for MONOPOLY



Every 60 minutes

Every 20 minutes

Every 5 minutes

Every 2 minutes

Another way to diagram loops

Game design “onion”

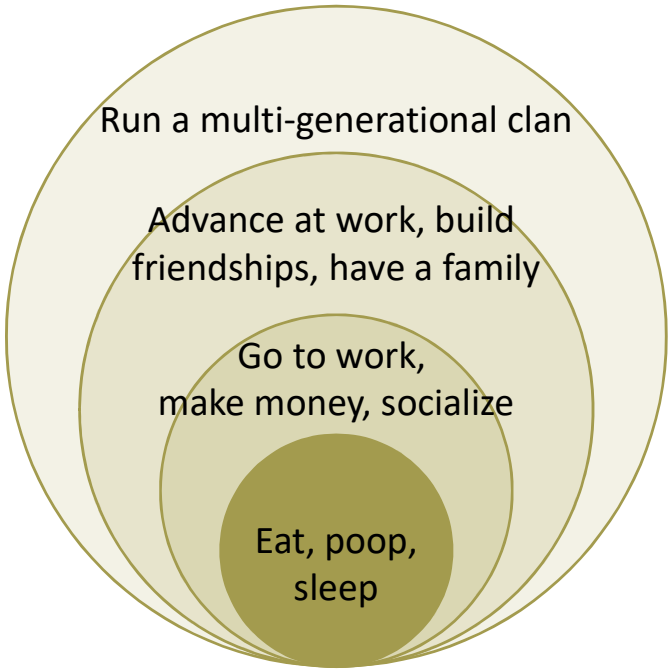
- Fastest / smallest feedback loops in the center
- Larger and larger loops going out

Can show relative time frame

Hides feedback effects!

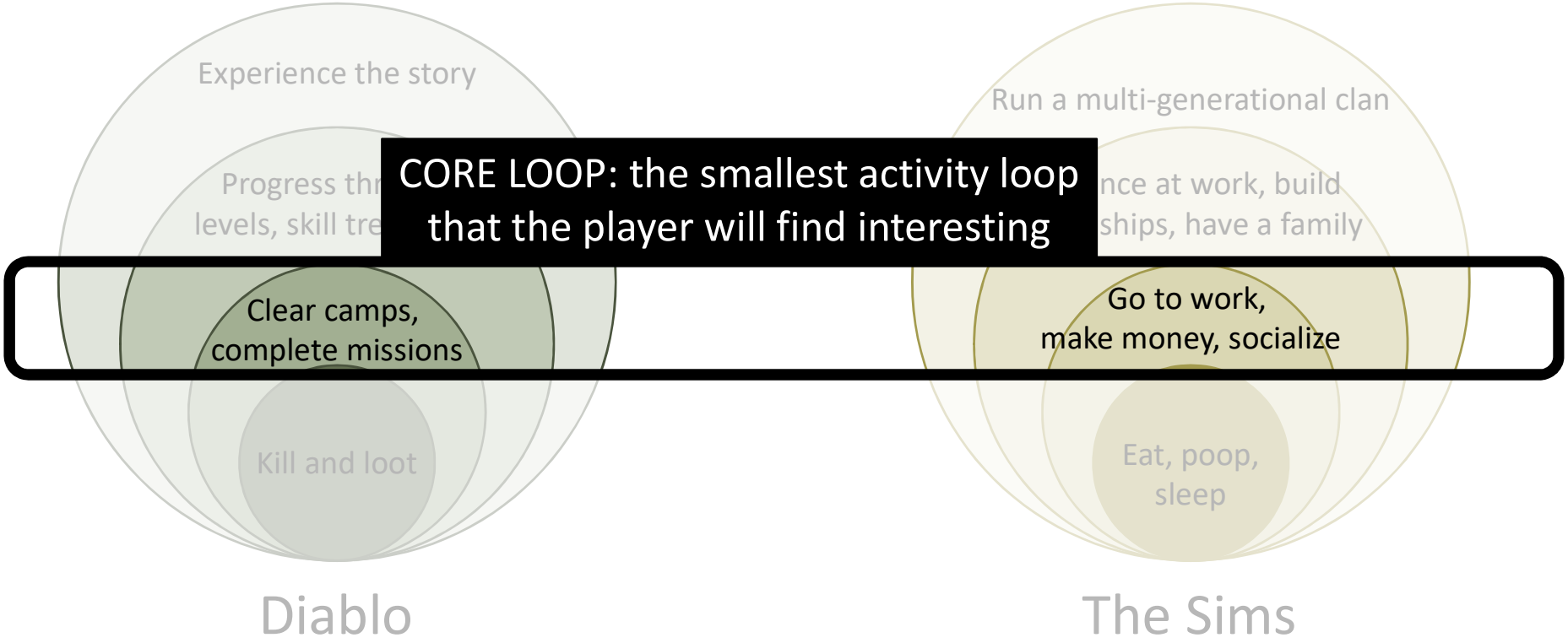
More onions!

Diablo



The Sims

“Core loop”


Experience the story
Progress thr
levels, skill tre
Clear camps,
complete missions
Kill and loot
Diablo
Run a multi-generational clan
nce at work, build
ships, have a family
Go to work,
make money, socialize
Eat, poop,
sleep
The Sims
CORE LOOP: the smallest activity loop
that the player will find interesting

Player motivation and progression mx

1. Players like to be rewarded.
2. Players need to feel it when they're making progress.
3. Progress *of the kind they care about* makes people
   - enjoy the game, and
   - want to keep playing.

Players like to be rewarded

Examples

- XP – always goes up, each small action earns XP
- Levels – reach an XP milestone to level up and unlock reward
- Gold – earn by playing the game, spend on rewards
- Equipment – find by playing the game, use in play

Why does this work?

On a very crass level: let's look at operant conditioning

Reward schedules

Experiments on pigeons (and other animals)

Setup:

- Teach a pigeon to peck to get food
- Vary how many pecks it takes to get food based on time etc (“schedule”)

Observe;

- How much they peck (reinforcement)
- How long until they stop (extinction)

Reward schedules

Types of schedules

- Fixed interval
  - reward every 5 seconds (if pecking)
- Fixed ratio
  - reward every 5 pecks
- Variable interval
  - reward every 3-7 seconds (if pecking)
- Variable ratio
  - reward every 3-7 pecks

(There are more types of schedules, too)

Reward schedules

Types of schedules

- Fixed interval
  - reward every 5 seconds (if pecking)
- Fixed ratio
  - reward every 5 pecks
- Variable interval
  - reward every 3-7 seconds (if pecking)
- Variable ratio
  - reward every 3-7 pecks

Best reinforcement
Lowest extinction

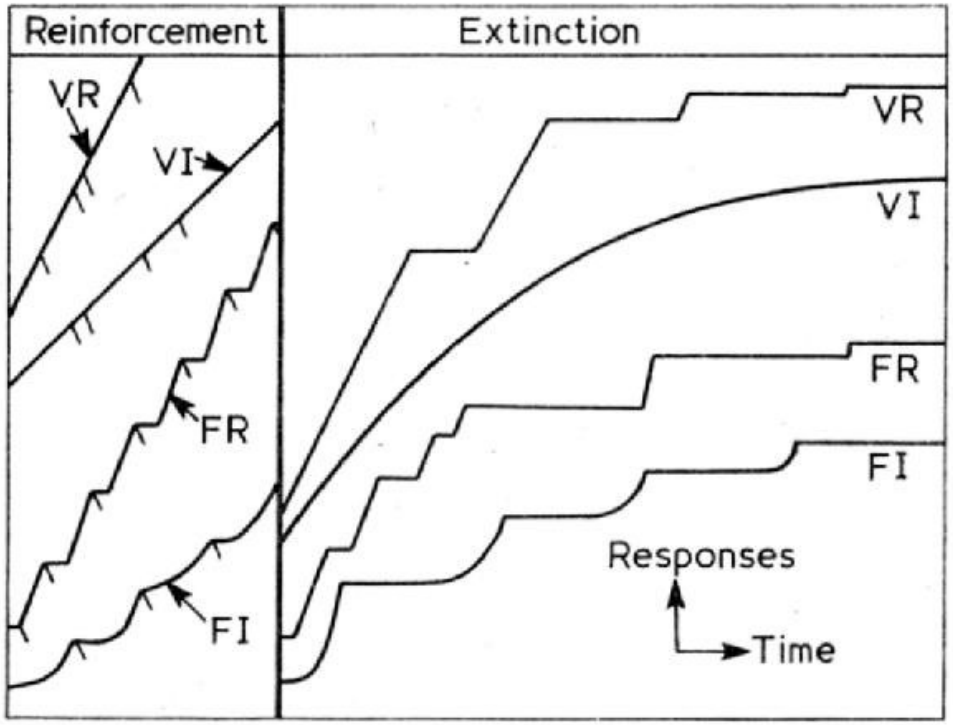


Chart from: Walker, S.F. (1976) Learning and Reinforcement, London, Methuen
http://s-f-walker.org.uk/pubs/lr/lrframechaps.html

Diablo examples

Core loop

- Go out and kill hordes of monsters

Monsters drop gold and loot randomly

- Variable ratio schedule, like **slot machines**
- Reward correlated to monster level
  - Which is correlated to your player level
  - This encourages you to level up

Rarely, also drop collectible armor pieces

- Big reward like **jackpot** payouts
- **Second** variable reward schedule **layered on**

Diablo examples

Loot chests scattered around levels

- Need to scout the area to find them
- Sometimes need extra effort/skills to open
- Drop some variety of random items (or nothing)

Variable ratio schedule strikes again!

- But rewards *exploration*, rather than *killing*
- Different work/reward rate keeps things interesting

Progression MX

1. Players like to be rewarded.

2. Players need to feel it when they're making progress.

3. Progress *of the kind they care about* will result in wanting to play more.

Diablo example

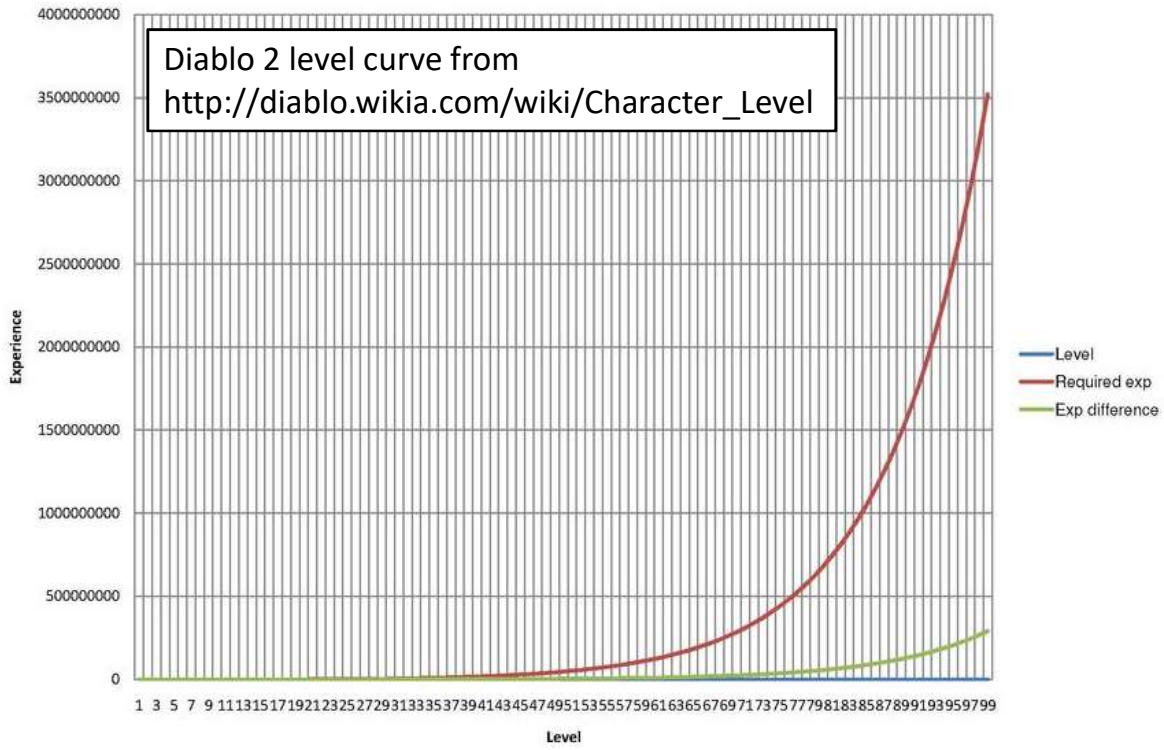
Many ways to measure progress:

- Playing the game gives you XP
- XP causes level ups
- Level ups unlock more abilities, better equipment, harder monsters

Feedback loop: harder monsters → more XP → faster accumulation → …

Level curve – map from XP to level

- Superlinear to match the feedback loop



Diablo example

Player will want to unlock more abilities, new areas to explore, etc.

What other things will give a sense of progression?

- Gameplay items (weapons etc)
- Cosmetic items (clothing etc)
- Status items (badges, achievements)

There are many ways to give players a sense of progression.

Find what works for your game.

Metagame

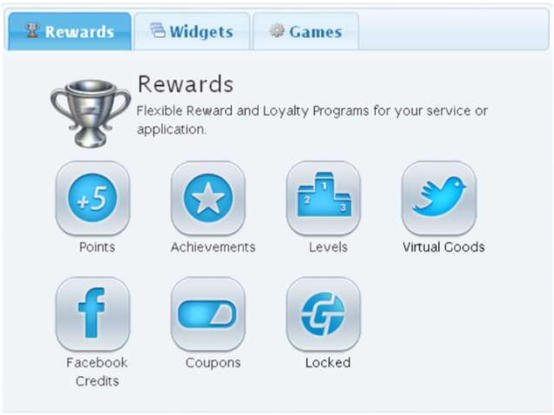
Progression that happens outside of core loops

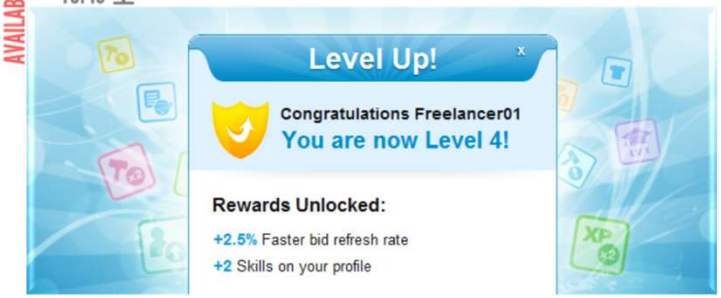
- Character unlocks in a fighting game

- Card game in Witcher 3

- Special unlocks that persist
  between different game sessions

Progression MX without a game…





Question: do progression MX work if the player is not feeling rewarded?

Gameplay ← Mechanics

Gameplay is how the interaction with the game unfolds over time

Mechanics are the rules / game pieces that give rise to gameplay

Next Session

Wednesday 1/24 we'll talk about project preparation

Q&A