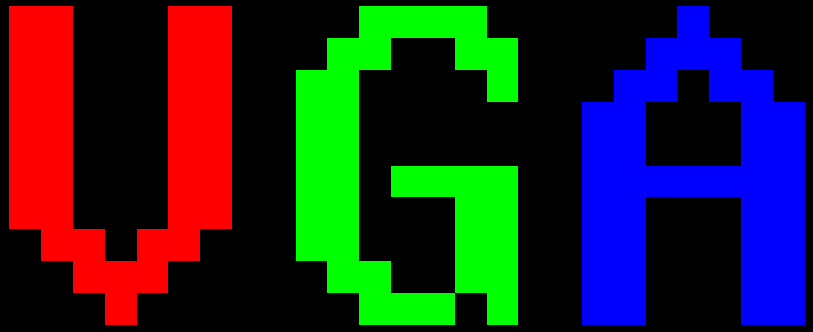


Herní fyzika

Jan Matoušek

29. 10. 2026

Změny v prostředí

- Prostředí hry se neustále mění
  - vlivem hráče,
  - vlivem dalších entit v herním prostředí,
  - podle pravidel herního světa.
- Změny se odehrávají…
  - před začátkem hry (např. součástí procedurálního generování),
  - v reálném čase během hry,
  - občas, v definovaných okamžicích.

Typické procesy ve hrách

- Fyzikální procesy
  - Kinematika
  - Newtonovská dynamika
  - Další oblasti fyziky
- Simulace velkých systémů
  - Společnost, počasí, eroze terénu…
- Procesy podle herní logiky
  - Vzájemné interakce herních entit
  - Počítačem ovládané entity

Propojení s herním designem

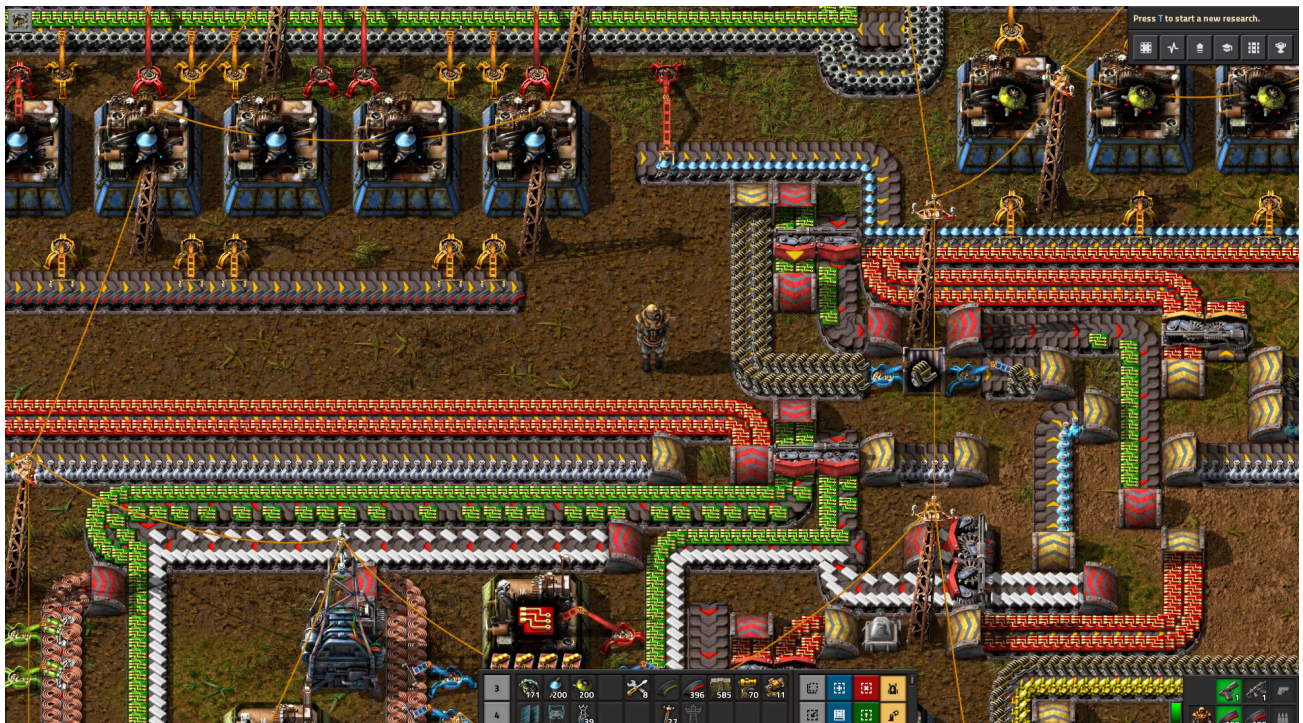
- Herní mechaniky diktují potřebnou dynamiku
- Požadavky herních mechanik spolu s výpočetními možnostmi určují přesnost a rozsah simulací.
- Simulace můžeme podle toho dělit na:
  - nezbytné pro herní mechaniky,
  - důležité, ale nemusí být přesné,
  - volitelné, dekorativní.

Příklad: *Fish Fillets* (1998)



- Pohyb ryb a předmětů, kolize, hmotnost, simulace podvodního prostředí vlněním pozadí.
- Fyzika má vlastní (jednoduchá) pravidla podle mechanik logické hry.

Příklad: *Factorio* (2016 / 2020)



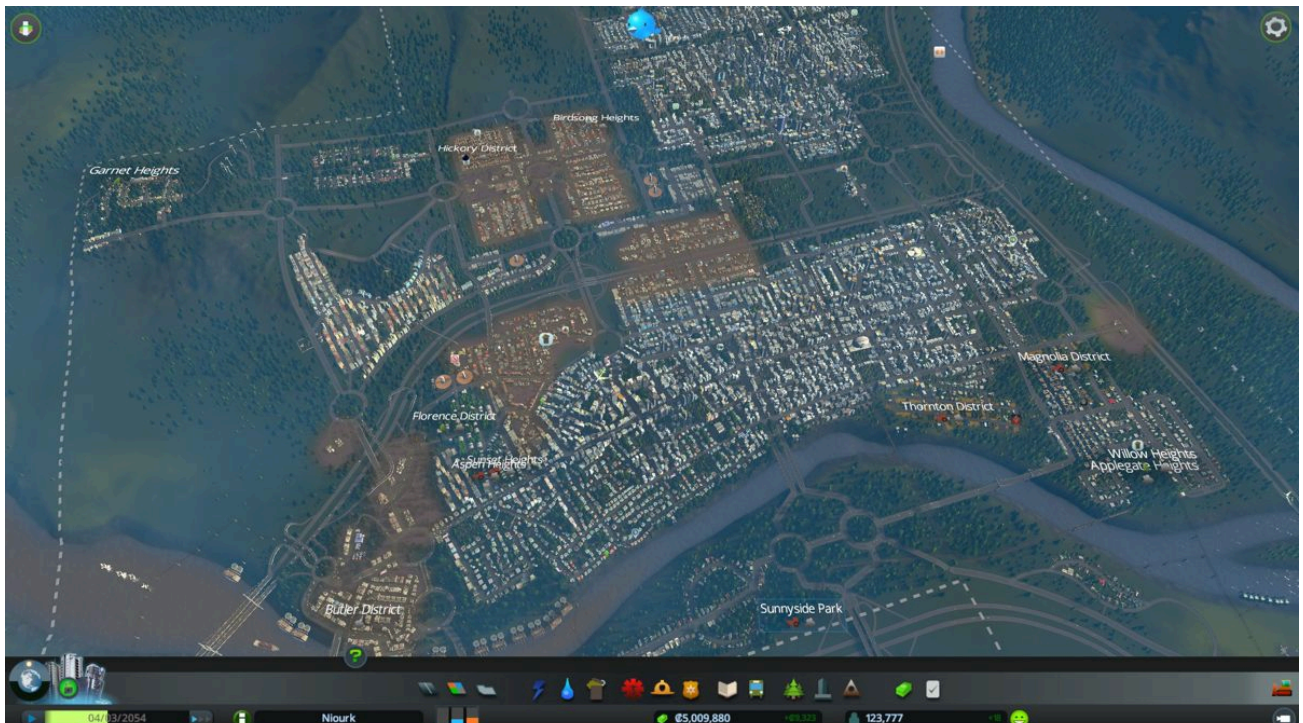
- Pohyb hráče a nepřátel, ruce, pásy, potrubí, vlaky, drony, znečištění prostředí, požáry, denní cyklus, elektřina, projektily, exploze, jedovatý sliz.

Příklad: *The Sims* (2000)



- Životní potřeby simíků, mezilidské vztahy, denní cyklus, ekonomika a zaměstnání, vytopení koupelny, šíření ohně, poruchy přístrojů, zkažení jídla, pohyb osob.

Příklad: *Cities Skylines* (2015)



- Ekonomika, znečištění vzduchu a vody, pohyb vody, jednotlivé služby města (hasiči, policie, zdravotnictví, školství…), obyvatelé, doprava.

Příklad: *Quake* (1996)



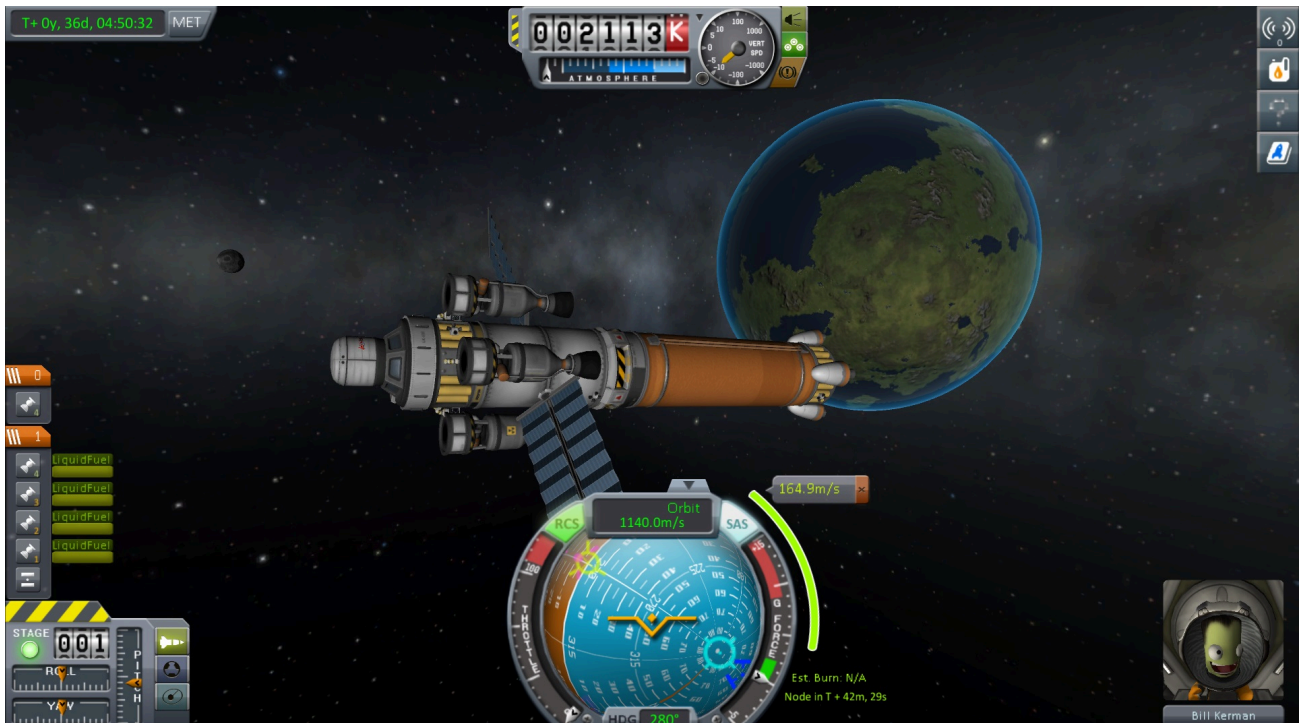
- Pohyb hráče a nepřátel, běhání, skoky, plavání, kolize. Pohyby střel, exploze.

Příklad: *Half-Life 2* (2004)



- Pohyb hráče a nepřátel, běhání, skoky, plavání, kolize. Pohyby střel, exploze.
- Fyzika pevných těles, ragdoll, fyzika vozidel, magnetismus.

Příklad: *Kerbal Space Program* (2015)



- Orbitální mechanika, spotřeba paliva, kolize umělých vesmírných objektů.

Příklad: *Spintires* (2014)



- Fyzika vozidel, spotřeba paliva, bahno, jeřáby.

Příklad: *Bridge Builder* (2000)



- Příhradové konstrukce.

Způsoby simulací

- Naivní, podle jednoduchých pravidel
- Grafové algoritmy
- Numerické výpočty
- Celulární automaty
- Neuronové sítě

Náročnost simulací

- Čím přesnější a věrnější simulace, tím více dat je potřeba zpracovat a tím složitější postupy je potřeba volit.
- Zejména interakce každý s každým jsou výpočetně velmi náročné.
- V návrhu hry balancujeme se spotřebou výpočetních prostředků a náročností vývoje.
- Pro méně důležité simulace volíme jednodušší postupy, např.
  - použít různé způsoby simulace blízkého a vzdáleného prostředí,
  - použít jednodušší a méně přesné algoritmy,
  - snížit počet/rozsah simulovaných entit,
  - převést systém na animované a skriptované sekvence,
  - nesimulovat.

Simulace v reálném čase a multithreading

- Simulovat lze po fixních kvantech v rámci generování snímku.
- Simulaci lze oddělit do jiného vlákna a zvolit jinou frekvenci simulace nezávislou na generovaných snímcích hry.
  - Při vykreslování může být potřeba provádět interpolace stavů entit.
- Při použití více vláken je potřeba dbát na správnou synchronizaci přístupů k paměti.
- GPU je dnes na některé paralelní operace vhodnější než CPU.
  - Lze využít compute shadery, zejména k výpočtům na mřížkách.
  - Díky tomu lze ušetřit přenosy dat mezi simulací a vykreslováním.
  - Synchronizaci přístupů k paměti je nutno stále řešit.

Animace a efekty

Pohyb po předem dané dráze

- Pohyb entity po předem dané dráze, např. úsečka, křivka
- Plynulý přesun mezi dvěma body je švenk (*tween*)
- Vzorec pohybu entity bývá explicitní (z času vypočítám polohu)
  - neočekává se vnější zásah do pohybu
- Různé druhy pohybů:
  - rovnoměrný – u některých křivek může být těžší zařídit
  - zrychlující se, zpomalují se (a kombinace; *easing*)
- Švenk nemusí plynule měnit jen polohu (a rotaci), ale i jiné vlastnosti (např. výši skóre)

Chaotický pohyb (1)

- Každá trajektorie lze rozložit jako součet různých oscilací (a základního pohybu)
- Každá oscilace je sinusovka, s amplitudou, periodou a fázovým posunem

$$l(t) + \sum A_i \sin(\varphi_i + 2\pi f_i)$$

- Vhodnou sadou náhodně vygenerovaných oscilací lze vytvořit různé efekty:
  - padající sníh
  - částice unášené větrem
  - stromy a tráva vlnící se ve větru
  - otřes obrazu nebo země při zemětřesení nebo po výbuchu
  - ...

Chaotický pohyb (2)

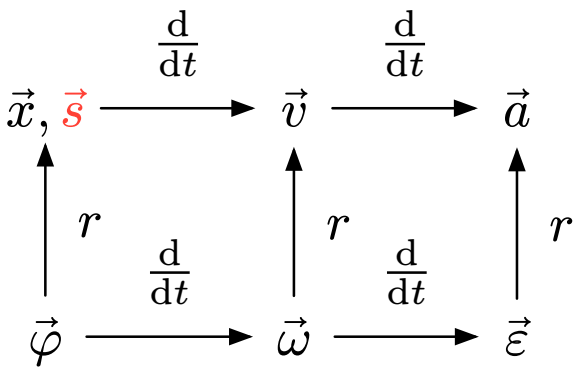
- Využít lze také náhodně generovanou procházku
- Vhodné efekty:
  - motýli, hmyz a ptáci na pozadí scén
  - …

Předpřipravené animace

- Animace mohou být součástí assetů
  - dráhy pohybu objektů
  - střídání snímků a textur
  - skeletální animace (pohyby kostí v objektu, deformace objektu)
- Mohou být vytvořeny
  - ručně
  - komplexní simulací v jiném programu
  - procedurálně ve hře

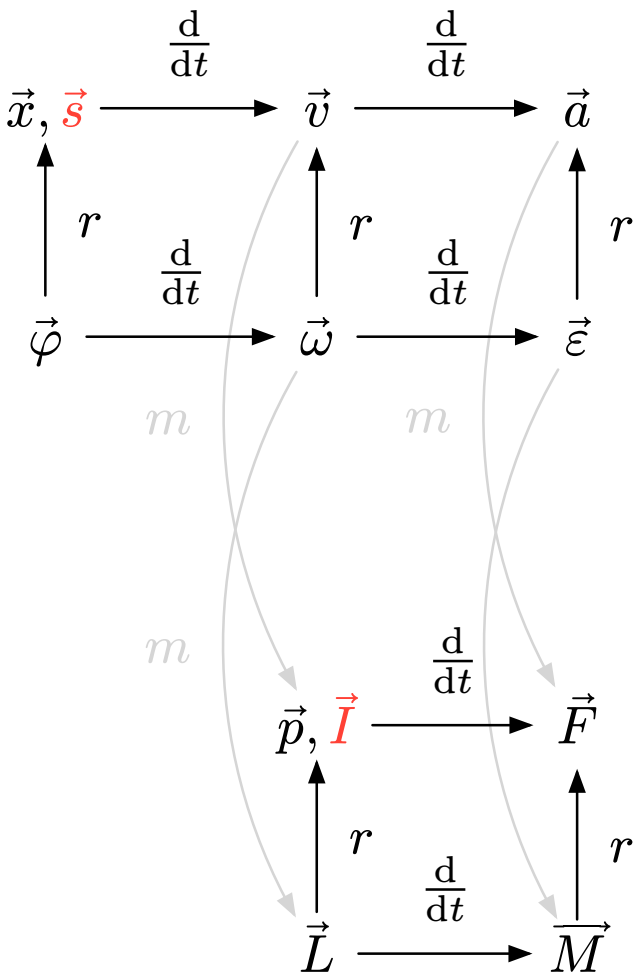
Fyzika

Rychlý přehled kinematiky: Bod a pevné těleso



$\vec{t}$	s	čas	time
$\vec{x}$	m	poloha	position
$\vec{s}$	m	vzdálenost	distance
$\vec{v}$	m/s	rychlost	velocity
$\vec{a}$	m/s^2	zrychlení	acceleration
$\vec{r}$	m	poloměr	radius
$\vec{\varphi}$	1	úhel (v radiánech)	angle
$\vec{\omega}$	1/s	úhlová rychlost	angular velocity
$\vec{\varepsilon}$	1/s^2	úhlové zrychlení	angular acceleration

Rychlý přehled dynamiky: Hmotný bod a pevné těleso



$\vec{t}$	s	čas	time
$\vec{x}$	m	poloha / vzdálenost	position / distance
$\vec{s}$	m	vzdálenost	distance
$\vec{v}$	m/s	rychlost	velocity
$\vec{a}$	m/s²	zrychlení	acceleration
$\vec{r}$	m	poloměr	radius
$\vec{\varphi}$	1	úhel (v radiánech)	angle
$\vec{\omega}$	1/s	úhlová rychlost	angular velocity
$\vec{\varepsilon}$	1/s²	úhlové zrychlení	angular acceleration
m	kg	hmotnost	mass
$\vec{p}$	kgm/s, Ns	hybnost	momentum
$\vec{I}$	kgm/s, Ns	impuls	impulse (J, Imp)
$\vec{F}$	kgm/s², N	síla	force
$\vec{L}$	kg/s	moment hybnosti	angular momentum
$\vec{M}$	kg/s², Nm	moment síly	torque

Integrál

Obvykle chceme spočítat ze zrychlení rychlost a z rychlosti polohu atd.

- Jako uživatelé fyzikálního enginu nastavujeme zrychlení, případně tělesům zadáváme silové impulsy.
- Fyzikální engine počítá od těchto derivací pozpátku, tedy počítá integrál.
- Výpočty derivací nebývají ve fyzikálních enginech běžné.

Příklady integrálů v kinematice a dynamice:

$$\vec{v}(t) = \int_0^t \vec{a}(t)\,\mathrm{d}t + v_0$$

$$\vec{s}(t) = \int_0^t \vec{v}(t)\,\mathrm{d}t + s_0$$

$$\vec{p}(t) = \int_0^t \vec{F}(t)\,\mathrm{d}t + p_0$$

Numerická integrace

V simulaci není čas spojitý. Převedeme diferenciál d na rozdíl a integrál na sčítání.

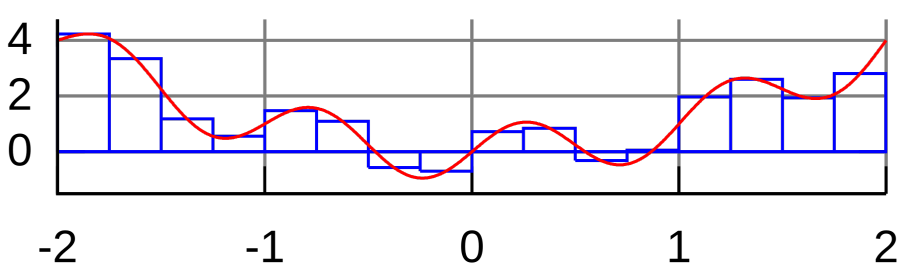
Je-li integrovaná veličina konstantní, přičítání obdélníků postačuje:

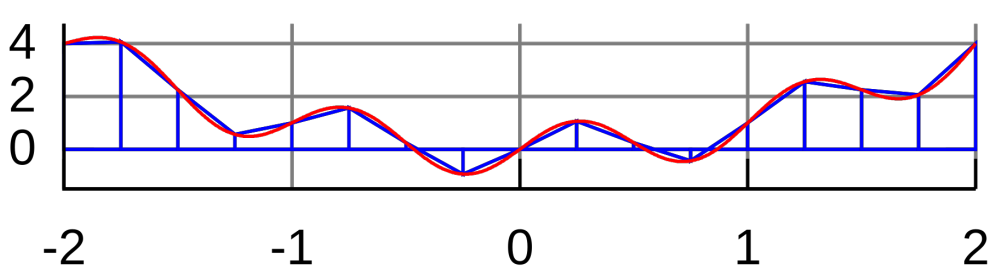
$$\vec{v}(t) = \vec{v}(t - \Delta t) + \vec{a}\Delta t$$

Mění-li se integrovaná veličina v čase, je lepší interpolovat, např. lineárně (trapezoid):

$$\vec{s}(t) = \vec{s}(t - \Delta t) + \frac{\vec{v}(t)+\vec{v}(t-\Delta t)}{2}\Delta t$$

Obdobným způsobem lze interpolovat kvadraticky atd.



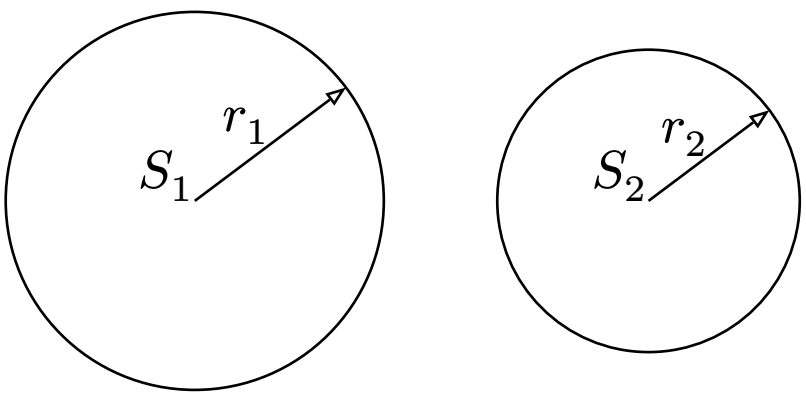


Kolize (rázy)

- Často potřebujeme vědět, zda se nějaká tělesa dotkla, nebo ne…
- Může nám stačit fakt pouhého dotyku (jedno těleso se mělo dostat do prostoru druhého)
- Někdy potřebujeme i fyzikální reakci na dotyk.
- Typický průběh kolize: detekce, resoluce
- Typická resoluce: odraz (*bounce*), skluz (*slide*), odstranění, deformace, rozbití…
- Pro různé druhy těles jsou výpočty detekcí kolizí různě složité.
  - Naivní postup: vzít každý bod z tělesa A a zkontrolovat, zda je v tělese B.
  - Analytický postup: Pro každou dvojici druhů těles použijeme jiný způsob výpočtu.
  - Zavádíme fyzikální primitiva (body, kruhy, přímky, AABB, konvexní polygony atd.)

Průnik dvou bodů/kruhů/koulí

- Kruh/koule je množina všech bodů, které jsou od středu S vzdáleny nejvýše r (poloměr).
- Bod můžeme chápat jako kruh/kouli s $r = 0$.



- Tělesa se protínají, pokud $|S_1S_2| \leq r_1 + r_2$.

Průnik AABB (axis aligned bounding box)

- AABB je množina bodů X, jejichž souřadnice (na každé ose) se nacházejí v oblasti vymezené mezemi $X_{\min}$ a $X_{\max}$.
- Tedy, platí $\bigwedge_i x_{\min_i} \leq x_i \leq x_{\max_i}$. (Lze využít pro průnik bodu s AABB.)
- Průnikem dvou AABB (A a B) je také AABB (C), meze lze vypočítat:
  - $C_{\min_i} = \max(A_{\min}, B_{\min})$
  - $C_{\max_i} = \min(A_{\max}, B_{\max})$
- Pokud neplatí $\bigwedge_i c_{\min_i} \leq c_{\max_i}$, průnik neexistuje.
- Velmi jednoduché a rychlé zpracování, využívá se skoro všude, zejména:
  - GUI
  - rychlé vyloučení průniku u složitějších objektů
- Nelze rotovat (resp. nelze jednoduše spočítat průnik dvou různě otočených AABB).

Průnik bodů/kruhů/koulí s polorovinou/poloprostorem

- Mám přímku/rovinu danou obecnou rovnicí $\vec{x}\vec{n} + d = 0$. n je normálový vektor roviny, d je druhá mocnina vzdálenosti přímky/roviny od počátku prostoru (pokud je normálový vektor normalizovaný).
- Tento koncept mohu využít k řezání prostoru (a to se dělá).
- Vzniká mi tedy polorovina/poloprostor.
- Část před rovinou $\ldots > 0$ bude prázdná, část za rovinou $\ldots < 0$ plná.
- Pokud je střed kruhu/koule mimo polorovinu/poloprostor, vyhodnotíme vzdálenost od řezné přímky/roviny.

Průniku bodů s konvexním tělesem

- Konvexní těleso mohu definovat jako průnik polorovin/poloprostorů.
  - např. *Quake* engine to tak dělá, a mnohé další.
- Detekce průniku bodu s konvexním tělesem je pouze vyhodnocení rovnice úsečky/roviny pro každou hranu/stěnu tělesa.

Průnik AABB s polorovinou/poloprostorem

- Stačí ověřit, zda libovolný z vrcholů AABB je v polorovině/poloprostoru
- To znamená 4× nebo 8× pomalejší výpočet.
- Některé hry používají jednoduchý trik:
  - Posun řezné přímky / roviny o průmět vzdálenosti mezi nejbližším vrcholem AABB a středem AABB, a úpravu na bod.
  - Toto je vhodné provádět v rámci předzpracování dat, např. *Quake* engine toto dělá.
    - *Quake* simuluje hráče jako bod v prostoru, který je patřičně „zmenšen“.
    - Zvláštní verze kolizního modelu mapy pro stojícího a skrčeného hráče.

Průnik dvou konvexních těles

- Naivní algoritmus: Nesmím najít dělící rovinu mezi tělesy. Dělící rovina prochází 2 vrcholy tělesa A a 1 vrcholem tělesa B tak, aby se obě tělesa této roviny pouze dotýkala, každé z opačné strany.
  - Velmi neefektivní.
- Další možnost: Vypočítat tzv. Minkowského součet těles A a B. Výsledné těleso C je také konvexní. Potom zjišťujeme, zda těleso C prochází počátkem prostoru.
  - Naivním způsobem neefektivní (Minkowského součet má $|V(A)||V(B)|$ vrcholů).
  - Efektivnější algoritmus *GJK* systematicky hledá nejbližší body k počátku.
- Stručně: Tady už to přestává být jednoduché.
- Konkávní tělesa je nejlepší skládat z konvexních těles.

Diskrétní versus spojitá detekce kolize

- Při detekci kolize bych ideálně měl zachytit situaci v okamžiku dotyku.
- Při simulaci s diskrétním časem dotyk s velkou pravděpodobostí netrefím.

Diskrétní detekce

- Zjišťuji v jednotlivých snímcích, zda se tělesa protínají.
- Zjistím-li kolizi, mohu mezi snímky najít přesný okamžik dotyku:
  - Lineární interpolací (použitelné u některých primitiv a přibližně lineárního pohybu)
  - Binárním půlením (přesnější, ale náročnější na výkon a obecnější)
- Při příliš rychlém pohybu / tenkém kolidéru kolizi nezaznamenám, těleso proletí skrz.

Spojitá detekce

- Tělesa extruduji po jejich dráze v čase mezi snímky.
- Hledám polohu jejich průniku.
- Spolehlivá detekce kolizí, velice náročná (z 2D je 3D, z 3D je 4D), zřídka se používá.

Ray casting

- V mnoha případech stačí kolize řešit výpočtem průniku polopřímky (*ray*) s tělesem.
- Tento průnik se počítá relativně snadno
  - a) transformací souřadnic do lokálního prostoru polopřímky (ta pak míří např. v ose z)
  - b) s využitím dělení prostoru a ořezáváním paprsku
- Využití:
  - laserové zbraně, resp. zbraně s rychlými projektily
  - kontrola volného místa v blízkosti tělesa
  - sledování povrchu pod entitou
  - zjišťování vzájemné viditelnosti dvou entit

Resoluce kolize

- Když máme polohu a čas objektů při kontaktu, řešíme další pohyb
- Absolutní odraz:
  - obě tělesa se odrazí podle směru normály
  - těleso se může roztočit, pokud náraz nemíří na těžiště
- Absolutní skluz:
  - těleso pokračuje podél stěny
  - těleso se pohybem podél stěny roztočí
- Odraz se třením:
  - vlastní rotace tělesa může ovlivnit směr odrazu
- Po odrazu musíme dopočítat zbytek cesty;
  - cestou možné řešení dalších kolizí
  - limit na počet kolizí versus potenciální zacyklení

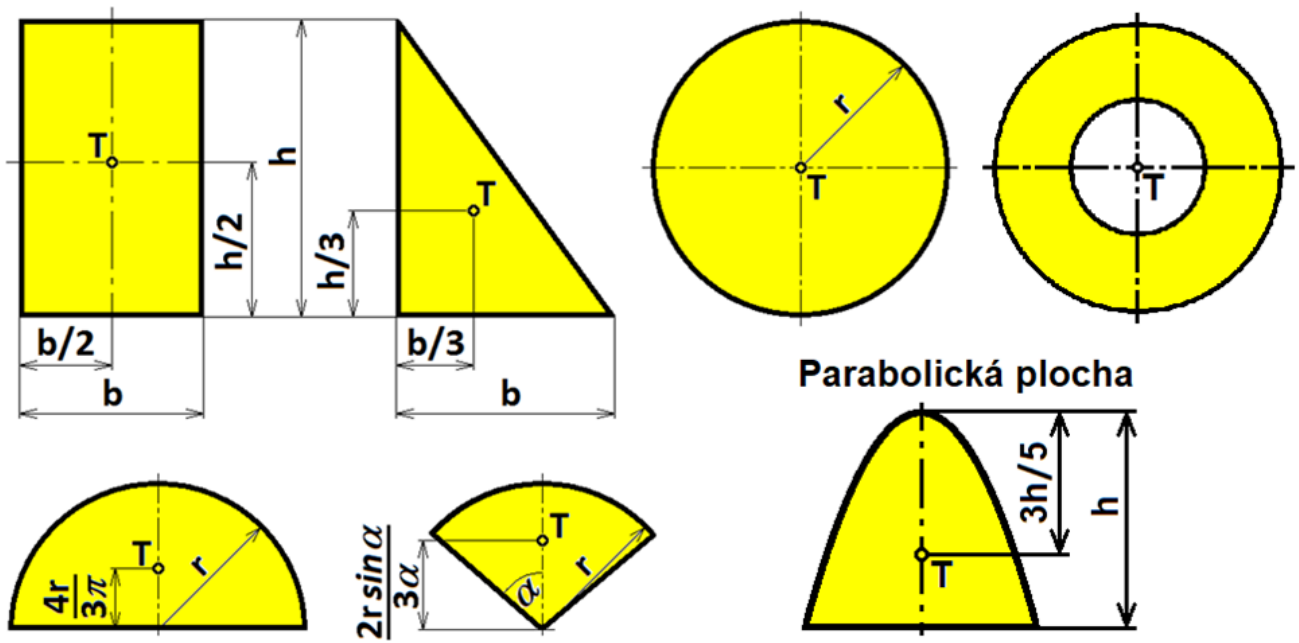
Těžiště (center of mass)

Průměrný bod, pokud integrujeme podle hmoty:

$$\overrightarrow{x_T} = \frac{\int \vec{x}\,\mathrm{d}m}{m} = \frac{\int \vec{x}\rho\,\mathrm{d}V}{m}$$

V praxi počítáme těžiště známých těles (bodů, úseček, kruhů, koulí, čtverců, krychlí, válců, trojúhelníků, čtyřstěnů…), jejichž skládáním aproximujeme libovolná tělesa.

$$\overrightarrow{x_T} = \frac{\sum_i x_i m_i}{m}$$



Pohyb po podložce

- V praxi nemá smysl opakovaně počítat pád tělesa, které leží na podložce
- Obvyklé řešení:
  - stav tělesa *na zemi* (*grounded*)
  - v tomto stavu „ignoruje“ gravitaci a vykonává pohyb podél podložky
  - místo kontroly kolize s podložkou se raycastem kontroluje opuštění podložky

Optimalizace tělesa v klidu

- Těleso, které nevykonává žádný pohyb, je obvyklé *zmrazit* (*freeze*)
- Za žádný pohyb se může považovat i extrémně pomalý pohyb (např. 1 nm/s).
- U zmraženého tělesa se nepočítají nové hodnoty veličin, zcela se přeskočí
- Těleso je obvykle odmraženo kolizí, nebo přímou změnou stavu zmražení herní logikou
- Šetří výpočetní výkon, ale může být i zdrojem bugů / podivného chování.

Fyzikální systémy

- Spojením více fyzikálních entit vzniká fyzikální systém
- Entity jsou spolu spojeny vazbami (*joints*), které zajišťují vzájemnou polohu
  - pevná vazba (*rigid joint*), kloubová vazba (*hinged joint*), kulový čep (*ball joint*), kolo na hřídeli, posuvný styk ploch atd.
  - omezení v pohybu (0 až 3 osy) a v natočení (0 až 3) se nazývají stupně volnosti (*degrees of freedom*, *DOF*)
- Známé systémy ze základní školy: páka, kladka, kladkostroj, kyvadlo
- Na strojních a stavebních středních školách: příhradové konstrukce
- Chování fyzikálních systémů můžeme popsat soustavou rovnic
  - každý stupeň volnosti přidává 1 neznámou
  - každý prvek systému přidává 1 nebo více rovnic
  - rovnice jsou obvykle lineární, ale často jsou také diferenciální

Příhradové konstrukce; obrázek: *Poly Bridge* (2016)

- Distribuce sil a momentů sil přes nosníky
- Jednotlivé nosníky jsou namáhány v tlaku nebo v tahu
- Nosník je pružina, mění délku a působí silou proti svému stlačení / natažení



Vozidla; obrázek: *Need for Speed III: Hot Pursuit* (1998)

- Kola jsou jen 4 raycasty sledující povrch silnice
- Natočení předních kol vytváří oblouk, který vůz sleduje
- Průjezd oblouku vysokou rychlostí versus setrvačnost = smyk



Kapaliny

- Dvě hlavní metody fyzikální simulace:
  - částicová – kapalina je složena z odpuzujících se částic
  - mřížková – výpočet pracuje se souhrnnými veličinami (tlak atd.)
- Alternativně existují nefyzikální simulace celulárními automaty

Textilní látky; obrázek: *Switchball* (2007)

- Jemná čtvercová mřížka pružin
- Počet elementů roste s plochou látky
- Pro látku s 10×10 elementy stovky podobných rovnic… vhodné pro GPU



Hadrový panák (*ragdoll*); obrázek: *Half-Life 2* (2004)

- Pružiny spolu s klouby jsou uspořádány do grafu (kostra modelu).
- Na jednotlivé elementy působí různé síly (gravitace, ostatní elementy a klouby, vnější síly).
- Model se deformuje podle deformace kostry.



Inverzní kinematika: *Factorio* (2016 / 2020)

- Řešíme úlohu přemístění některého z elementů na konkrétní místo
- Hledání optimální cesty ve funkci většího počtu proměnných
- Lze řešit i zafixováním části kostry, nebo rozdělením přesunu na úseky
- Možné postupy: inverze jakobiánu (netriviální), iterativní optimalizace



Fyzikální enginy

- Programovat si sám fyziku…
  - v omezeném rozsahu není problém (základní kinematika bez rotací je na několik hodin)
  - plně věrohodná fyzika ve 3D s různými fyzikálními systémy znamená roky práce
- Lepší použít existující engine:
  - Box2D, Matter.js, LiquidFun
  - Bullet, Jolt, PhysX, Avian, Box3D
  - … (a stovky dalších)
- **Vždy musíme znát a respektovat omezení zvoleného enginu!**

Glitching

- Projevy zvláštního chování způsobené nedokonalostí výpočtů.
- Čím komplexnější výpočty, tím větší rizika a závažnější projevy glitchů.

Typické *glitch(e)*:

- Průlety těles skrz
  - hrubý diskrétní čas, velká rychlost těles, tenké/chybějící kolizní modely
- Poskakování a zasekávání těles ve stěnách a v rozích
  - těleso se po kolizi odrazí do jiného tělesa, a potom už se neumí dostat ven
- Klepání, trhavé pohyby a deformace ve vzdálenějších oblastech prostředí
  - nepřesnosti způsobené číselnou reprezentací
  - běžně používaný 32bitový float má přesnost 1 mm ve vzdálenosti 16 km od počátku
- Narůstající kmity, prudké výstřely/výskoky těles
  - možný výsledek dělení potenciálně velmi nízkými čísly
  - nestabilní fyzikální systémy s kladnou zpětnou vazbou / bez tlumení

Dopady *glitch(ů)* na hru

- Některé jsou zábavné, nebo je lze lehce vyřešit hráčovou akcí
- Ztráty entit (postav, předmětů…) kritických pro hru – entity nutno hlídat

Herní fyzika versus…

…středoškolská fyzika:

- Tření je na střední škole omezeno na pár příkladů, ale ve hrách mají systémy bez tření tendenci oscilovat.
- Odpor vzduchu existuje. A závisí na hustotě vzduchu.
- Existují pružné materiály.
- Tělesa nejsou homogenní.

…reálný svět:

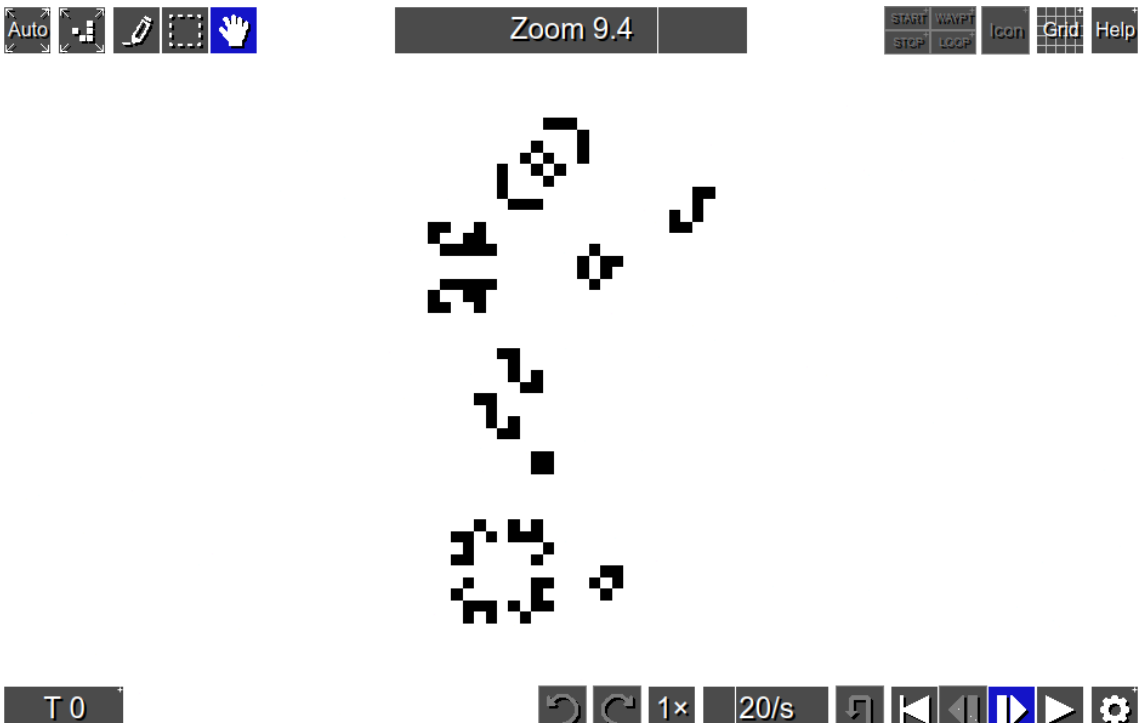
- Tělesa nejsou dokonale tuhá a při kolizích se deformují.
- Neexistuje nic jako statické těleso. I skála se při nárazu předmětu rozvibruje.
- Rychlost světla překročit nelze.

Celulární automaty

Základní myšlenka

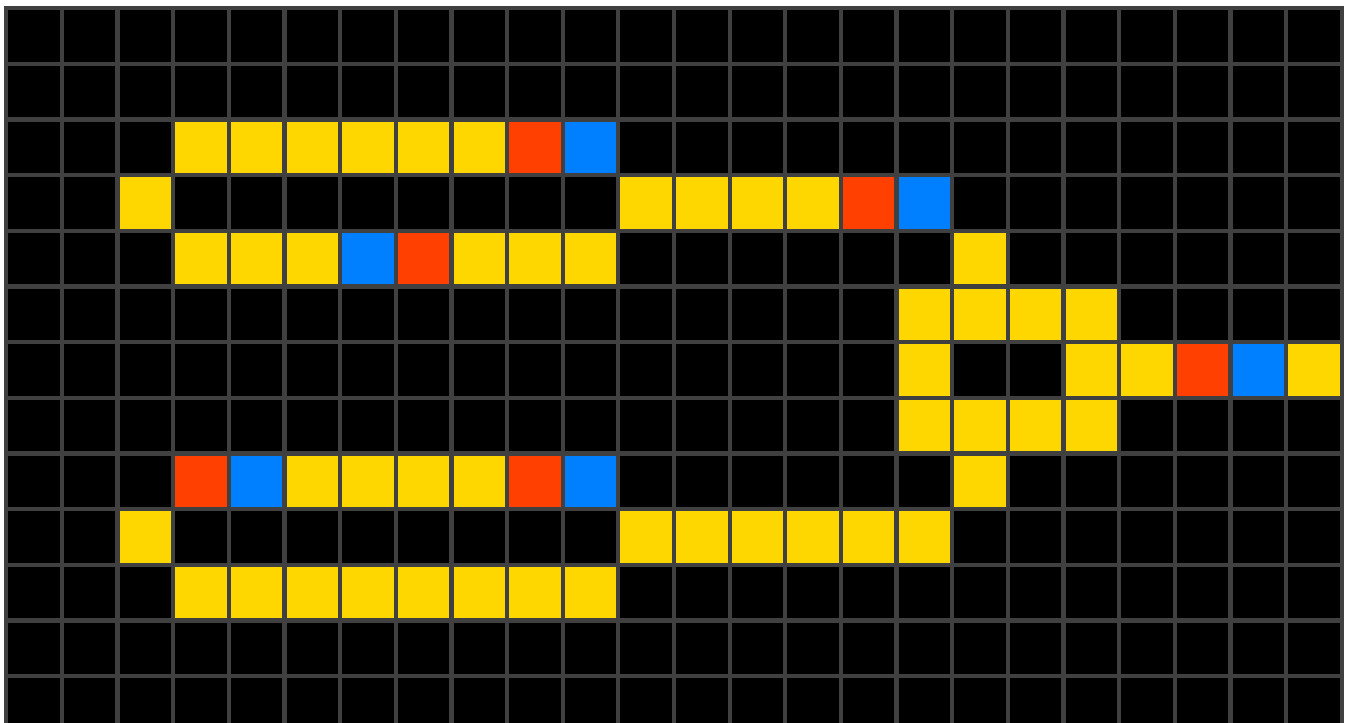
- Stav entit je poskládán z buňek, které jsou vzájemně propojené, typicky čtvercová síť
- V každé generaci se počítá nový stav každé buňky z původního stavu jí a jejích sousedů
- Lze snadno paralelizovat nebo převést na GPU
- I s jednoduchými pravidly lze dosáhnout komplexního chování

Příklad: *Conway's Game of Life* (1970)



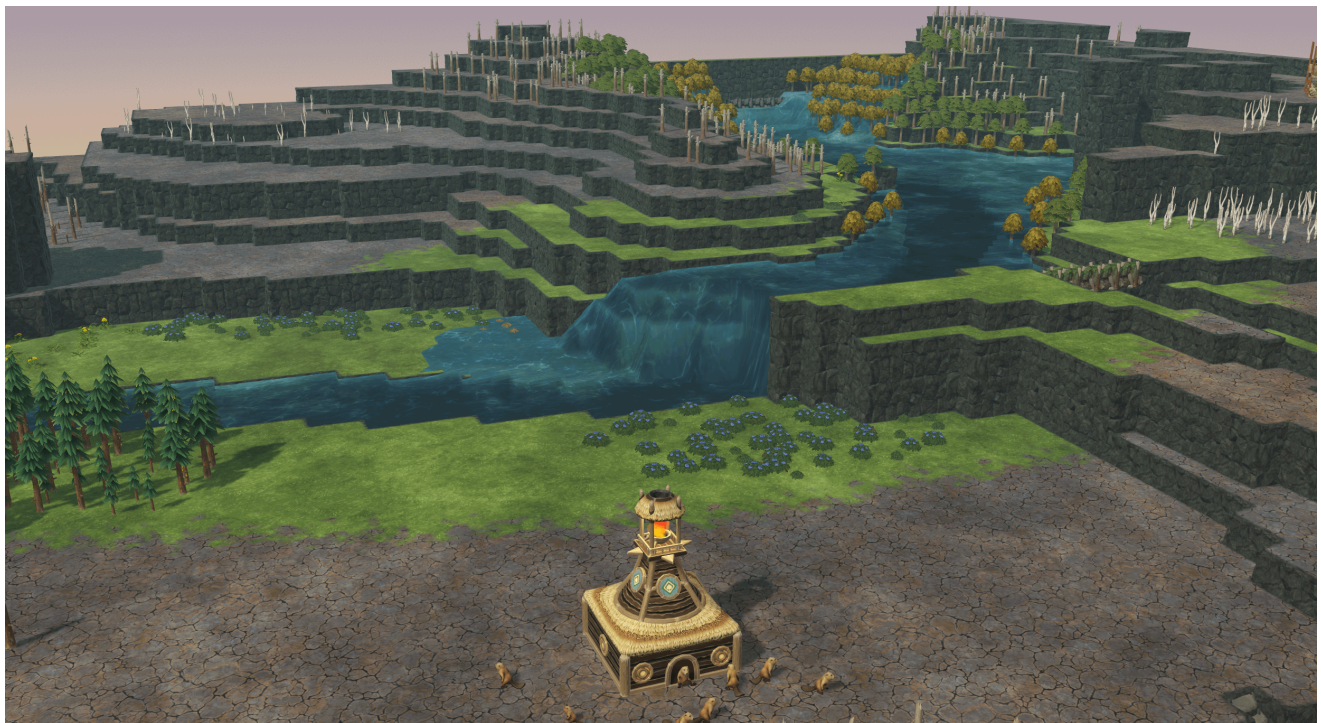
- Živá buňka (černá) žije, pokud měla dva nebo tři živé sousedy. Mrtvá buňka s právě třemi živými sousedy ožije.

Příklad: *Wireworld* (1987)



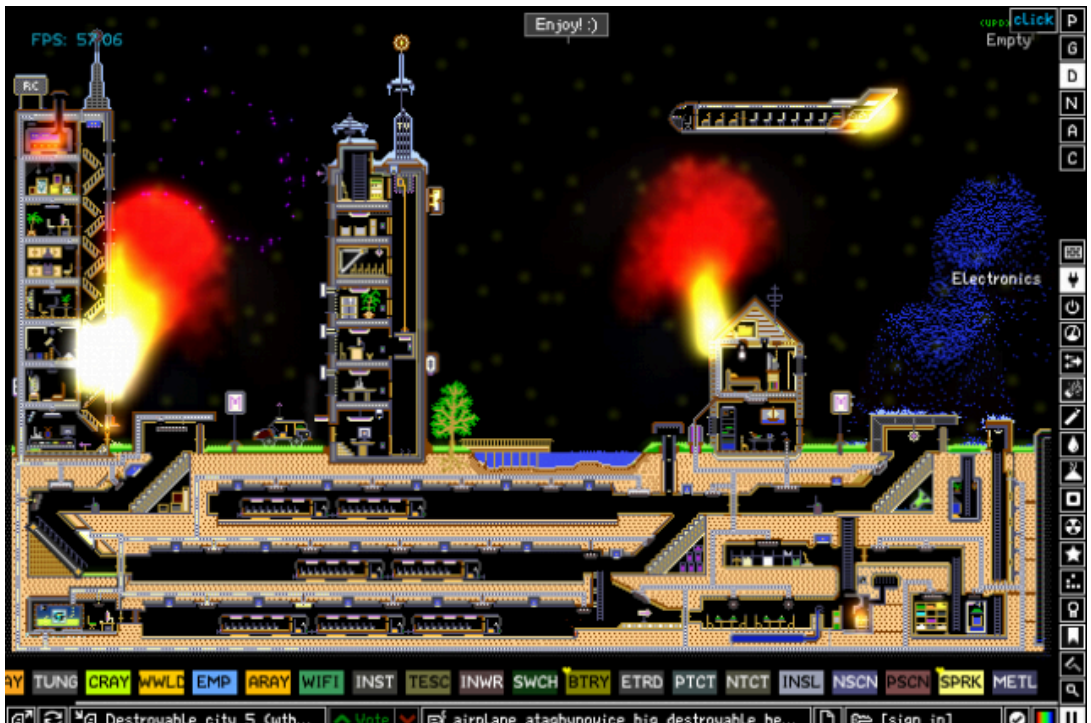
- Ze zadní části elektronu se stane vodič, z elektronu se stane zadní část. Z vodiče v sousedství elektronu se stane elektron.

Příklad: rozlévání vody ve spoustě strategií, zde *Timberborn* (2021)



- Pole má úroveň hladiny vody podle sousedů (průměr, případně se započítává i rozdíl hladin). Existují zdroje vody a voda vytéká z mapy.

Příklad: *Powder Toy*



- Interakce jednotlivých pixelů napodobují typické reakce ve fyzice a chemii (ale nejsou přesné); každý pixel má teplotu a tlak.