

Simulování života mikroorganismů v modelovém mikrosvětě

Filip Majer

Vedoucí práce: Emil Miler

Gymnázium Jana Keplera

<https://github.com/nat-int/mikrosim>

Obsah

1. Úvod	2
1.1. Cíle	2
1.2. Související práce	2
2. Model	2
2.1. Prostředí	2
2.2. Proteiny	3
2.2.1. Aktivní místa	4
2.2.2. Allosterická místa	5
2.2.3. Reakce	5
2.2.4. Genom	6
2.3. Buňky	7
2.3.1. Struktury	8
3. Implementace	9
3.1. Použité technologie	9
3.2. Struktura kódu	10
3.3. Implementační detaily	10
4. Návod k použití	11
4.1. Kompilace	11
4.2. Návod k aplikaci	12
4.2.1. okno „Controls“	12
4.2.2. okno „mikrosim“	13
4.2.3. okno „effect blocks“	14
4.2.4. okno „extra info“	14
4.2.5. okno „cell list“	15
4.2.6. okno „protein view“	15
4.2.7. okno „cell view“	16
4.2.8. Zobrazení simulace	17
5. Závěr	18
6. Seznam odkazů	18

1. Úvod

Život je plný komplexních systémů, ale pro jeho zkoumání je někdy užitečné mít zjednodušený model, který jsme schopni plně propočítat. Naše schopnosti počítání posunuly vpřed počítače, kterým dnes už můžeme zadávat k propočítání i poměrně komplikované modely.

1.1. Cíle

Cílem tohoto projektu je vytvořit grafickou aplikaci, která zobrazuje a simuluje život modelových mikroorganismů. Model je zvolen tak, aby nebyl moc daleko od středoškolského modelu, jak mikroorganismy fungují. Zároveň je dostatečně jednoduchý, aby pomocí něj šlo zároveň simulovat více variant organismů i na běžně výkonných počítačích.

1.2. Související práce

Existuje mnoho modelů i jejich simulátorů zjednodušeného života. Velká část z nich funguje tak, že se nadefinuje několik typů buněk, kde má každá své specifické chování. Z těchto modelových buněk se poskládá organismus. Ten se pak může replikovat do stejné nebo podobné buněčné konfigurace. Jedním takovým simulátorem je Biogenesis (<https://biogenesis.sourceforge.net/>).

Jiná velká skupina modelů simuluje organismy jako objekty, které podle definovaných pravidel interagují s okolím a podle vstupů provádějí různé akce. Vstupy se typicky zpracovávají s využitím neuronových sítí. Příkladem pro tuto skupinu je The bibites (<https://thebibites.itch.io/the-bibites>).

Oba tyto přístupy tak modelují spíše mnohobuněčné organismy než jednobuněčné mikroorganismy.

O něco obecnější model má projekt ALIEN (<https://alien-project.org/>), který je založený na simulaci částic, které mohou tvořit vazby a předávat si signály. ALIEN se snaží o jednoduchý model a z něj vyvstávající chování, a tak v něm život není tak blízko realitě, jako bychom chtěli. Kromě toho je napsaný v CUDA (<https://developer.nvidia.com/cuda-toolkit>), takže vyžaduje počítač s grafickou kartou od společnosti Nvidia.

2. Model

Model je ve dvou dimenzích, aby bylo snadnější jej implementovat a přehledně vykreslovat.

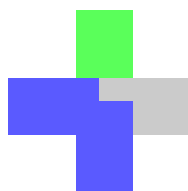
2.1. Prostředí

Mikroorganismy žijí v definovaném prostředí. Jako základ prostředí této simulace byla zvolena tekutina, protože i skutečné mikroorganismy často žijí ve vodě. Standardní způsoby simulování tekutin jsou částicové, eulerovské (kontinuální) a kombinované. V tomto prezentovaném modelu byl použit částicový model z tohoto simulátoru: <https://github.com/SebLague/Fluid-Sim/tree/Episode-01>, především proto, že daný model je snadno implementovatelný, zároveň je pro částicové simulace snažší zabránit tvorbě hmoty z ničeho. Do prezentovaného modelu byl tento částicový model převzat bez korekcí shlukování částic při rychlých ztrátách tlaku, protože chování tekutiny příliš neovlivní a jejich vynecháním se model zjednoduší. Snadnost implementace tu je na úkor realismu, tekutina je zde svými vlastnostmi mezi plynem, kapalinou a gelem. Přesto je dostatečně blízko reálné tekutině.

2.1 Model — Prostředí

Tento model ve zkratce funguje tak, že tekutinu rozdělí na částice, kde každá z nich představuje malý objem tekutiny. Částice se zrychlují takovým směrem, aby se hustota (a tím i tlak) blížil požadované hodnotě. Mimo to se rychlosti předávají mezi blízkými částicemi pro efekt viskozity.

Kromě tekutiny jsou v typickém prostředí, kde jednobuněčné mikroorganismy žijí, různé látky, ze kterých získávají energii nebo stavební materiály. Aby se simulace vešla do paměti a zároveň lépe paralelizovala, jsou zde látky modelovány zjednodušeně jako čtveřice podčástí, které jsou červené, zelené, modré a šedé. Látky jsou symetrické vzhledem k rotaci (když se podčásti protočí, tak je to stále stejná látka). Lze je zobrazit jako čtyřcípou hvězdu s cípy obarvenými na jednu ze čtyř barev.



Obrázek 1: „látka“ složená ze zelené, šedé, modré a modré podčástí

Chemické chování látek (tvorba vazeb a vazebná energie) je inspirováno chováním malé organické látky, která obsahuje jeden uhlík se čtyřmi skupinami. Červená podčást se vlastnostmi blíží karboxylové skupině, zelená hydroxylové, modrá aminové skupině a šedá neutrálnímu vodíku. Každé látce je přiřazena energie (slučovací entalpie) (uložená v jejich vazbách), která je určena takto:

$$\sum_i^{\text{podčásti}} E_{z(i)} + c(i) \cdot (f(\text{vedle } i \text{ na jedné straně}) + f(\text{vedle } i \text{ na druhé straně}) + \varphi(\text{naproti } i))$$

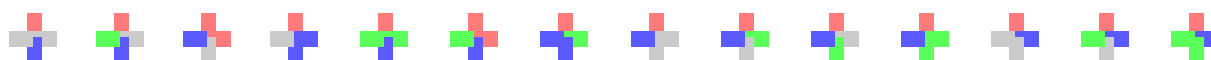
E_z dává energii vazby mezi podčástmi a středem látky. f určuje sílu indukčního efektu mezi vedlejšími pozicemi, φ mezi protějšími pozicemi. c dává sílu vlivu indukčního efektu z ostatních podčástí na podčást. Hodnoty energií látek jsou tak rozmanité a jen občas shodné (ale jsou shodné pro překlopené látky) a můžeme získávat zajímavé výsledky při počítání s jejich rozdíly.

Každá částice (každá kapka tekutiny) v sobě může mít libovolné množství každé látky. V průběhu simulace látky difundují. Při rozptylování dané látky se koncentrace látky z každé částice rovnoměrně rozmístí do všech částic v okruhu se stejným poloměrem, jaký je dosah jejich silových interakcí.

2.2. Proteiny

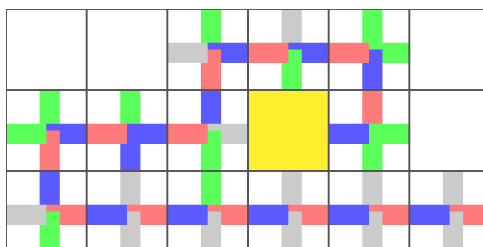
Nejdůležitějšími funkčními jednotkami v buňkách v tomto modelu jsou proteiny. Ty se skládají z aminokyselin, které se zřetězí a složí do tvaru, který jim umožní pracovat. Skládání proteinů je na tuto simulaci moc složité, a proto je zde model opět hodně zjednodušený. V tomto modelu jsou aminokyseliny definovány jako látky vybrané z těch, které obsahují červenou a modrou podčást:

2.2 Model — Proteiny



Obrázek 2: látky vybrané jako proteinogení

Ze seznamu těchto látek pak skládáme protein tak, že za červený konec předchozí látky přidáme další látku tak, aby navazovala modrým koncem (jako translace probíhá od N-terminu k C-terminu za tvorby peptidických vazeb). Protože jsou látky pravoúhlé, skládají se jejich lineární řetězce do mřížky, která simuluje jejich prostorovou strukturu. Když existuje více možností, jak by se mohla další látka navázat, vybere se ta, která v řetězci červených a modrých podčástí pokračuje co nejdéle rovně.




Obrázek 3: ukázka proteinu (aa19c6a80100e (více v sekci genom)), žluté pole značí aktivní místo


Ze struktury je určena stabilita proteinu, což je podíl proteinů, která se nerozloží během jednoho kroku simulace:

$$\min(1 - 15\% \cdot e^{-s_c}, 99\%), s_c = \frac{n}{80} + \frac{n_e}{4}$$

kde n je počet polí v mřížce, kde je více látek najednou (protein překrývá sám se sebou - kříží) a n_e je počet míst, kde je více látek najednou a zároveň je vedle prázdného pole. Pokud se řetězec proteinu překříží, znamená to, že je zabalenější a je náročnější ho rozkládat, takže je stabilnější. Když je překryv vedle prázdného pole, musí řetězec postupovat stejnou cestou, struktura se zpevňuje a je o to stabilnější.

2.2.1. Aktivní místa

V modelových proteinech jsou aktivní místa, kde dochází k výměnám podskupin. Jako aktivní místa jsou definována prázdná pole, která mají na všech čtyřech sousedních polích právě jednu látku. Do takového místa se může vázat látka, která je doplňková k podčástem, které do aktivního místa směřují. K červené je doplňková modrá a naopak. Zelené a šedé podčásti jsou doplňkové samy sobě. Obrázek 3 obsahuje vyznačené místo, do kterého by se vážala látka . Aktivní místa mohou být maximálně 4 (preferují se ta vlevo a poté nahoře).

V tomto modelu se aktivní místa, která jsou maximálně 4 pole od sebe (opět přednostně zleva a shora), spárují a katalyzují reakci výměny podčástí mezi látkami. Vyměňují se ty podčásti, které jsou k sobě nejbližší. Zbývá aktivní místa katalyzují reakci výměny vedlejších podčástí vázané látky, tedy těch, které jsou nejbližší středu (těžišti) aktivních míst (přednostně nalevo nahoře). Pro Obrázek 3 bude výsledek reakce .

2.2 Model – Proteiny

Takto vytvořený model reakce umožňuje změnu podčástí podle struktury proteinu. Při mutaci může být reakce, kterou protein provádí, zachována, mírně pozměněna i radikálně změněna s nezanedbatelnými pravděpodobnostmi. Takovýto model je snadné implementovat a je nenáročný na výpočet. Dokud nenastanou krajní případy, nezáleží na orientaci proteinu. (existuje ale významný krajní případ - jen jedno aktivní místo, pak na preferencích záleží).

2.2.2. Allosterická místa

Proteinům by neměla chybět možnost chovat se různě podle látek v jejich prostředí (aby buňky mohly měnit chování podle podnětů). Proto je součástí modelu allosterická modulace, která umožňuje proteinům obsahovat receptor a měnit podle něj svou efektivitu. Allosterická místa jsou v tomto modelu podobná aktivním - jsou to prázdná pole, která vedle sebe mají právě tři pole, která obsahují právě jednu látku. Do takového místa se mohou vázat 4 různé látky, které mají tři podčásti doplňkové k těm v proteinu. Modulační efekt je určen funkcí $\sin(\text{vzdálenost allosterického od nejbližšího aktivního místa})$. Sinus byl vybrán pro svůj průběh, nemění se moc rychle, ale mění se, a aby se při posouvání allosterického místa postupně měnil modulační efekt.

Podle modulačního efektu se určuje celkový efekt allosterických modulátorů na protein:

$$M = \max\left(0, 1 + \left(\sum_i^{\text{modulátory}} \max(f_i(c_m), 0)\right) + \left(\min_i^{\text{modulátory}} \min(f_i(c_m), 0)\right)\right)$$

(součet $f_i(c_m)$, ale započítaná je jen nejnižší hodnota). kde c_m je koncentrace modulující látky v částici kde protein působí. f_i je funkce toho, jak látka ovlivňuje průběh reakce proteinu podle své koncentrace. podle modulačního efektu (m_e) se f_i přelévá mezi

$$f_{\text{aktivační}}(c) = c^2 \quad (\text{pro } m_e = 1)$$

$$f_{\text{potřebný}}(c) = c^2 - 1 \quad (\text{pro } m_e = 0)$$

$$f_{\text{inhibiční}}(c) = -c \quad (\text{pro } m_e = -1)$$

takto:

$$f_i(c) = \text{lerp}(f_{\text{potřebný}}, f_{\text{aktivační}}, m_e) \quad \text{pro } m_e > 0$$

$$f_i(c) = \text{lerp}(f_{\text{potřebný}}, f_{\text{inhibiční}}, \text{smoothstep}(-m_e)) \quad \text{pro } m_e \leq 0$$

Pro allosterické aktivátory je přechod mezi funkcemi lineární, pro allosterické inhibitory je přidán smoothstep k interpolačnímu koeficientu, aby se urychlil přechod přes bod, kde f_i vychází přibližně jako záporná konstanta.

2.2.3. Reakce

V tomto modelu mohou být uvnitř částic proteiny, které katalyzují reakce. Předpokládá se, že bez pomoci proteinu běží reakce zanedbatelně.

2.2 Model – Proteiny

U reakcí jsou známy reaktanty a produkty, takže lze získat i změnu entalpie:

$$\Delta H = \sum_i^{\text{produkty}} \Delta H(i) - \sum_i^{\text{reaktanty}} \Delta H(i)$$

kde ΔH je již zmíněná slučovací entalpie látky. Pokud se zanedbá $\Delta S \cdot T$ (předpokladem pro to je, že se entropie během reakcí příliš nezmění), lze z

$$\Delta G = \Delta H - \Delta S \cdot T \rightarrow \Delta G = \Delta H$$

$$K = e^{-\frac{\Delta G}{RT}} \rightarrow K = e^{-\frac{\Delta H}{RT}}$$

spočítat rovnovážnou konstantu reakce. K té by se měl blížit poměr součinů koncentrací produktů a reaktantů, s rychlostí určenou účinností enzymatické katalýzy. Během kroku, kdy protein provádí danou reakci, se spočítá aktuální poměr (K'') a během kroku se změní na

$$K' = \text{lerp}(K'', K, c_r \cdot M \cdot c_p)$$



kde c_r je konstanta rychlosti všech reakcí, M je efekt modulátorů a c_p je koncentrace proteinu. Nechť je δ změna koncentrací v tomto kroku. Pak

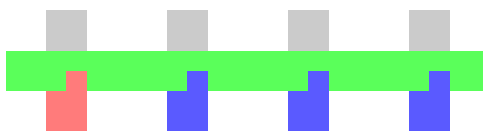
$$K' = \frac{\prod_i^{\text{produkty}} (c_{pi} + \delta)}{\prod_i^{\text{reaktanty}} (c_{ri} - \delta)}$$

$$K' \cdot \left(\prod_i^{\text{reaktanty}} (c_{ri} - \delta) \right) - \prod_i^{\text{produkty}} (c_{pi} + \delta) = 0$$

což je po roznásobení rovnice maximálně čtvrtého stupně v δ (protože maximum jsou 4 aktivní místa) a tak je řešitelná. Z řešení se vybere takové δ , které je v absolutní hodnotě nejmenší a zároveň nedostane žádnou koncentraci do záporné hodnoty. Pokud by měla přesáhnout do záporu, δ se zmenší.

2.2.4. Genom

Jako návod pro sestavení proteinu slouží genom. Ten je tvořen z látek  a , které jsou přes zelené podčásti spojené do řady (v tomto modelu jsou jen 2 báze, červená a modrá):



Obrázek 4: ukázka úseku genetického kódu (FTTT/8)

tento genetický kód začíná nalevo. Protože je aminokyselin více, než bází, genetický kód se dělí na kodony, které tu mají délku 4, aby mohly kódovat všech 14 zdejších aminokyselin. Genetický kód se přepisuje pomocí F a T, kde F značí červenou bázi a T modrou. Pro přehlednější zápis je i zápis po kodonech. Číslo v hexadecimálním zápisu, tedy číslice nebo

malé písmeno z rozsahu a-f, značí jeden kodon. Každé číslo převedeme na kodon tak, že do genetického kódu přidáme F v případě lichého a T v případě sudého čísla. Poté číslo vydělíme dvěma a tento postup ještě třikrát zopakujeme. Přiřazení aminokyselin ke kodonům ukazuje okno „extra info“ v aplikaci (v dokumentaci je i jeho obrázek).

Navázaný transkripční faktor blokuje v daném místě transkripci, z důvodu snadné implementace působí v místě nejbližší začátku genomu.

Pokud na okraji také obsahuje řadu alespoň tři zelených podčástí (ve stejném smyslu okraje jako v předchozích odstavcích), je transkripční faktor pozitivní a od daného místa transkripci spouští. Zelená část okraje slouží jako místo, na které nasedá ekvivalent RNA transkriptázy.

Modelové buňky obsahují centrum, což je speciální typ částice. Každá buňka má svůj genom, který kóduje několik proteinů. Začátek místa transkripce v genomu značí 6 po sobě jdoucích červených bází (počítá se od konce úseku) (nebo pozitivní transkripční faktor). Od tohoto místa se genom čte po kodonech. Když se narazí na kodon a, který značí start kodon, začnou se do proteinu zapojovat další látky podle kodonů, až do kodonu e (stop kodonu) včetně.

Každá buňka může obsahovat libovolné množství každého proteinu zakódovaného ve svém genomu, podobně jako to je u látek. Během každého kroku se proteiny mohou tvořit a rozpadat. Rozpadne se podíl proteinů podle již zmíněné stability. Vytvořit se může $\frac{1}{2}$ jednotek množství proteinů za krok, $M' \cdot c_p'$, pokud je tvorba daného proteinu podmíněna pozitivním transkripčním faktorem. M' je efekt allosterických modulatorů na transkripční faktor a c_p' je jeho koncentrace. Toto množství se sníží o součet $M' \cdot c_p'$ všech negativních transkripčních faktorů na celém úseku, který protein kóduje. Toto množství se sníží v případě, že buňka neobsahuje dostatečné množství látek na tvorbu proteinu.

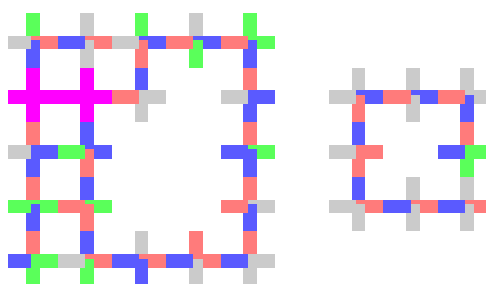
Modelové buňky potřebují k napodobení života ještě několik funkcí, které nespadají pod katalyzátory (enzymy) a transkripční faktory. Jednou z nich je dělení, při kterém se musí nakopírovat celý genom. Genom zde kopírují „genomové polymerázy“, což jsou proteiny, které mají nanejdvůh čtyři pole od sebe aktivní místa pro obě látky, ze kterých se tvoří genom (a

✚) (takže „pracují s genomem“). Tato aktivní místa se nepodílejí na reakcích. Kromě toho musí provádět nějakou jinou reakci, ze které získají energii pro tvorbu nového genomu. Vždy, když tuto reakci provede, namnoží se kousek genomu (nebo dojde k mutaci). Kopírování probíhá od začátku, postupně. Za každý kousek, k nakopírování je malá šance, že se přeskočí báze nebo kodon, že se přidá náhodná báze nebo kodon a nebo, nebo že místo, odkud se právě kopíruje, skočí na náhodné místo v celém genomu. Při přidávání bází určuje šance na vybrání báze jejich koncentrace (koncentrace látky je váha při náhodném výběru). Při normálním vkládání další báze se také naváže náhodná báze, ale s velkou pravděpodobností se odváže, pokud je špatně. Když se odváže, nastane další pokus.

Buňky v modelu mají body zdraví. S každým krokem simulace 2 body ztratí, což představuje poškozující vliv prostředí - genom se může rozpadat nebo poškozovat a je potřeba ho udržovat. Údržbu provádí opravovací proteiny, které mají alespoň 4 pole dlouhý úsek červených a modrých konců (jako mají transkripční faktory, ale delší) a aktivní místo na některou z látek, ze kterých je tvořen genom (✚ nebo ✚). Toto aktivní místo se pak nepodílí v reakcích. Když tento protein provede jinou reakci, ze které se uvolní energie, dostane buňka, ve které protein je, určité množství bodů zdraví (podle uvolněné energie).

2.3.1. Struktury

Kromě centra může modelová buňka obsahovat jiné struktury. První z nich je „velká struktura“. Tou je další částice, která je k centru vázána „pružinou“, představující cytoskelet. Velká struktura se sestavuje z „proteinů velké struktury“, které mají tvar čtverce velikosti 5x5 polí.

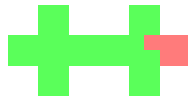


Obrázek 6: proteiny struktury, nalevo velké, napravo malé (růžová značí překryv látek v jednom poli)

Tyto proteiny neprovádějí žádné reakce a nemohou být modulovány. Když se v buňce vyrobí dostatek proteinů velké struktury, částice se vytvoří, ale může být pouze jedna na buňku. Proteiny mohou být označeny tím, že jejich kód končí 3e. Označené proteiny působí v částici struktury místo centra buňky. Značka se po vytvoření proteinu oddělí, takže se protein chová tak, jako kdyby značku neměl.

Na velkou strukturu může buňka působit motory. To jsou proteiny, které jsou velké alespoň 5x5 polí a jsou bodově symetrické podle jejich středu. Zároveň musí provádět reakci, ze které získají energii. Podle energie získané z reakce se zapůsobí silou na velkou strukturu ve směru kolem centra buňky. Pokud je protein označený, tak po směru hodinových ručiček, jinak proti směru hodinových ručiček.

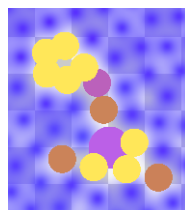
Druhý typ struktury jsou strukturní částice. Ty vyrábí slučovač částic, což je protein, který obsahuje dvě modulační místa vedle sebe, do kterých zapadá tato látka:



Obrázek 7: kombinace látek tvořící strukturní částice

Tato modulační místa pak nefungují jako modulační místa. Při provedení reakce uvolňující energii se sestaví malé množství těchto větších látek (podle energie a množství látek, které v nich jsou). Poté, co se jich sestaví dostatečné množství, přidá se strukturní částice. Když je slučovač označený, přidá se strukturní částice do řetízku za částici velké struktury. Když ne, přidá se do membrány (o té bude více v následujícím odstavci).

Třetí typ struktury je „malá struktura“. Malé struktury jsou částice, které jsou také k centru vázány pružinou, ale také se vážou navzájem a rozestavují se kolem centra. Podobně jako velké struktury tvoří proteiny malé struktury, které mají tvar čtverce o velikosti 3x3 pole. Malé struktury slouží jako základ pro membránu. Když se do membrány přidá strukturní částice, zapojí se do náhodného místa v cyklu malých struktur. Strukturní částice se může do membrány přidat, pokud má buňka alespoň 3 částice malé struktury.



Obrázek 8: buňka se strukturou (barvy jsou popsány v části Zobrazení simulace)

Buňka může mít maximálně 4 částice velké a malé struktury dohromady a maximálně 8 strukturních částic.

Když má buňka alespoň 4 strukturní částice v uzavřené (neroztrhlé) membráně, ztrácí body zdraví poloviční rychlostí, protože ji membrána chrání.

3. Implementace

3.1. Použité technologie

Pro implementaci byl zvolen jazyk C++, který umožňuje velkou míru abstrakce společně s velkou kontrolou nad průběhem programu. Na sestavování projektu byl vybrán CMake, který má dobrou podporu na více platformách i pro většinu rozšířených knihoven.

Pro vykreslování a část simulování byl vybrán Vulkan, což je API pro práci s grafickými jednotkami podporující značnou část platform i čipů. Skrze něj je možné grafické čipy nebo karty ovládat podrobně, takže je možné jejich výpočetní potenciál efektivně využívat. Vulkan je náročnější používat přímo. Pro ulehčení práce byla použita knihovna Vulkan Memory Allocator a základ projektu tvořila knihovna na vykreslování s Vulkanem, kterou jsem si před

3.1 Implementace — Použité technologie

několika lety napsal, obsahující sadu abstrakcí k Vulkanu. Na implementaci shaderů byl použit jazyk GLSL.

Na multiplatformní vytvoření okna aplikace byla použita knihovna GLFW pro její jednoduchost.

Na výpočty byly využity knihovny glm, a komponenta math z knihovny boost. Glm pomáhá lépe zapisovat efektivní výpočty z lineární algebry. Z boost math byly využity funkce na řešení kvadratických, kubických a kvartických rovnic, které vyvstávají při výpočtech reakcí.

Na zobrazování informací a interakci s uživatelem byla vybrána knihovna ImGui (<https://github.com/ocornut/imgui>), kterou je především snadné používat.

3.2. Struktura kódu

V podsložkách `format`, `input`, `log`, `rendering` a `scripts` a souborech `util.cpp` a `util.hpp` se nachází již zmíněná (mírně upravená) knihovna s abstrakcemi k Vulkanu.

V podsložce `shaders` jsou zdrojové kódy shaderů a většina projektu je přímo ve složce `src`.

Vstup do programu se nachází v `main.cpp`, kde se jen nastaví náhoda a spustí aplikace z `app.cpp`. V těchto souborech je především vykreslování a zpracovávání vstupů uživatele.

V `cell.cpp` je chování buňky a proteinů během simulace (bez interakce s jinými částicemi), v `protein.cpp` je statická analýza proteinů. `compounds.cpp` obsahuje několik funkcí pro práci s látkami.

`parse.cpp` definuje funkce pro načítání hodnot ze souborů.

V `particles.cpp` je kód pro řízení interakcí mezi částicemi.

`views.cpp` definuje složitější podokna („protein view“ a „cell view“).

3.3. Implementační detaily

Aby byla simulace numericky stabilní, je změna v čase mezi kroky simulace konstantní.

Pozice a interakce částic se počítají z většiny na grafické jednotce, aby jich mohlo být co nejvíce (interakce se lépe paralelizují). Procesor simuluje akce buněk a proteinů, které obsahují netriviální výpočty.

Algoritmus pro hledání blízkých částic byl použit tento: https://ramakarl.com/pdfs/2014_Hoetzlein_FastFixedRadius_Neighbors.pdf. Pro počítání prefixových součtů byl využit algoritmus z Nvidia GPU Gems 3 kapitoly 39 (<https://developer.nvidia.com/gpugems/gpugems3/part-vi-gpu-computing/chapter-39-parallel-prefix-sum-scan-cuda>) bez oprav „bank“ konfliktů, protože jsou tyto problémy a opravy specifické pro Nvidia hardware. Místo CUDA jsou tyto techniky implementovány v compute shaderech.

Rozpouštění látek probíhá tak, že se do paměti grafické jednotky nakopírují koncentrace 4 látek (4 jsou protože jsou grafické jednotky obvykle dělané na počítání se čtveřicemi čísel) a během počítání interakcí částic se provede i rozpouštění. Při následujícím kroku simulace se

3.3 Implementace — Implementační detaily

koncentrace látek po rozpuštění zkopírují zpět do paměti procesoru a nahrají se koncentrace následujících 4 látek. Takto se rozpouštěné látky postupně střídají.

Aby se urychlila práce na procesoru v případech, kdy buňky obsahují mnoho různých proteinů, zhodnotí se každým krokem jen jeden protein a buď vytváření, nebo aktivita proteinu. Proteiny se postupně procházejí po krocích a koeficienty jsou upravené tak, aby se započítaly změny za kroky, kdy se počítají jiné proteiny.

Do koncentrací látek, které se právě rozpouští nebo kopírují zpět po rozpouštění, nelze právě zapisovat, protože by se přepsaly novými hodnotami (výsledky rozpouštění). Když lze zapisovat do všech „aminokyselin“ (přibližně v $\frac{1}{3}$ případů), vyhodnocuje se vytváření proteinů, jinak se počítají jejich aktivity. Pokud by měl nějaký protein měnit koncentraci látek, které se zrovna rozpouštějí, jeho vyhodnocení se vynechá.

4. Návod k použití

4.1. Kompilace

Na sestavení projektu je potřeba mít nainstalované následující programy a knihovny (knihovny se hledají pomocí `find_package` v CMake):

- kompilátor c++, který (dostatečně) podporuje c++23, například g++ 14 nebo novější nebo clang++ 18 nebo novější
- CMake 3.22 nebo novější (<https://cmake.org/download/>)
- sestavovací systém, který je podporovaný cmake, například GNU make
- knihovny pro vulkan (včetně vulkan-hpp verze 1.3.301 nebo novější), typicky z LunarG Vulkan SDK (<https://vulkan.lunarg.com/>), ale pro linux bývají v repozitářích
- knihovnu vulkan memory allocator (<https://github.com/GPUOpen-LibrariesAndSDKs/VulkanMemoryAllocator>)
- knihovnu glfw3 (<https://www.glfw.org/>)
- knihovnu glm (<https://github.com/g-truc/glm>)
- knihovnu boost 1.81 nebo novější (stačí komponenta math) (<https://www.boost.org/users/download/>)

Vše lze dohromady systému Ubuntu nainstalovat následujícím bash skriptem, ale pravděpodobně je v repozitářích několik z těchto věcí v moc starých verzích a tak je bude potřeba nainstalovat jinak:

```
sudo apt install build-essential cmake
sudo apt install vulkan-tools libvulkan-dev spirv-tools
sudo apt install libglfw3-dev libglm-dev libboost-dev

git clone https://github.com/GPUOpen-LibrariesAndSDKs/VulkanMemoryAllocator.git
cd VulkanMemoryAllocator
cmake -S. -Bbuild -DCMAKE_BUILD_TYPE=Release
cmake --build build
sudo cmake --build build -t install
```

Tento projekt pak jde naklonovat a zkompileovat následujícím bash skriptem:

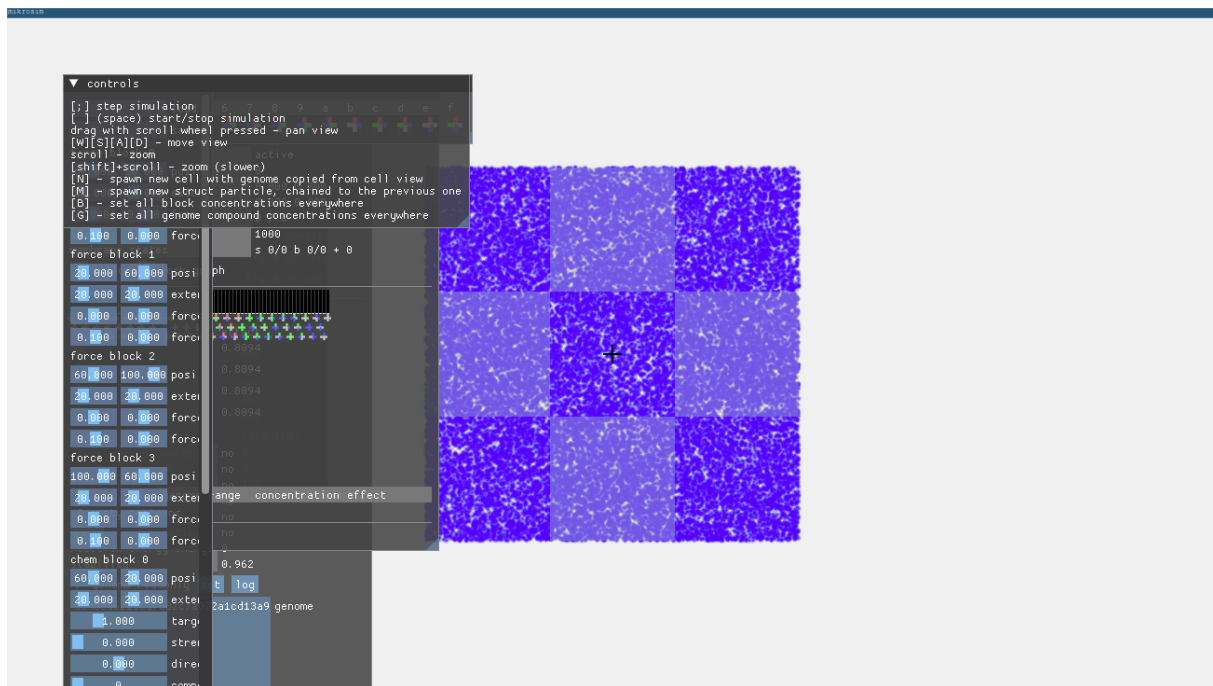
4.1 Návod k použití — Kompilace

```
git clone --recurse-submodules https://github.com/nat-int/mikrosim.git
cd mikrosim
cmake -S. -Bbuild -DCMAKE_BUILD_TYPE=Release -DVULKAN_VALIDATION=OFF \
-D_COMPILE_SHADERS=OFF
cmake --build build
```

Výsledný spustitelný soubor pak je `./out/mikrosim`, případně `./out/mikrosim.exe`, (relativně k cestě po příkazu `cd mikrosim`). Je ho potřeba spouštět ze složky `./out/`.

4.2. Návod k aplikaci

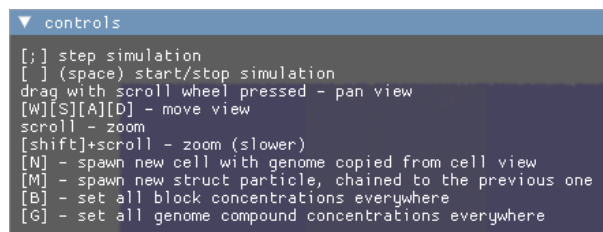
Po prvním spuštění vypadá aplikace přibližně takto:



Obrázek 9: aplikace po spuštění

Obsahuje několik podoken, se kterými se dá levým tlačítkem myši pohybovat a připínat k hlavnímu oknu nebo do sebe navzájem. Mimo podoken je vidět zobrazení simulace (modrý čtverec za okny).

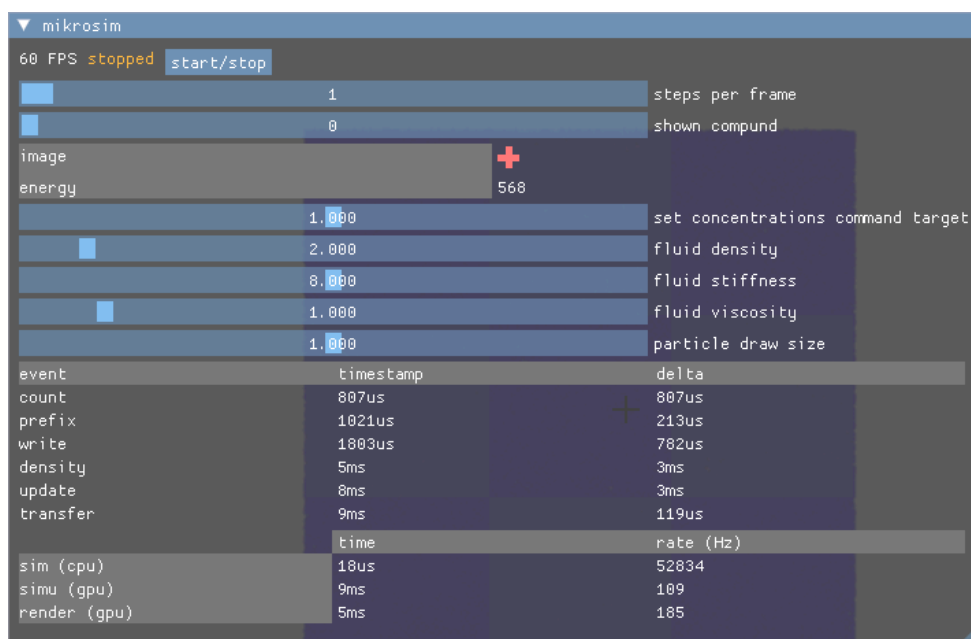
4.2.1. okno „Controls“



Okno „Controls“ obsahuje seznam kláves/akcí pro interakci se zobrazením simulace s popisem jejich efektu.

4.2 Návod k použití — Návod k aplikaci

4.2.2. okno „mikrosim“



Okno mikrosim začíná počtem snímků v poslední vteřině, poté text říkající, zda simulace běží a tlačítko, které simulaci spustí, nebo zastaví.

Na druhém řádku je posuvník, na kterém se nastavuje počet kroků simulace, které se provedou mezi každými dvěma snímky (když simulace běží).

Na třetím řádku je posuvník, na kterém se nastavuje látka, která se právě zobrazuje. Pod ním je obrázek dané látky a energie uložená v jejích vazbách.

Další posuvník nastavuje koncentraci, na kterou klávesy G a B nastavují (když se stiskne klávesa B když žádné okno nemá „focus“, tak se koncentrace všech proteinogenních látek nastaví na tuto hodnotu).

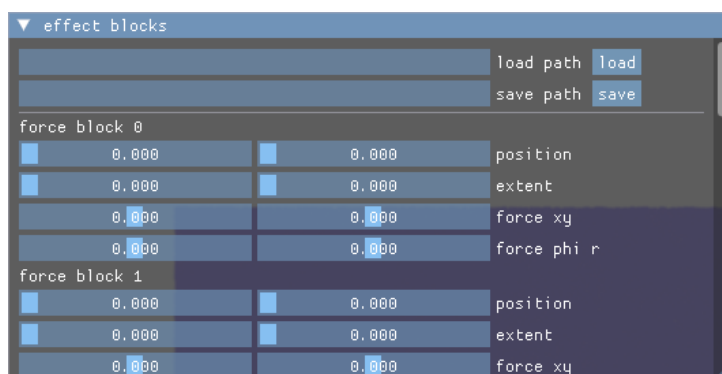
Následující 3 posuvníky nastavují vlastnosti tekutiny - hustotu, které se snaží dosáhnout, koeficient síly vyrovnávání hustoty a viskozitu.

Poslední posuvník nastavuje velikost částic při jejich vykreslování.

Pod posuvníky jsou tabulky s časy jednotlivých částí výpočtů simulace a vykreslování.

4.2 Návod k použití — Návod k aplikaci

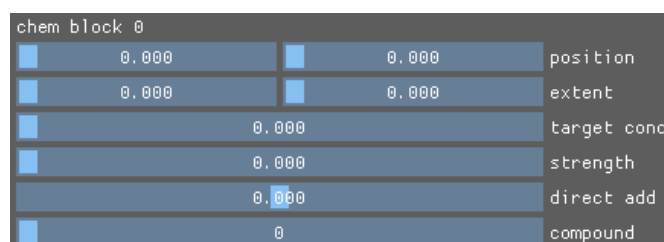
4.2.3. okno „effect blocks“



Toto okno obsahuje nastavení „efektových bloků“, kterými se dá ovlivňovat simulace. Existují jich 2 typy: silové („force blocks“) a chemické („chem blocks“). Silové bloky umožňují působit silou na částice v obdélníkové oblasti simulace. Chemické umožňují ovlivňovat koncentrace nějaké látky v obdélníkové oblasti simulace. Od obou typů jsou dostupné 4 bloky. Bloky se vykreslují přes zobrazení simulace s vysokou průhledností, silové bloky červeně a chemické zeleně.

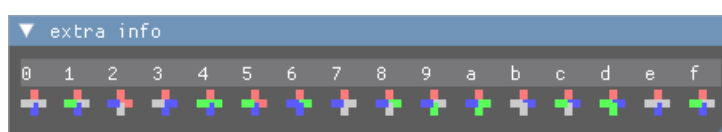
Na začátku jsou dvě textová pole s tlačítky - první umožňuje načíst nastavení všech bloků ze souboru, jehož cesta je napsána v textovém poli. Druhý umožňuje do souboru v textovém poli nastavení všech bloků uložit.

Pak následují nastavení silových bloků. Každé začíná pozicí a velikostí v osách x a y. Na třetím řádku posuvníků je nastavení homogení síly v osách x a y. Na čtvrtém řádku je nastavení síly kolem středu bloku (proti směru hodinových ručiček) a síla ke středu bloku. Síly jsou malé, takže chvíli trvá, než zapůsobí.



Poté následují nastavení chemických bloků. Stejně jako u silových bloků začínají dvěma řádky nastavení pozice a velikosti. Poté následuje cílová koncentrace, ke které se koncentrace přibližuje, poté síla, která určuje rychlost tohoto přibližování. Posuvník „direct add“ udává koncentraci, která se přičítá ve všech částicím v oblasti. Nakonec je posuvník určující látku, kterou blok zrovna ovlivňuje (čísla látek jsou stejná jako v okně „mikrosim“).

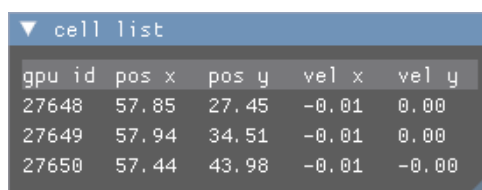
4.2.4. okno „extra info“



Okno extra info obsahuje tabulku pro převod mezi označením v genomu a látkou v proteinu.

4.2 Návod k použití — Návod k aplikaci

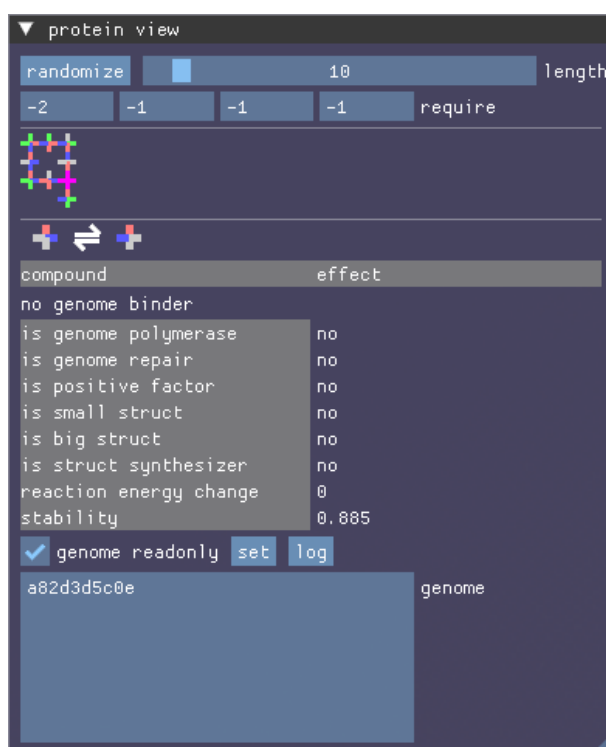
4.2.5. okno „cell list“



gpu id	pos x	pos y	vel x	vel y
27648	57.85	27.45	-0.01	0.00
27649	57.94	34.51	-0.01	0.00
27650	57.44	43.98	-0.01	-0.00

Okno cell list obsahuje seznam všech živých buněk s jejich identifikačním číslem, pozicí a rychlostí.

4.2.6. okno „protein view“



randomize ☐ 10 length

-2 -1 -1 -1 require

+

+

+

compound	effect
no genome binder	
is genome polymerase	no
is genome repair	no
is positive factor	no
is small struct	no
is big struct	no
is struct synthesizer	no
reaction energy change	0
stability	0.885

☒ genome readonly set log

a82d3d5c0e genome

Na začátku okna je ovládání generátoru proteinů. Po kliknutí na tlačítko „randomize“ se začnou generovat náhodné proteiny délky nastavené na posuvníku vedle, dokud se nenajde takový, který splňuje podmínky. Podmínky se nastavují ve čtyřech textových polích na druhém řádku. Nápověda k nim se zobrazí, když na ně najedete myší.

Následuje obrázek proteinu, který je v tomto okně vybraný. Pod ním je reakce, kterou katalyzuje a seznam modulátorů s jejich efektem.

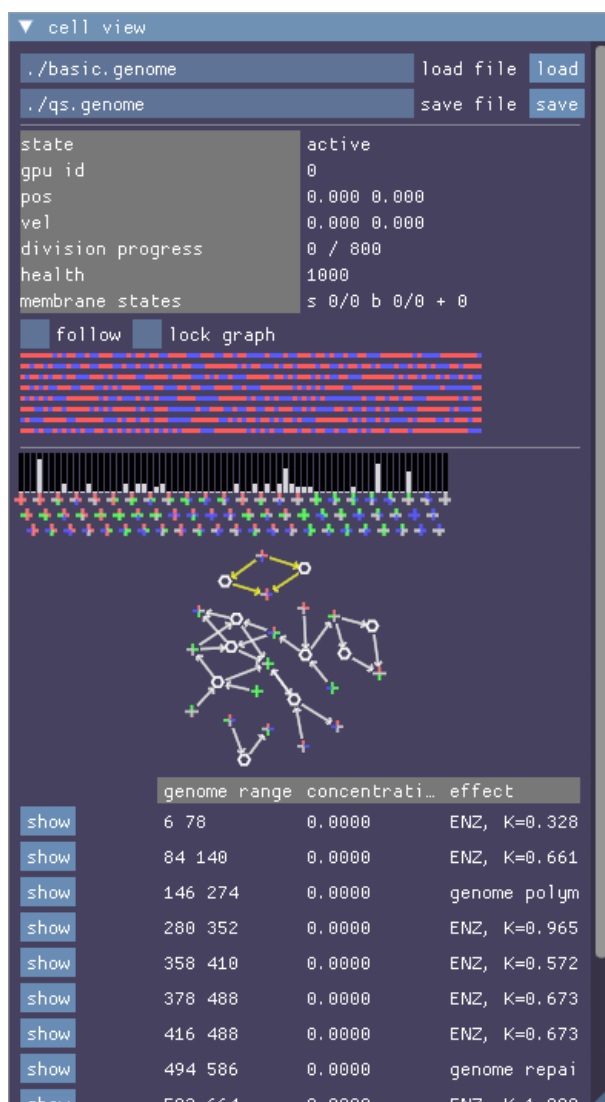
Dále je informace o tom, na jaký úsek genetického kódu se váže.

Pod ní je tabulka speciálních vlastností a hodnot proteinu.

Na konci je textové pole, ve kterém je seznam kodonů právě vybraného proteinu se zaškrťovacím políčkem. Když je políčko zaškrtnuté, tak pole nelze upravovat. Vedle jsou tlačítka „set“, které vybere protein do okna a „log“, které vypíše zprávu obsahující obsah pole na stdout.

4.2 Návod k použití — Návod k aplikaci

4.2.7. okno „cell view“



Stejně jako okno „effect blocks“, začíná toto okno poli na načítání a ukládání ze souboru. Při načítání ze souboru se ze souboru přečtou znaky označující kodony a báze (0-9, a-f, T, F) a do okna se vybere buňka s takovým genomem. Soubor může obsahovat komentáře, které začínají středníkem a končí odřádkováním.

Poté je v okně tabulka informací o buňce - stav, identifikátor, pozice, rychlost, pozice místa, ze kterého se právě kopíruje genom, zdraví a stavy membrány (první dvojice čísel je množství malé struktury a využití množství malé struktury, druhá dvojice je pro velkou strukturu a poslední číslo je postup v tvorbě membrány).

Dále jsou zaškrťovací pole „follow“ a „lock graph“. Pokud je „follow“ zaškrtnuto, pohled ve zobrazení následuje vybranou buňku. Když je zaškrtnuto „lock graph“, graf metabolismu (o kterém bude řeč níže) se nepohybuje.

Následuje genom buňky nakreslený jenom barvami bází. Když se myší najede na protein v seznamu proteinů (o kterém také bude řeč později), podtrhne se žlutě místo, kde je v genomu kódován. Pokud je to transkripční faktor, vyznačí se fialově místa, kam se váže. Zároveň jsou

4.2 Návod k použití — Návod k aplikaci

vyznačeny bílou místa, kde se působí negativní transkripční faktory a zelenou místa, kde působí pozitivní transkripční faktory.

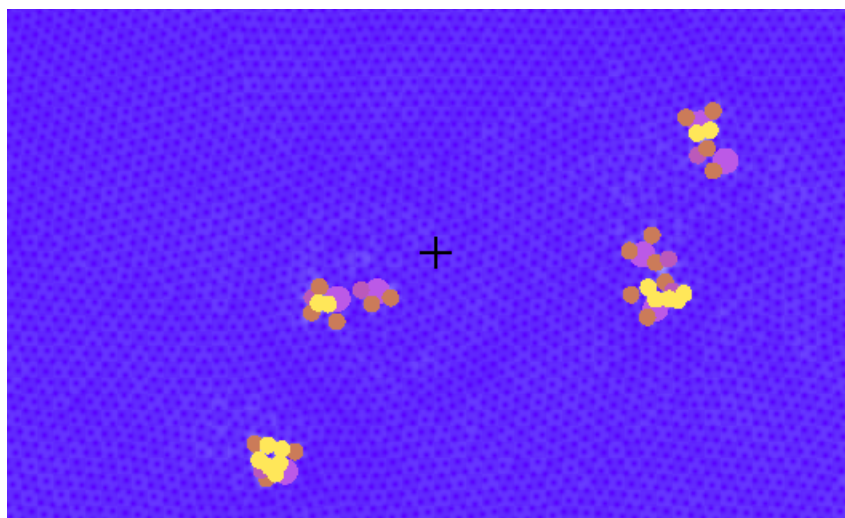
Dále se nachází graf koncentrací látek ve vybrané buňce. Když se myší najede na protein v seznamu proteinů, tak se červeně vyznačí jeho reaktanty, zeleně produkty a modře modulátory. Barvy zvýraznění se mísí, takže žluté zvýrazněný sloupek znamená, že je látka reaktant i produkt.

Okno pokračuje grafem metabolismu, kde šestiúhelníky značí proteiny a vykreslují se šipky k proteinům od jejich reaktantů a od proteinů k jejich produktům. Šipky působí jako pružiny a látky a proteiny se od sebe odpuzují. Žluté šipky značí, že protein během dané reakce provádí speciální akci. Držením kolečka myši lze pohybovat s pohledem v grafu, držením levého kolečka a pohybem myši vertikálně lze zvětšovat a zmenšovat zorné pole. Levým tlačítkem lze s proteiny a látkami pohybovat.

Další je seznam proteinů. Při kliknutí na tlačítko „show“ se daný protein vybere do okna „protein view“. Dále seznam obsahuje koncentraci proteinů a zkrácený zápis toho, co protein dělá.

Nakonec je v okně tlačítko test, které provede testy na funkci počítající chemické reakce a případné chyby vypíše na stdout.

4.2.8. Zobrazení simulace



Na pozadí je šachovnice, kde strana každého políčka má délku stejnou, jako je dosah interakcí částic. Částice tekutiny se vykreslují modře až světle modře podle své koncentrace látky, která je vybrána v okně „mikrosim“. Buňky a částice velké struktury se vykreslují fialově až světle modře. Částice malé struktury se vykreslují oranžově. Strukturní částice se vykreslují žlutě.

Ve zobrazení lze držením kolečka myši nebo klávesami W (nahoru), S (dolu), A (vlevo) a D (vpravo) pohybovat s pohledem. Točením kolečkem myši se pohled přibližuje / oddaluje. Při držení levé klávesy shift jsou pohyby (až na tah se zmáčknutým kolečkem myši) pomalejší.

Středníkem se provede jeden krok simulace, mezerníkem se spustí nebo pozastaví.

4.2 Návod k použití — Návod k aplikaci

Klávesou N se v místě křížku uprostřed obrazovky přidá nová buňka s genomem zkopírovaným od vybrané buňky z okna „cell view“, pokud vybraná buňka není mrtvá.

Klávesou G se nastaví koncentrace látek, ze kterých se tvoří genom, ve všech částicích na hodnotu nastavenou v okně „mikrosim“. Klávesa B má podobný efekt, jen nastavuje koncentrace všech modelových aminokyselin.

5. Závěr

Aplikace dokáže simulovat modelové mikroorganismy, které jsou schopné v dobrých podmínkách přežívat a množit se. Do jisté míry se mohou adaptovat na podmínky nové.

6. Seznam odkazů

(ke dni 3.3.2025)

- <https://biogenesis.sourceforge.net/>
- <https://thebibites.itch.io/the-bibites>
- <https://alien-project.org/>
- <https://developer.nvidia.com/cuda-toolkit>
- <https://github.com/SebLague/Fluid-Sim/tree/Episode-01>
- <https://github.com/ocornut/imgui>
- https://ramakarl.com/pdfs/2014_Hoetzlein_FastFixedRadius_Neighbors.pdf (https://web.archive.org/web/20250113223404/https://ramakarl.com/pdfs/2014_Hoetzlein_FastFixedRadius_Neighbors.pdf)
- <https://developer.nvidia.com/gpugems/gpugems3/part-vi-gpu-computing/chapter-39-parallel-prefix-sum-scan-cuda>
- <https://cmake.org/download/>
- <https://vulkan.lunarg.com/>
- <https://github.com/GPUOpen-LibrariesAndSDKs/VulkanMemoryAllocator>
- <https://www.glfw.org/>
- <https://github.com/g-truc/glm>
- <https://www.boost.org/users/download/>