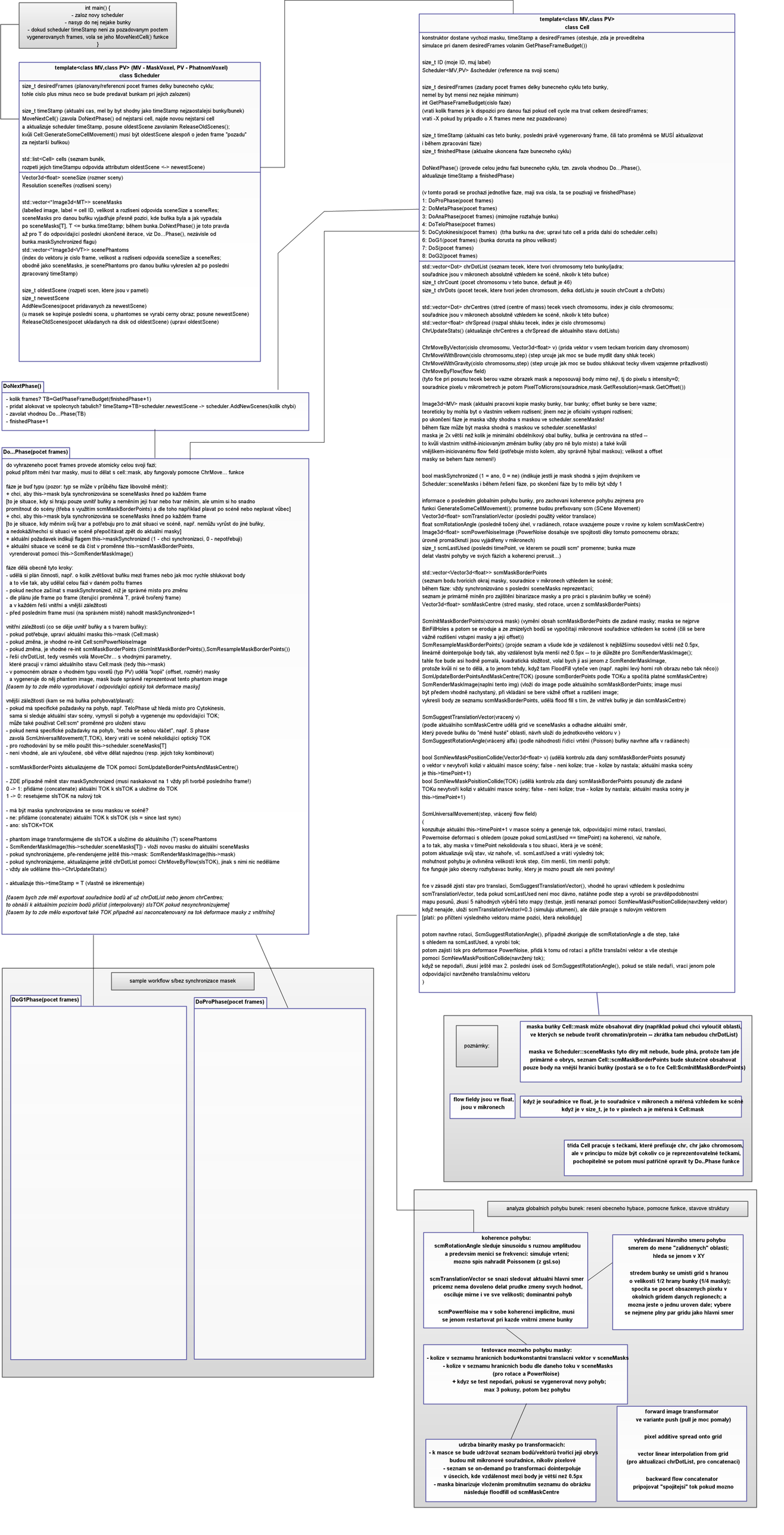
int main() {
- zaloz novy scheduler
- nasyp do nej nejake bunky
- dokud scheduler timeStamp neni za pozadovanym poctem vygenerovanych frames, vola se jeho MoveNextCell() funkce
}
template<class MV,class PV> (MV - MaskVoxel, PV - PhatnomVoxel)
class Scheduler
size_t desiredFrames (planovany/referencni pocet frames delky bunecneho cyklu;
tohle cislo plus minus neco se bude predavat bunkam pri jejich zalozeni)
size_t timeStamp (aktualni cas, mel by byt shodny jako timeStamp nejzaostalejsi bunky/bunek)
MoveNextCell() (zavola DoNextPhase() od nejstarsi cell, nagde novou nejstarsi cell
a aktualizuje scheduler timeStamp, posune oldestScene zavolanim ReleaseOldScenes();
kvůli Cell:GenerateSomeCellMovement() musí být oldestScene alespoň o jeden frame "pozadu"
za nejstarší buňkou)
std::list<Cell> cells (seznam buněk,
rozpeti jejich timeStampu odpovida attributum oldestScene <-> newestScene)
Vector3d<float> sceneSize (rozmer sceny)
Resolution sceneRes (rozliseni sceny)
std::vector<"Image3d<MT>> sceneMasks
(labelled image, label = cell ID, velikost a rozliseni odpovida sceneSize a sceneRes;
sceneMasks pro danou buňku vyjadřuje přesně pozici, kde buňka byla a jak vypadala
po sceneMasks[T], T <= bunka.timeStamp; během bunka.DoNextPhase() je toto pravda
až pro T do odpovídající poslední ukončené iterace, viz Do...Phase(), nezávisle od
bunka.maskSynchronized flagu)
std::vector<"Image3d<VT>> scenePhantoms
(index do vektoru je cislo frame, velikost a rozliseni odpovida sceneSize a sceneRes;
obodně jako sceneMasks, je scenePhantoms pro danou buňku vykreslen až po poslední
zpracovaný timeStamp)
size_t oldestScene (rozpeti scen, ktere jsou v pameti)
size_t newestScene
AddNewScenes(pocet pridavanych za newestScene)
(u masek se kopiruje posledni scena, u phantomes se vyrabi cerny obraz; posune newestScene)
ReleaseOldScenes(pocet ukladanych na disk od oldestScene) (upraví oldestScene)
DoNextPhase()
- kolik frames? TB=GetPhaseFrameBudget(finishedPhase+1)
- pridat alokovat ve spolecnych tabulich? timeStamp+TB>scheduler.newestScene -> scheduler.AddNewScenes(kolik chybi)
- zavolat vhodnou Do...Phase(TB)
- finishedPhase+1
Do...Phase(počet frames)
do vyhrazeneho pocet frames provede atomicky celou svoji fazi;
pokud přitom mění tvar masky, musí to dělat s cell::mask, aby fungovaly pomocne ChrMove... funkce
fáze je buď typu (pozor: typ se může v průběhu fáze libovolně měnit):
+ chci, aby this->mask byla synchronizována se sceneMasks ihned po každém frame
[to je situace, kdy si hraju pouze uvnitř buňky a neměním její tvar nebo tvar měním, ale umím si ho snadno
promítnout do scény (třeba s využitím scmMaskBorderPoints) a dle toho například plavat po scéně nebo neplavat vůbec]
+ chci, aby this->mask byla synchronizována se sceneMasks ihned po každém frame
[to je situace, kdy měním svůj tvar a potřebuju pro to znát situaci ve scéně, např. nemůžu vyrůst do jiné buňky,
a nedokáži/nechci si situaci ve scéně přepočítávat zpět do aktuální masky]
+ aktuální požadavek indikuji flagem this->maskSynchronized (1 - chci synchronizaci, 0 - nepotřebuji)
+ aktuální situace ve scéně se dá číst v proměnné this->scmMaskBorderPoints,
vyrenderovat pomocí this->ScmRenderMaskImage()
fáze dělá obecně tyto kroky:
- udělá si plán činnosti, např. o kolik zvětšovat buňku mezi frames nebo jak moc rychle shlukovat body
a to vše tak, aby udělal celou fázi v daném počtu frames
- pokud nechce začínat s maskSynchronized, nič je správné místo pro změnu
- dle plánu jde frame po frame (iterující proměnná T, právě tvořený frame)
a v každém řeší vnitřní a vnější záležitosti
- před posledním frame musí (na správném místě) nahodit maskSynchronized=1
vnitřní záležitosti (co se děje uvnitř buňky a s tvarem buňky):
- pokud potřebuje, upraví aktuální masku this->mask (Cell::mask)
- pokud změna, je vhodné re-init Cell:scmPowerNoiseImage
- pokud změna, je vhodné re-init scmMaskBorderPoints (ScmInitMaskBorderPoints(),ScmResampleMaskBorderPoints())
- řeší chrDotList, tedy vesměs volá MoveChr... s vhodnými parametry,
které pracují v rámci aktuálního stavu Cell::mask (tedy this->mask)
- v pomocném obraze o vhodném typu voxelů (typ PV) udělá "kopii" (offset, rozměr) masky
a vygeneruje do něj phantom image, mask bude správně reprezentovat tento phantom image
[časem by to zde mělo vyprodukovat i odpovídající optický tok deformace masky]
vnější záležitosti (kam se má buňka pohybovat/plavat):
- pokud má specifické požadavky na pohyb, např. TeloPhase už hledá místo pro Cytokinesis,
sama si sleduje aktuální stav scény, vymyslí si pohyb a vygeneruje mu opdovídající TOK;
může také používat Cell:scm* proměnné pro uložení stavu
- pokud nemá specifické požadavky na pohyb, "nechá se sebou vláčet", např. S phase
zavolá ScmUniversalMovement(T,TOK), který vrátí ve scéně nekolidující optický TOK
- pro rozhodování by se mělo použít this->scheduler.sceneMasks[T]
- není vhodné, ale ani vyloučené, obě větve dělat najednou (resp. jejich toky kombinovat)
- scmMaskBorderPoints aktualizujeme dle TOK pomocí ScmUpdateBorderPointsAndMaskCentre()
- ZDE případně měnit stav maskSynchronized (musí naskakovat na 1 vždy při tvorbě posledního frame!)
0 -> 1: přidáme (concatenate) aktuální TOK k slsTOK a uložíme do TOK
1 -> 0: resetujeme slsTOK na nulový tok
- má být maska synchronizována se svou maskou ve scéně?
- ne: přidáme (concatenate) aktuální TOK k slsTOK (sls = since last sync)
- ano: slsTOK=TOK
- phantom image transformujeme dle slsTOK a uložíme do aktuálního (T) scenePhantoms
- ScmRenderMaskImage(this->scheduler.sceneMasks[T]) - vloží novou masku do aktuální sceneMasks
- pokud synchronizujeme, pře-renderujeme ještě this->mask: ScmRenderMaskImage(this->mask)
- pokud synchronizujeme, aktualizujeme ještě chrDotList pomocí ChrMoveByFlow(slsTOK), jinak s nimi nic neděláme
- vždy ale uděláme this->ChrUpdateStats()
- aktualizuje this->timeStamp = T (vlastně se inkrementuje)
*[časem bych zde měl exportovat souřadnice bodů ať už chrDotList nebo jenom chrCentres;
to obnáší k aktuálním pozicím bodů přičíst (interpolovaný) slsTOK pokud nesynchronizujeme]*
[časem by to zde mělo exportovat také TOK případně asi naconcatenovaný na tok deformace masky z vnitřního]
sample workflow s/bez synchronizace masek
DoG1Phase(pocet frames)
DoProPhase(pocet frames)
template<class MV,class PV>
class Cell
konstruktor dostane vychozi masku, timeStamp a desiredFrames (otestuje, zda je proveditelna
simulace pri danem desiredFrames volanim GetPhaseFrameBudget())
size_t ID (moje ID, muj label)
Scheduler<MV,PV> &scheduler (reference na svoji scenu)
size_t desiredFrames (zadany pocet frames delky bunecneho cyklu teto bunky,
nemel by byt mensi nez nejake minimum)
int GetPhaseFrameBudget(cislo faze)
(vrati kolik frames je k dispozici pro danou fazi pokud cell cycle ma trvat celkem desiredFrames;
vrati -X pokud by pripadlo o X frames mene nez pozadovano)
size_t timeStamp (aktualni cas teto bunky, posledni právě vygenerovaný frame, čili tato proměnná se MUSÍ aktualizovat
i během zpracování fáze)
size_t finishedPhase (aktualne ukoncena faze bunecneho cyklu)
DoNextPhase() (provede celou jednu fazi bunecneho cyklu, tzn. zavola vhodnou Do...Phase(),
aktualizuje timeStamp a finishedPhase)
(v tomto poradi se prochazi jednotlive faze, maji sva cisla, ta se pouzivaji ve finishedPhase)
1: DoProPhase(pocet frames)
2: DoMetaPhase(pocet frames)
3: DoAnaPhase(pocet frames) (mimojine roztahuje bunku)
4: DoTeloPhase(pocet frames)
5: DoCytokinesis(pocet frames) (trha bunku na dve; upraví tuto cell a prida dalsi do scheduler.cells)
6: DoG1(pocet frames) (bunka dorusta na plnou velikost)
7: DoS(pocet frames)
8: DoG2(pocet frames)
std::vector<Dot> chrDotList (seznam tecek, ktere tvoří chromosomy teto bunky/jadra;
souřadnice jsou v mikronech absolutně vzhledem ke scéně, nikoliv k této buňce)
size_t chrCount (pocet chromosomu v teto bunce, default je 46)
size_t chrDots (pocet tecek, ktere tvori jeden chromosom, delka dotListu je soucin chrCount a chrDots)
std::vector<Dot> chrCentres (stred (centre of mass) tecek vsech chromosomu, index je cislo chromosomu;
souřadnice jsou v mikronech absolutně vzhledem ke scéně, nikoliv k této buňce)
std::vector<float> chrSpread (rozpal shluku tecek, index je cislo chromosomu)
ChrUpdateStats() (aktualizuje chrCentres a chrSpread dle aktualniho stavu dotListu)
ChrMoveByVector(cislo chromosomu, Vector3d<float> v) (prida vektor v vsem teckam tvoricim dany chromosom)
ChrMoveWithBrown(cislo chromosomu,step) (step urcuje jak moc se bude mydlit dany shluk tecek)
ChrMoveWithGravity(cislo chromosomu,step) (step urcuje jak moc se budou shlukovat tecky vlivem vzajemne pritazlivosti)
ChrMoveByFlow(flow field)
(tyto fce pri posunu tecek berou vazne obrazek mask a neposouvaji body mimo nej!, tj do pixelu s intensity=0;
souradnice pixelu v mikrometrech je potom PixelToMicrons(souradnice,mask.GetResolution)+mask.GetOffset())
Image3d<MV> mask (aktualni pracovni kopie masky bunky, tvar bunky; offset bunky se bere vazne;
teoreticky by mohla byt o vlastnim velkem rozliseni; jinem nez je oficialni vystupni rozliseni;
po ukončení fáze je maska vždy shodna s maskou ve scheduler.sceneMasks!
během fáze může být maska shodná s maskou ve scheduler.sceneMasks!
maska je 2x větší než kolik je minimální obdélníkový obal buňky, buňka je centrována na střed --
to kvůli vlastním vnitřně-iniciovaným změnám buňky (aby pro ně bylo místo) a také kvůli
vnějškem-iniciovanému flow field (potřebuje místo kolem, aby správně hýbal maskou); velikost a offset
masky se behem faze nemenil!)
bool maskSynchronized (1 = ano, 0 = ne) (indikuje jestli je mask shodná s jejím dvojnikem ve
Scheduler::sceneMasks i během řešení fáze, po skončení fáze by to mělo být vždy 1
informace o posledním globalním pohybu bunky, pro zachovani koherence pohybu zejmena pro
funkci GenerateSomeCellMovement(): promenne budou prefixovany scm (SCene Movement)
Vector3d<float> scmTranslationVector (posledni použitý vektor translace)
float scmRotationAngle (poslednè točenu úhel, v radiánech, rotace uvazujeme pouze v rovine xy kolem scmMaskCentre)
Image3d<float> scmPowerNoiseImage (PowerNoise dosahuje sve spojitosti díky tomuto pomocnemu obrazu;
úrovně promáčknutí jsou vyjádřeny v mikronech)
size_t scmLastUsed (posledni timePoint, ve kterem se pouzil scm* promenne; bunka muze
delat vlastni pohyby ve svých fázích a koherenci prerusit...)
std::vector<Vector3d<float>> scmMaskBorderPoints
(seznam bodu tvoricich okraj masky, souradnice v mikronech vzhledem ke scéně;
během fáze: vždy synchronizováno s posledni sceneMasks reprezentací;
seznam je primárně míněn pro zajištění binarizace masky a pro práci s plaváním buňky ve scéně)
Vector3d<float> scmMaskCentre (stred masky, sted rotace, urcen z scmMaskBorderPoints)
ScmInitMaskBorderPoints(vzorová mask) (vymění obsah scmMaskBorderPoints dle zadané masky; maska se nejprve
BinFillHoles a potom se eroduje a ze zmizelých bodů se vypočítají mikronové souřadnice vzhledem ke scéně (čili se bere
vážně rozlišení vstupní masky a její offset))
ScmResampleMaskBorderPoints() (projde seznam a všude kde je vzdálenost k nejbližšímu sousedovi větší než 0.5px,
lineárně dointerpoluje body tak, aby vzdálenost byla menší než 0.5px -- to je důležité pro ScmRenderMaskImage();
tahle fce bude asi hodně pomalá, kvadratická složitost, volal bych ji asi jenom z ScmRenderMaskImage,
protože kvůli ni se to dělá, a to jenom tehdy, když tam FloodFill vyteče ven (např. naplní levý horní roh obrazu nebo tak něco))
ScmUpdateBorderPointsAndMaskCentre(TOK) (posune scmBorderPoints podle TOKu a spočítá platné scmMaskCentre)
ScmRenderMaskImage(naplní tento img) (vloží do image podle aktualního scmMaskBorderPoints; image musi
být předem vhodně nachystaný, při vkládání se bere vážně offset a rozlišení image;
vykreslí body ze seznamu scmMaskBorderPoints, udělá flood fill s tím, že vnitřek buňky je dán scmMaskCentre)
ScmSuggestTranslationVector(vracený v)
(podle aktuálního scmMaskCentre udělá grid ve sceneMasks a odhadne aktuální směr,
který povede buňku do "méně husté" oblasti, návrh uloží do jednotkového vektoru v)
ScmSuggestRotationAngle(vrácený alfa) (podle náhodnosti řídící vrtění (Poisson) buňky navrhne alfa v radiánech)
bool ScmNewMaskPositionCollide(Vector3d<float> v) (udělá kontrolu zda daný scmMaskBorderPoints posunutý
o vektor v nevytvoří kolizi v aktuální masce scény; false - neni kolize; true - kolize by nastala; aktuální maska scény
je this->timePoint+1)
bool ScmNewMaskPositionCollide(TOK) (udělá kontrolu zda daný scmMaskBorderPoints posunutý dle zadané
TOKu nevytvoří kolizi v aktuální masce scény; false - neni kolize; true - kolize by nastala; aktuální maska scény je
this->timePoint+1)
ScmUniversalMovement(step, vrácený flow field)
(
konzultuje aktuální this->timePoint+1 v masce scény a generuje tok, odpovídající mírné rotaci, translaci,
Powernoise deformaci s ohledem (pouze pokud scmLastUsed == timePoint) na koherenci, viz nahoře,
a to tak, aby maska v timePoint nekolidovala s tou situací, která je ve scéně;
potom aktualizuje svůj stav, viz nahoře, vč. scmLastUsed a vrátí výsledný tok;
mohutnost pohybu je ovlivněna velikostí krok step, čím menší, tím menší pohyb;
fce funguje jako obecný rozhybavac bunky, ktery je mozno pouzit ale neni povinny!
fce v zásadě zjistí stav pro translaci, ScmSuggestTranslationVector(), vhodně ho upraví vzhledem k poslednímu
scmTranslationVector, teda pokud scmLastUsed není moc dávno, natáhne podle step a vyrobí se pravděpodobnostní
mapu posunů, zkusí 5 náhodných výběrů této mapy (testuje, jestli nenarazí pomocí ScmNewMaskPositionCollide(navržený vektor)
když nenajde, uloží scmTranslationVector/=0.3 (simuluju utlumení), ale dále pracuje s nulovým vektorem
[platí: po přičtení výsledného vektoru máme pozici, která nekoliduje]
potom navrhne rotaci, ScmSuggestRotationAngle(), případně zkoriguje dle scmRotationAngle a dle step, také
s ohledem na scmLastUsed, a vyrobí tok;
potom zajistí tok pro deformace PowerNoise, přidá k tomu od rotací a přičte translační vektor a vše otestuje
pomocí ScmNewMaskPositionCollide(navržený tok);
když se nepodaří, zkusí ještě max 2. poslední úsek od ScmSuggestRotationAngle(), pokud se stále nedaří, vrací jenom pole
odpovídající navrženého translačnímu vektoru
)
poznámky:
maska buňky Cell::mask může obsahovat díry (například pokud chci vyloučit oblasti,
ve kterých se nebude tvořit chromatin/protein -- zkrátka tam nebudou chrDotList)
maska ve Scheduler::sceneMasks tyto díry mít nebude, bude plná, protože tam jde
primárně o obrys, seznam Cell::scmMaskBorderPoints bude skutečně obsahovat
pouze body na vnější hranici buňky (postará se o to fce Cell:ScmInitMaskBorderPoints)
flow fieldy jsou ve float,
jsou v mikronech
když je souřadnice ve float, je to souřadnice v mikronech a měřená vzhledem ke scéně
když je v size_t, je to v pixelech a je měřená k Cell::mask
třída Cell pracuje s tečkami, které prefixuje chr, chr jako chromosom,
ale v principu to může být cokoliv co je reprezentovatelné tečkami,
pochopitelně se potom musí patřičně opravit ty Do..Phase funkce
analyza globalnich pohybu bunek: reseni obecneho hybace, pomocne funkce, stavove struktury
koherence pohybu:
scmRotationAngle sleduje sinusoidu s ruznou amplitudou
a predevsim menici se frekvenci: simuluje vrteni;
mozno spis nahradit Poissonem (z gsl.so)
scmTranslationVector se snazi sledovat aktualni hlavni smer
pricemz nema dovoleno delat prudke zmeny svych hodnot,
osciluje mirne i ve sve velikosti; dominantni pohyb
scmPowerNoise ma v sobe koherenci implicitne, musi
se jenom restartovat pri kazde vnitrni zmene bunky
vyhledavani hlavního smeru pohybu
smerem do mene "zalidnenych" oblasti;
hleda se jenom v XY
stredem bunky se umisti grid s hranou
o velikosti 1/2 hrany bunky (1/4 masky);
spocita se pocet obsazenych pixelu v
okolnich gridem danych regionech; a
mozna jeste o jednu uroven dale; vybere
se nejmene plny par gridu jako hlavni smer
testovace mozneho pohybu masky:
- kolize v seznamu hranicnich bodu+konstantni translacni vektor v sceneMasks
- kolize v seznamu hranicnich bodu dle daneho toku v sceneMasks
(pro rotace a PowerNoise)
+ kdyz se test nepodari, pokusi se vygenerovat novy pohyb;
max 3 pokusy, potom bez pohybu
udrzba binarity masky po transformacich:
- k masce se bude udržovat seznam bodů/vektorů tvořící její obrys
budou mít mikronové souřadnice, nikoliv pixelové
- seznam se on-demand po transformaci dointerpoluje
v úsecích, kde vzdálenost mezi body je větší než 0.5px
- maska binarizuje vložením promítnutím seznamu do obrázku
následuje floodfill od scmMaskCentre
forward image transformator
ve variante push (pull je moc pomaly)
pixel additive spread onto grid
vector linear interpolation from grid
(pro aktualizaci chrDotList, pro concatenaci)
backward flow concatenator
pripojovat "spojitejsi" tok pokud mozno