RESEARCH ARTICLE

Gell: A GPU-powered 3D hybrid simulator for large-scale multicellular system

Jiayi Du¹, Yu Zhou², Lihua Jin², Ke Sheng³*

¹ Department of Radiation Oncology, University of California, Los Angeles, Los Angeles, California, United States of America, ² Department of Mechanical and Aerospace Engineering, University of California, Los Angeles, Los Angeles, California, United States of America, ³ Department of Radiation Oncology, University of California, San Francisco, San Francisco, California, United States of America

* Ke.Sheng@ucsf.edu

Abstract

As a powerful but computationally intensive method, hybrid computational models study the dynamics of multicellular systems by evolving discrete cells in reacting and diffusing extra-cellular microenvironments. As the scale and complexity of studied biological systems continuously increase, the exploding computational cost starts to limit large-scale cell-based simulations. To facilitate the large-scale hybrid computational simulation and make it feasible on easily accessible computational devices, we develop Gell (GPU Cell), a fast and memory-efficient open-source GPU-based hybrid computational modeling platform for large-scale system modeling. We fully parallelize the simulations on GPU for high computational efficiency and propose a novel voxel sorting method to further accelerate the modeling of massive cell-cell mechanical interaction with negligible additional memory footprint. As a result, Gell efficiently handles simulations involving tens of millions of cells on a personal computer. We compare the performance of Gell with a state-of-the-art paralleled CPU-based simulator on a hanging droplet spheroid growth task and further demonstrate Gell with a ductal carcinoma in situ (DCIS) simulation. Gell affords ~150X acceleration over the paralleled CPU method with one-tenth of the memory requirement.

Introduction

Computational modeling has become an important tool for studying the dynamics of tissue development and tumor response to different therapeutic interventions over the past three decades. Three major types of models are commonly utilized in these studies: discrete, continuum, and hybrid models [1]. Discrete models, also known as cell-based models or agent-based models, simulate the individual behaviors and the mutual interactions of the cells in a system. Continuum models consider biological tissues as domains composed of different solid and fluid phases and describe the system evolution using partial differential equations. Hybrid models combine the aforementioned two methods as they model discrete cells in a continuum environment.

Cell-based and hybrid simulations adopt the discrete representation of the cell of interest. Compared to the continuum representation, the discrete modeling of individual cells better captures the heterogeneity of tumorous tissue with independently tracked cell states and
enables a more straightforward translation from biological hypothesis to simulation rules [2].

Cell-based/hybrid simulation has been utilized to explore various kinds of oncological topics [3], such as epithelial ducts and cysts through the epithelial acini models [4, 5], the development of the initial phases of avascular tumor development through multicellular tumor spheroid (MCS) models [6], angiogenesis, vascular network formation problems [7], and anti-cancer treatments effectiveness modeling [8, 9]. The method for discrete representation of cells can either be lattice-based or off-lattice. Lattice-based methods are mesh-based, including cellular automaton (CA) models [10], lattice gas CA (LGCA) models [11], and Cellular Potts models (CPM) [12]. Lattice-based models are susceptible to grid biases [13], which do not affect the off-lattice methods. Two major types of off-lattice models are Boundary-tracking models [14] and Center-Based models (CBMs) [1, 15]. Boundary-tracking models dedicate more computational resources to the cells’ morphological dimension and are thus more computationally expensive than CBMs [2]. CBMs assume a spherical cell shape and represent cell movement by displacing the cell center position. Cell movements can be realistically modeled by incorporating forces such as adhesive, repulsive, locomotive, and drag-like forces [15]. Center-based representation is often a superior choice for large-scale cell-based/hybrid simulations interested in the heterogeneous development of biological tissue due to its realistic modeling of multicellular interactions and lower computational cost than other cell-based methods.

However, even with a center-based representation of discrete cells, as the scale and complexity of the studied biological systems continuously increase, the exploding computational cost still limits the large-scale cell-based/hybrid simulations. Several CPU-based cell-based/hybrid simulation software frameworks have been proposed to enable large-scale biological system modeling on high-performance computers. Biocellion [16], a closed-source commercial software, has simulated millions to billions of cells on a computer cluster. PhysiCell [15] is an open-source parallel simulation platform capable of simulating 18.2 days of hanging drop spheroid growth with up to one million cells in 3 days on a high-performance computer. BioDynaMo [17] is another open-source parallel simulator that is 945X faster on the ‘epidemiology (medium-scale)’ benchmark using 72 CPUs on a server compared to a single thread version. Because of the demonstrated potential to accelerate cell-based simulation with parallelization, recent research has shifted to the graphic processing unit (GPU), which has an intrinsic parallel architecture with thousands of computational cores. Ya||a [18] is a paralleled agent-based model on GPU. Its extended spheroid cell model with spin-like polarities can simulate epithelial sheets and tissue polarity. Although Ya||a [18] can achieve 10X acceleration compared to CPU-based cell-based simulation library Chaste [19], their simulation software is not designed for large systems. Simulation of large systems using Ya||a [18] is limited by its computational complexity of O(N^2) for cell-cell interaction. CBMOS [20] is another GPU-based software that provides a platform to study the effects of force functions, ODE solvers, time step sizes, and cellular events in CBMs. CBMOS [20] utilizes fast GPU vector operations provided by CuPy for efficient calculations and achieves a simulation speed 30X faster than their CPU version. Their emphasis, however, is on a better user interface for fast prototyping of new models. Its ability to handle large systems is still limited by the platform design, e.g., the force calculation time complexity is O(N^2), and the GPU memory consumption can exceed 16 GB (e.g., NVIDIA Tesla T4) for >10^4 cells. GPU BioDynaMo [21] upgrades BioDynaMo [22] by enabling GPU co-processing. With a GTX 1080Ti GPU, GPU BioDynaMo can be 130X faster than the single thread CPU-only version BioDynaMo. However, GPU BioDynaMo does not solve PDEs on GPU, and its uniform grid method for force calculation still needs CPU for linked list maintenance, which significantly limits its real-world performance.

Therefore, although acceleration of cell simulation using GPU has been demonstrated, the potential has not been fully realized. Specifically, the cell-cell interaction has not been fully...
parallelized, and the slow data transfer between CPU and GPU results in significant overhead. In this study, we develop a new open-source fast and memory-efficient fully GPU-based hybrid simulation software, GPU cell (Gell), to overcome these bottlenecks for large-scale hybrid cell simulation.

**Methods**

**Cell model**

**Cell cycle and death.** We model five cell phases in Gell. The premitotic, postmitotic, and quiescent phases are for living cells, and the necrotic and apoptotic phases are for cell death. The phase transition from premitotic to postmitotic and from postmitotic to quiescent are deterministic with a fixed gap time, respectively. Meanwhile, the phase transition from the quiescent phase to the premitotic phase and any phase transition from the living cell phase to the dead cell phase are all stochastic with a certain transition rate. The respective transition rates are listed in Table 1.

The probability for any stochastic transition $\alpha$ with transition rate $r_\alpha$ to take place in a short time interval $\Delta t$ is given by:

$$\text{Prob}_\alpha = r_\alpha \Delta t.$$  \hfill (1)

During the quiescent phase, cells maintain their standard volume and remain to be stochastically activated for division preparation at the rate $r_{\text{pro}}(P_{\text{oxy}})$:

$$r_{\text{pro}}(P_{\text{oxy}}) = \begin{cases} \frac{r_{\text{pro, max}}}{P_{\text{oxy}} - T_{\text{h pro}}} & P_{\text{oxy}} \geq S_{\text{a pro}}, \\ \frac{r_{\text{pro, max}}}{S_{\text{a pro}} - T_{\text{h pro}}} & T_{\text{h pro}} < P_{\text{oxy}} < S_{\text{a pro}}, \\ 0 & P_{\text{oxy}} \leq T_{\text{h pro}}. \end{cases}$$ \hfill (2)

The proliferation rate increases linearly with local oxygen concentration in the given range $T_{\text{h pro}} < P_{\text{oxy}} < S_{\text{a pro}}$, where $T_{\text{h pro}}$ is the minimum oxygen partial pressure required for proliferation, $S_{\text{a pro}}$ is the saturation oxygen partial pressure when the transition rate for proliferation reaches the maximum value $r_{\text{pro, max}}$.

Once activated for proliferation, cells enter the premitotic phase and prepare for division. Premitotic cells gradually gain mass/volume and then divide at the end of this phase and enter the postmitotic phase. The division process is mechanically modeled to finish in an instant, and the two daughter cells equally inherit half of the parent cell volume and be placed around the parent cell center with the displacement $x_{\text{disp}}$\[15\].

$$x_{\text{disp}} = \pm \left( R - \frac{R}{\sqrt{2}} \right) \theta.$$ \hfill (3)

**Table 1. Phase transition in Gell.**

| Transition Rate | Premitotic Phase | Postmitotic Phase | Quiescent Phase | Apoptotic Phase | Necrotic Phase |
|-----------------|------------------|-------------------|----------------|----------------|---------------|
| Premitotic Phase | $1/t_{\text{prem}}$ | 0                 | 0              | 0              | 0             |
| Postmitotic Phase | 0               | $1/t_{\text{postm}}$ | 0              | 0              | 0             |
| Quiescent Phase | $r_{\text{pro}}(P_{\text{oxy}})$ | 0               | $r_{\text{apop}}$ | 0              | $r_{\text{nec}}(P_{\text{oxy}})$ |

The phase transition rate in the cell cycle model from the left column phase to the top row phase. The transition rate for deterministic phase duration is noted as one over the fixed phase duration.

https://doi.org/10.1371/journal.pone.0288721.t001
Where $R$ is the cell radius of the parent cell, and $\theta$ is a three-dimensional random unit vector. The postmitotic phase accounts for the duration required for daughter cells to reach mechanical equilibrium and grow to be proliferation ready.

Cell death is activated stochastically for all living cells. The transition rate to enter the apoptotic phase is a constant for all cells, $r_{apop}$, while the transition rate for necrotic death depends on the local oxygen partial pressure $P_{oxygen}$:

$$
r_{nec}(P_{oxygen}) = \begin{cases} 
0 & P_{oxygen} \geq Th_{nec}, \\
\frac{Th_{nec} - P_{oxygen}}{Th_{nec} - Sa_{nec}} Sa_{nec} & Sa_{nec} < P_{oxygen} < Th_{nec}, \\
r_{nec\_max} & P_{oxygen} \leq Sa_{nec}.
\end{cases}
$$

(4)

Necrosis happens only when local oxygen partial pressure is lower than the necrosis threshold $Th_{nec}$. The transition rate increases linearly as the oxygen concentration decreases till the maximum necrosis rate $r_{nec\_max}$ is reached when oxygen partial pressure equals the necrosis saturation threshold $Sa_{nec}$.

All cell components start to shrink for apoptotic death upon entering the phase. While for the necrotic phase, early necrotic cells first absorb fluid and swell in the oncosis process, and then enter the late necrosis process after the membrane ruptures and start to lose fluid components. Modeling of two-stage necrotic death can be critical when studying the microstructures in the hanging drop spheroid necrotic center \[15\]. Phase transition-related parameters can be found in Table 2, where the listed values are all adopted from the "Ki67 Advanced" model from PhysiCell \[15\].

Following PhysiCell \[15\], we divide the total cell volume $V$ into the fluid volume $V_F$ and the solid biomass volume $V_S$. The solid biomass volume is further divided into total nuclear volume $V_{NS}$ and cytoplasmatic volume $V_{CS}$. Different cell components have different rates of volume gain and loss in different phases. All the volume changes of different cell components are modeled using ordinary differential equations (ODEs):

$$
\frac{dV_i(t)}{dt} = r_{p,i} \left( V_i(t) - V^p_i \right).
$$

(5)

Where $V_i$ is the volume of component $i$, $r_{p,i}$ is the volume change rate of component $i$ in phase $p$, and $V^p_i$ is the desired volume of component $i$ in phase $p$. Specially, the desired fluid volume of a cell is a function of the current total cell volume $V$. Related parameters can be found in Table 3, adopted from MCF-10A human breast cancer cell line \[15\].

### Table 2. Phase transition-related parameters.

| Parameter $T_{prem}$ | Biophysical meaning | Reference Value |
|----------------------|---------------------|-----------------|
| $T_{prem}$ | Duration of premitotic phase | 13 hour |
| $T_{postm}$ | Duration of postmitotic phase | 2.5 hour |
| $r_{apop}$ | Apoptosis rate | 0.0060 hour$^{-1}$ |
| $r_{pro\_max}$ | Max proliferation rate | 0.1176 hour$^{-1}$ |
| $Sa_{pro}$ | The oxygen level when the proliferation rate reaches maximum | 10 mmHg |
| $Th_{pro}$ | The oxygen level when the proliferation rate drops to zero | 5 mmHg |
| $r_{nec\_max}$ | Max necrosis rate | 0.1667 hour$^{-1}$ |
| $Sa_{nec}$ | The oxygen level when the necrosis rate reaches maximum | 2.5 mmHg |
| $Th_{nec}$ | The oxygen level when the necrosis rate drops to zero | 5 mmHg |

https://doi.org/10.1371/journal.pone.0288721.t002
Cell mechanics. Cells in Gell are mechanically modeled as elastic balls with varied volumes and center positions. Cells adhere to each other while attached and push against each other upon compression. The motion of cell i at position $x_i(t)$, with velocity $v_i(t)$, and with a set $N_i(t)$ of nearby cells can be modeled as [15]:

$$m_i \ddot{v}_i = \sum_{j \in N(i)} (F_{cca}^j + F_{ccr}^j) + F_{mot} - \eta_i v_i.$$  \hspace{1cm} (6)

Where $\ddot{v}_i$ is the acceleration, $F_{cca}^j$ denotes the adhesive force from cell j to cell i, and $F_{ccr}^j$ represents the repulsive force from cell j to cell i. $F_{mot}$ accounts for the force related to cell migration. $\eta_i v_i$ represents the resistance contributed by the local microenvironment, such as fluid resistance and cell-matrix adhesion forces. $\eta_i$ is a fluid-drag-coefficient parameter, and $v_i$ is the cell velocity.

The force equilibrates at relatively short time scales relative to the time scale of cell volume change and multicellular patterning. Therefore, we can safely apply the zero acceleration inertialess condition to Eq (6) and explicitly solve $v_i$ by:

$$v_i = \frac{1}{\eta_i} \left( \sum_{j \in N(i)} (F_{cca}^j + F_{ccr}^j) + F_{mot} \right).$$  \hspace{1cm} (7)

The adhesive force and repulsive force experienced by cell i are modeled as:

$$F_{cca}^j = \begin{cases} C_{cca} \left(1 - \frac{|r_{ij}|}{R_A} \right)^2 \frac{r_{ij}}{|r_{ij}|} & \text{if } |r_{ij}| \leq R_A, \\ 0 & \text{otherwise.} \end{cases}$$  \hspace{1cm} (8)

$$F_{ccr}^j = \begin{cases} -C_{ccr} \left(1 - \frac{|r_{ij}|}{R_R} \right)^2 \frac{r_{ij}}{|r_{ij}|} & \text{if } |r_{ij}| \leq R_R, \\ 0 & \text{otherwise.} \end{cases}$$  \hspace{1cm} (9)

$R_R$ is the maximum repulsive interaction distance that equals the sum of the radius of cell i and j. $R_A$ is the maximum adhesive interaction distance, which is slightly larger than $R_R$ due to the deformability of the two cells. $C_{cca}$ is the cell-cell adhesion parameter, and $C_{ccr}$ is the cell-cell repulsion parameter. $r_{ij}$ is a vector pointing from the center of cell i to the center of cell j.

Once the sum of the experienced force is calculated for all cells, the velocity of any cell can be directly calculated. Cell position is then updated using the second-order Adam-Bashforth method:

$$x_i(t + \Delta t) = x_i(t) + \frac{\Delta t}{2} (3 \cdot v_i(t) - v_i(t - \Delta t)).$$  \hspace{1cm} (10)
Extracellular microenvironment

The tumor is surrounded by a complex ecosystem named tumor microenvironment (TME), composed of tumor cells, stromal cells, and other extracellular physical and chemical factors. The mutual and dynamic crosstalk between the tumor and tumor microenvironment, together with the genetic/epigenetic change in tumor cells, are two factors that influence the formation and progression of the tumor [23]. In our model, cells can absorb environmental nutrients and release biochemical factors into the extracellular fluid. In addition, critical environmental factors can also regulate cell behaviors. To simulate the spatio-temporal variation of environmental factors during tumor development, we consider a continuous extracellular fluid space and use PDEs to describe the secretion, diffusion, uptake, and decay of diffusive substances such as oxygen and vascular endothelial growth factor. The continuum environment and the discrete cells are explicitly linked. Cell phases, sizes, and positions are treated as static while updating the continuous molecular space and vice versa. The equation for any diffusive substance in the extracellular fluid domain $\Omega$ can be written as [24]

$$\frac{\partial \rho}{\partial t} = \nabla \cdot (D \nabla \rho) - \lambda \rho + S(\rho^* - \rho) - U \rho. \quad (11)$$

Depending on the problem, the domain boundary $\partial \Omega$ can be either Dirichlet or Neumann type. $\rho$ is the substance concentration, $\rho^*$ is the saturation concentration, $D$ is the diffusion coefficient, $\lambda$ is the decay rate, $S$ is the supply rate, $U$ is the uptake rate.

In the provided code, the oxygen concentration is the only considered environmental factor and discrete cells are the only contributor of oxygen consumption. Cells absorb oxygen from the extracellular fluid and regulate their behavior according to local oxygen concentration. The oxygen consumption of each cell is modeled as proportional to both the oxygen concentration and the cell volume. With a cartesian grid, for each isotropic voxel $i$, the total uptake rate of oxygen equals the sum of the local cell consumption:

$$U_i = \sum_{j \in \Omega} \frac{V_j}{V_{\text{voxel}}} U_\sigma. \quad (12)$$

The uptake rate here represents the oxygen concentration decrease rate as a proportion of current concentration. $U_\sigma$ is the default oxygen consumption rate of living cells that equals 10 per min. $V_{\text{voxel}}$ is the volume of the given voxel, $j$ is the index of cells inside the voxel, and $V_j$ is the corresponding cell volume.

In the simulation, each voxel’s total oxygen consumption rate is first calculated according to the position, size, and phase of all the cells. Then the molecular space is updated using the static consumption rate map. With the calculated molecular concentration, cells in the discrete model could read the local oxygen concentration and carry on their stochastic phase transitions according to these values.

For the numerical processing of the PDE, following BioFVM [24], a first-order splitting method is first applied to split the right-hand side into simpler operators: a supply and uptake operator and a diffusion-decay operator.

$$\begin{cases} \frac{\sigma - \rho^n}{\Delta t} = \nabla \cdot (D \nabla \sigma) - \lambda \sigma, \\ \rho^{n+1} - \sigma = S(\rho^{n+1} - \sigma) - U \sigma. \end{cases} \quad (13)$$

The supply and uptake operators are handled analytically. The three-dimensional diffusion-decay operator is further split into a series of related one-dimensional PDEs using the...
locally-one dimensional (LOD) method.

\[
\begin{align*}
\eta - \rho^n &= \partial_y(D\partial_y \eta) - \frac{1}{3} \lambda \eta, \\
\eta^* - \eta &= \partial_y\left(D\partial_y \eta^*\right) - \frac{1}{3} \lambda \eta^*, \\
\sigma - \eta^* &= \partial_y(D\partial_y \sigma) - \frac{1}{3} \lambda \sigma.
\end{align*}
\tag{14}
\]

Discretized using the finite volume method, we obtain the updated concentration of each strip of voxels for each direction by solving Eq (15) using the Thomas algorithm [25].

\[
\begin{align*}
\left(1 + \frac{1}{3} \Delta t \lambda + \frac{\Delta t}{\Delta x^2} D\right)^\circ \eta_{(0,y,z)} - \frac{\Delta t}{\Delta x^2} D \eta_{(1,y,z)} &= \rho^n_{(0,y,z)} \\
- \frac{\Delta t}{\Delta x^2} D^n \eta_{(n_x-2,y,z)} + \left(1 + \frac{1}{3} \Delta t \lambda + \frac{\Delta t}{\Delta x^2} D\right)^\circ \eta_{(n_x-1,y,z)} - \frac{\Delta t}{\Delta x^2} D^n \eta_{(n_x+1,y,z)} &= \rho^n_{(n_x,y,z)} \\
- \frac{\Delta t}{\Delta x^2} D^n \eta_{(n_x-1,y,z)} + \left(1 + \frac{1}{3} \Delta t \lambda + \frac{\Delta t}{\Delta x^2} D\right)^\circ \eta_{(n_x-1,y,z)} &= \rho^n_{(n_x-1,y,z)}.
\end{align*}
\tag{15}
\]

Implementation details

Gell is developed using C++ and CUDA (v11.2), with the program’s design schematic illustrated in Fig 1. There are two major types of data in the simulation, the pre-allocated array of structure for cell data management and the isotopically discretized cartesian grid for spatial domain-related data, which includes information related to the tumor microenvironment (TME) and our Voxel Sorting (VS) method. Once initialized on the CPU, all the cellular and microenvironmental data are transferred to GPU memory. From this point, all the following computation steps are exclusively handled by GPU to eliminate costly back-and-forth data transfer between GPU and CPU.

In the provided diagram, white cylinders symbolize the simulation data, and colored boxes positioned atop this data represent associated operations. The abbreviations TME and VS are used to denote the Tumor Microenvironment and Voxel Sorting, respectively. The
Microenvironment Module, Cell Cycle Module, and Cell Mechanics Module are distinctly demarcated with dashed lines in blue, red, and green, respectively.

The main simulation loop of GeLL consists of three critical computational modules, as delineated in dashed lines in Fig 1: the Microenvironment Module, the Cell Cycle Module, and the Cell Mechanics Module. Each module is responsible for one of the key tasks in the hybrid cell-based simulation and is uniquely designed to enable efficient parallelization on the GPU. Further details on these modules will be explored in the following sections.

**Microenvironment module.** The Microenvironment Module is tasked with managing the diffusive transportation and cellular secretion/uptake of environmental factors, such as oxygen. This module is divided into two sections: a Supply-Uptake Submodule and a Diffusion-Decay LOD Kernel.

The Supply-Uptake Submodule deals with the supply and uptake of microenvironmental factors. This process involves two CUDA kernels: a 1D kernel parallelized over the cells to calculate each voxel’s total supply and uptake, followed by a 3D kernel parallelized over voxels to adjust the factor concentration after supply and uptake.

The Diffusion-Decay LOD Kernel addresses the diffusion and decay of environmental factors using the LOD algorithm. During each diffusion-reaction step, the LOD solver transforms the concentration update problem along the three axes into $3^N N^2$ tridiagonal linear systems, each with $N$ unknowns. Each parallel thread solves one of these tridiagonal linear systems using the Thomas algorithm. The overall computational complexity of each update step is $O(N_{\text{voxel}})$.

**Cell cycle update.** The Cell Cycle Module is responsible for managing cell proliferation, growth, and death. It is divided into two components, the cell cycle kernel, and the cell birth-death submodule.

The cell growth and phase transition only involve the current cell status as well as the local environmental factors. These processes are highly parallelizable and are directly distributed across different threads and computed by GPU in the Cell Cycle Kernel. The Cell Birth-Death Submodule manages cell proliferation and death, focusing on the memory management of structured data. By leveraging this submodule, fast simulations can be conducted with only minimal additional computation and memory cost for thread competition avoidance. During cell proliferation, the total cell count is atomically updated to enable the addition of new daughter cells to the cell array. For cell death, cells earmarked for removal are initially labeled in their respective phase update thread and then collectively deleted from the cell array in a subsequent kernel for efficient cell array maintenance. The overall computational complexity of the above-mentioned processes is approximately $O(N)$.

**Cell mechanics update.** N-body interaction simulations can be extremely expensive due to their $O(N_{\text{cell}}^3)$ computational complexity. For a large multicellular system with millions of cells, it is computationally impractical just to loop over all the cell pairs, even with GPU [18, 20]. Fortunately, the cell-cell mechanical interactions are short-range interactions, making it reasonable to calculate only the forces between neighboring cells, thus reducing the computational complexity to $O(N_{\text{cell}})$. PhysiCell utilizes the cell-cell interaction data structure (IDS) method [15]. A large number of lists are created and maintained to record the indices of cells inside each voxel. The force calculation for each cell only has to loop over the cells inside its nearest 27 voxels, according to the lists. GPU BioDynaMo, with its uniform grid method [21], improves memory usage efficiency by replacing the cell index list with the linked list. However, maintaining such cell lists or linked lists is not GPU-friendly. GPU does not allow us to dynamically allocate memory in the thread, making it challenging to create lists with dynamic lengths for all mechanical voxels in the IDS method. Suppose GPU memory for cell lists is all pre-allocated according to the maximum cell density. In that case, memory usage can be highly
inefficient due to the high cell density necrotic region. The challenge lies in the maintenance of the linked list for the uniform grid method on GPU. Thread locks are required to update the linked list correctly, but the wrapping mechanism of CUDA could easily create deadlocks during such processes and pause the program indefinitely.

To realize an efficient cell-cell mechanics computation on GPU, we have developed our Voxel-Sorting Method. Cells are stored in array-of-structures, and a Morton code is generated for each cell according to the i, j, k index of its containing mechanical voxel as a key for sorting. Then a fast GPU-based radix sort algorithm [26] of complexity $O(N_{\text{cell}})$ is used to rearrange the cell array according to the ascending Morton code value order. After sorting, cells in the same voxel are stored contiguously in the GPU memory, and cells in adjacent voxels are stored relatively close. Such a contiguous memory layout increases the memory fetch efficiency, especially when groups of cells in the same voxel have to be accessed together by the same thread. The array index range for cells inside each voxel is easily determined with $O(N_{\text{voxel}})$. Fig 2 illustrates the voxel sorting method in a 2D scene.

This rapid cell-cell interaction simulation method described above is implemented in the three-step Cell Mechanics Module. The first Morton Encoding kernel determines the order index of each cell. Then the second Voxel Sorting Submodule sorts the cells accordingly and determines the index range of cells within each voxel. In the last step, the aggregated force and cell movements of each cell are calculated by the Cell Mechanics Submodule.

**Time scale considerations.** Different biological processes evolve at different time scales: the temporal scale of cell colony biology and mechanics is on the order of minutes, while the equilibrium of transport diffusion is achieved in seconds [15]. Because the extracellular environment updates substantially faster than cellular evolution, it is computationally inefficient to

---

**Fig 2. 2D illustrations of voxel sorting method.** Left: The Morton code maps 2D position to 1D number while preserving the locality of the data points. The Morton code is generated by interleaving the binary x and y pixel index values. Adapted from [27]. Right: The force calculation example of a single cell (red). The red circle represents the maximum range of cell-cell mechanical interaction whose radius is shorter than the side length of voxels. The Force module only needs to loop over the cells in yellow voxels to calculate the aggregated force, and the indexes of cells inside these voxels can be easily figured out after the sorting. As long as the voxel side length is longer than the maximum cell-cell interaction distance, each cell’s aggregated cell-cell mechanical interaction can be calculated efficiently by looping only over cells inside its nearest 27 voxels. Our voxel sorting method achieved force calculation time complexity of $O(N_{\text{cell}})$ and high GPU memory utilization efficiency. An additional advantage of our voxel sorting method is memory access efficiency. With the cells in the same voxel stored next to each other in the GPU memory after voxel sorting, the following 27 times of memory fetch of these cell data can become much more efficient than fully random memory access. After the force calculation, the cell position update can be directly parallelized using the aggregated force information.

https://doi.org/10.1371/journal.pone.0288721.g002
synchronize the cell simulator and extracellular environment simulator updates. Instead, we first fixed cellular properties while solving the PDEs for the extracellular environment at a higher frequency and then evolved cell phenotype and mechanical interaction at a lower frequency to reduce the simulation cost.

Results

All calculations are tested on a personal computer (with Intel® Core™ i7-7700K CPU, 64GB memory, and an NVIDIA® GeForce® RTX 2080Ti graphics card) on 64-bit Windows 10. Tests are conducted to determine the optimal block size settings for GPU computation during the program development, which have been fixed for later simulations. Specifically, the kernels for cell cycle update have a 1D block width of 64, while the 3D kernels for oxygen consumption employ a block width of 4. As for the 2D kernels used for LOD calculation, we have adopted a small 2D block width of 4 due to the storage requirements of the Thomas algorithm on GPU. This algorithm necessitates the storage of a temporary array for each thread, and a small block size allows us to store all data on the registers.

Hanging drop spheroid

As an in vitro 3D multicellular model, multicellular tumor spheroids (MCTS) possess many in vivo tumor features, including cell-cell interaction, hypoxia, treatment response, and production of extracellular matrix [28, 29]. Tumor spheroids are widely used for various tumor growth dynamics and treatment response studies. Many modeling works have been published to mathematically bridge the observed spheroid experiment results to mechanistic understandings. Joshua A. Bull et al. [30] developed an off-lattice hybrid spheroid growth model to explore the growth dynamics of tumor spheroids and reproduced the migration and internalization of tumor cells observed in spheroid experiments [31]. Kevin O. Hicks et al. [32] developed an on-lattice hybrid spheroid model using experimentally determined parameters and accurately simulated the cell killing after radiation and hypoxia-activated prodrug interventions.

To compare the computational performance of our simulation framework with existing simulation software, we simulated a benchmark problem of hanging drop spheroid growth [15]. In the hanging drop spheroid (HDS) benchmark, a suspended multicellular aggregate is cultured in the middle of a growth medium with oxygen supplied through diffusion from the domain boundary. All the Gell simulations and PhysiCell simulations share identical simulation settings. The simulation started with 2347 cells and evolved to nearly one million cells after 450 hours of cultivation. The simulation domain contains one million isotropic voxels with a side length of 25 μm. Cell mechanics and phase update is calculated every 0.1 minutes, and the diffusion-reaction of oxygen in the extracellular fluid is solved every 0.01 minutes. At the end of the simulation, Gell and PhysiCell produce nearly identical results, predicting a spheroid radius of 1.87 mm and evolving visually identical spheroid structures, including the crack patterns in the necrotic cores. These findings suggest that the computation of Gell is accurate, and the use of single precision arithmetic has a negligible impact on the simulation result. The evolution of cell number and spheroid radius over time is shown in Fig 3. The result shows no necrosis development until the spheroid reaches a certain radius when oxygen diffusion from the outer rim becomes insufficient to support the inner cells. The radius growth curve exhibits a linear shape due to the approximately constant viable rim thickness, which agrees with the simulation result and theoretical prediction of PhysiCell [15].

The rendered images of the simulation results are shown in Fig 4. There is a clear, viable rim of actively proliferating cells at the outer shell of the spheroid and a necrotic core in the
Additionally, the subtle cell-cell mechanical adhesion and crack-like microstructures successfully emerged in the necrotic core entry. The simulation results of Gell agree with PhysiCell and \textit{in vitro} experiments \cite{15}.

Gell completed the entire simulation process in 47 minutes using the personal computer. As a comparison, the state-of-the-art CPU-based paralleled simulator PhysiCell used 119 hours for the same simulation on the same personal computer. In other words, Gell is 150X faster for the cell simulation problem of this scale.

Benefiting from the accelerated computation, we were able to explore the parameter space without agonizing pain. We surprisingly found that the crack pattern of the necrotic core has little to do with the two-stage necrosis process of tumor cells. The central slice of the spheroid of non-swelling tumor cells shows almost identical microstructure patterns (Fig 5B). However, this system ends up with a slightly smaller size, fewer cells, and a higher overall necrotic debris center (Fig 4A).
density. This suggests that the swelling process of tumor cells could facilitate spheroid growth by pushing the viable cells toward the more oxygenated outer regions. Fig 5C shows the spheroid of tumor cells with the cell-cell adhesion suppressed. The cell-cell adhesion factor $C_{\text{cca}}$ is decreased to one-fourth of the reference value. The weak adhesion discourages gathering necrotic cells, leading to a more scattered distribution of smaller necrotic cell clusters with more minor interleaving cracks. Fig 5D is a spheroid with the cell-cell adhesion enhanced by quadrupling $C_{\text{cca}}$. The strong adhesive force helps form the massive necrotic debris clusters and promotes a significantly higher cell density that intensifies the oxygen competition between tumor cells and ultimately hinders tumor growth. Such pattern differences in necrotic core microstructures that emerged in the simulation are also observed in in vitro experiments [33], as shown in Fig 6. Two spheroids of the same melanoma cell line (A2508) form clustered (Fig 6A) and scattered (Fig 6A) necrotic cores, respectively. The exact differences in cell treatment and mechanisms of pattern formation are not described in the original literature. However, our simulations hint that cell-cell adhesion could be an important factor that dramatically affects the spheroid morphology, especially the necrotic core microstructure.

Simulations have the potential to depict a causal and mechanistic path from certain microscopic cell properties to the development of qualitative macroscopic morphology and provide

Fig 5. HDS simulation with altered phenotype. The 60-um thick central slices of simulated spheroids with various cellular mechanical properties. All the spheroids start with a small cluster of 2347 randomly placed cells, and the cultivation duration is 450 hours. (a) The reference spheroid ends up with 0.9 million cells and a diameter of 1.87 mm. (b) Spheroid of tumor cells with no swelling during early necrosis, with 0.9 million cells and a diameter of 1.8 mm. (c) Spheroid of tumor cells with the cell-cell adhesion suppressed, with 1.0 million cells and a diameter of 1.97 mm. (d) Spheroid of tumor cells with the cell-cell adhesion enhanced, with 0.66 million cells and a diameter of 1.53 mm.

Fig 6. Melanoma cell line spheroids. Two spheroids of the same melanoma cell line (A2508) show distinct pattern differences in necrotic core microstructures due to differences in cell treatment. Images are adapted from [33], and treatment details are not mentioned in the original literature. (a) A pimonidazole stained spheroid. (b) A hematoxylin and eosin stained spheroid. Adapted from [33] with permission.
insights into real-world phenomena. However, exploring the parameter space and improving the models iteratively can be very time-consuming. Gell could help these studies by dramatically increasing the computational speed.

Besides the baseline HDS simulation task, further comparisons of simulation performance with varying initialized cell numbers (Fig 7A) and domain sizes (Fig 7B) suggest that Gell is consistently around two orders of magnitude faster than multi-thread PhysiCell on the personal computer. For serial PhysiCell using only one thread, Gell can be almost 400x faster (Fig 7C). The default simulation setting for the performance comparison contains one million $25 \times 25 \times 25 \, \mu m$ isotropic voxels and one million living cells. Cells are randomly initialized in a sphere with a specific cell density. The acceleration ratios of these tests differ from the whole HDS simulation because the HDS simulation has a more heterogeneous cell distribution and is closer to the equilibrium states after prolonged mechanical interactions.

Additionally, Gell can complete the simulation with its maximum memory footprint of only one-tenth of PhysiCell without sacrificing accuracy and system complexity (Table 4).

**Table 4. Memory footprint comparison.**

|                                | PhysiCell | Gell  |
|--------------------------------|-----------|-------|
| Memory Footprint of HDS Simulation (MB) | 6360      | 500   |
| Memory Footprint per Additional Million Cells (MB) | 5384.2   | 118.0 |
| Memory Footprint per Additional Million Voxels (MB) | 742.8     | 15.38 |

Memory footprint comparison of Gell and PhysiCell per additional million cells and per additional million domain voxels in the HDS simulation task. Our model is able to simulate the problem with a much lower memory occupation.

https://doi.org/10.1371/journal.pone.0288721.t004
Ductal carcinoma in situ

Ductal carcinoma in situ (DCIS) is non-invasive breast cancer that grows within the lumens of the mammary duct [34]. DCIS itself is not hard to treat, but as a precursor to invasive ductal carcinoma with a high incidence rate (26.6 per 100000 women [35]), its development and progression raise the interest of many modelers [36–40]. Following the work of Paul Macklin (30), we created a hybrid DCIS simulation example to illustrate Gell with a more complex task further. A cluster of tumor cells is placed at the dead-end of a fixed single-opening duct. The movement of tumor cells is confined within the tube lumen, and the oxygen is supplied through diffusion across the duct wall into the lumen. Breast ducts have a typical radius ranging from 100 μm to 200 μm [38]; therefore, we simulated three growth scenarios of ductal carcinomas in situ with the respective duct radius of 100 μm, 150 μm, and 200 μm.

With the same cell model as in the HDS simulation and a Dirichlet boundary condition of 7.2 mmHg oxygen concentration applied to the duct surface, the average rate of DCIS advance for ducts of radius 100, 150, and 200 μm is 77.0, 32.7, 19.0 μm/day, respectively. Our simulation results in Fig 8 show a linear growth speed of DCIS, and the tumor advance rate has an inverse relationship with the duct radius, which agrees with the clinical observations and other computational studies in 2D [36] or 3D [41].

Performance testing

Individual module time cost. We first evaluated our simulator with a randomly initiated spherical cell cluster with one million cells for performance assessment. The simulation domain contains one million 25 μm-long isotropic voxels. The time cost of each module is listed in Table 5. With the entire calculation paralleled on GPU, Gell maximizes computational efficiency in all the simulation modules. Meanwhile, unnecessary data transfer between CPU and GPU during the simulation is eliminated, ensuring a low delay between modules.

In the table, the cell-sorting-related process appears to be slow, but in practice, its impact on the overall computation is small. Firstly, the cell sorting module is less invoked than many other modules. The simulation faces a multiscale problem. The cell phase is updated every few simulation minutes, cell motion is updated every few seconds, and the diffusion-reaction of oxygen is updated more than once per second. In this case, the simulation is approximately equally dominated by the cell-motion-related modules and the diffusion-reaction model. Secondly, the sorting process accelerates the rest of the calculations. Take the oxygen consumption model as an example. This kernel is a part of the reaction-diffusion module that calculates...
each cell’s oxygen consumption rate and adds the value to the total consumption rate of each corresponding voxel. This kernel works with both unsorted and sorted cell structures. Changing from unsorted random memory access to sorted contiguous memory access, this module’s time cost per invocation is reduced from 0.345 ms to 0.170 ms by 51%. The force calculation module is expected to benefit most from the high data fetch efficiency of the voxel sorting method. Because force calculation does not work with unsorted data structures, the speed comparison cannot be performed. Nevertheless, the longer time spent on cell sorting benefits the overall computation and is a worthy investment.

**Performance scaling**

Unaffordable memory occupation is another potential barrier for large-scale cell-based simulations. Memory efficiency is another design goal to make Gell a suitable software for large-scale simulations on widely available devices. The GPU memory footprint of the HDS simulation with one million voxels and one million cells is limited to 500 MB, and the memory occupation increases with the cell number and domain size is linear and slow, as shown in Table 5. This enables Gell to fit extremely large-scale problems into a modern personal computer easily.

We tested Gell’s ability to handle ultra-large-scale problems with a hypothetical ultra-large spheroid model. In reality, the size of hanging droplets is diffusion constrained. Larger in vivo tumors inevitably involve angiogenesis and supporting tumor vasculature. However, angiogenesis and oxygen transportation simulations are beyond the scope of the current work. Instead, we simulate a series of hypothetical huge hanging droplet development problems in a huge domain with varying initial cell numbers, each for one hour. As shown in Fig 9, Gell has a linear computational cost scaling with the cell number. Simultaneously, the Gell GPU memory footprint peaked at 4392 MB, showing high efficiency in memory and computational source usage. This linear time complexity and low memory occupation demonstrate that Gell can handle potential ultra-large-scale problems.

Time and memory cost for one hour’s mechano-biological process simulation with varying cell numbers. The domain contains 250×250×250 25 μm-long isotropic voxels.

**Discussion**

Cell simulation is driven by the need to model larger, more complex digital tumors parallel to human tumors containing billions of cells. The modeling accuracy of the biology and
computational capacities has limited the simulation of such complex systems. Existing tools such as Biocellion [16] for such large-scale simulations are close-sourced and require expensive CPU clusters. Open source tools, including PhysiCell [15] and BioDynaMo [17] require high-end CPU clusters to perform computation in practical time. In theory, GPU is well suited to manage the large parallel components of cell simulation. However, due to the differences between GPU and CPU architecture, a direct translation of a CPU-based cell simulation code to a GPU-based one is not straightforward and can be inefficient. Specifically, global cell-cell interactions have a time complexity of $O(N^2)$. The nonlinear computational cost severely limits the number of cells that can be modeled and, subsequently, the size and complexity of the model. We need to maintain a neighboring cell list to overcome the challenge, which is difficult for GPU memory due to its dynamic nature. Besides this challenge, an incomplete translation of computation from CPU to GPU requiring frequent data transfer between them can also be rate limiting. As a result, the threshold of performing cell simulation to a size that is relevant to the small tissue scale remains out of reach for many biological researchers even with modern GPU architecture.

To overcome these challenges, we employed several novel computational techniques to fully leverage the GPU architecture, improve simulation performance, and minimize memory usage.

As the computational cost of cell-cell interaction rises quadratically with the cell number, we developed a GPU-friendly voxel sorting method that handles the short-range cell-cell interaction modeling and improves the simulation’s memory access efficiency. We also implemented a fully GPU-based LOD solver for the spatiotemporal variation of diffusive substance distribution in extracellular water. As a result, we optimized the evolution algorithm to achieve linear computational complexity $O(N)$ while minimizing the memory footprint. In the numerical implementation, we fully exploited the parallel architecture of modern GPU and different types of GPU memory for high computational speed and low memory access overhead. The computation is nearly 100% on GPU, thus avoiding slow data transfer between CPU and GPU memories.

Our GPU implementation significantly outperformed CPU methods, leading to almost 400X speedup over the single-thread version of the well-established CPU simulator on a personal computer. The acceleration is the highest among existing GPU-powered simulators. The
acceleration, in combination with the low memory footprint, makes Gell readily available to biology researchers. The easy-to-access platform would facilitate the fast prototyping of hybrid models and hypothesis testing for large-scale problems.

As a future research direction, Gell can be scaled to multiple GPUs for larger problems on the order of $10^9$ cells. As previously alluded, simulation of tumor vasculature and other scaffolding cells such as the stromal and immune cells, would be necessary for a biologically relevant model. Besides our effort to incorporate these extremely complex biological processes for digital tumor twins, we support our peers to join the effort by providing and updating Gell as a user-friendly open-source tool. The source code can be found at https://github.com/PhantomOtter/Gell.

**Conclusion**

Large-scale cell simulations are valuable for hypothesis generation and testing experimental parameters. However, the high computational cost of the CPU-based simulation has limited the simulation size and practicality. In the current study, we describe a novel GPU cell simulation platform Gell to fully leverage the highly parallel nature of GPU. For the first time, we demonstrated a GPU-friendly voxel sorting method that reduced the quadratic cell-cell interaction computational complexity to be linear. The full GPU implementation avoided unnecessary CPU-GPU data transfer overhead. As a result, Gell achieved a 400X acceleration and 1–2 order of magnitude reduction in the memory footprint compared to a state-of-the-art CPU cell simulation platform, PhysicCell, for the same hanging droplet tumor spheroid and ductal carcinoma in situ simulation tasks.

**Author Contributions**

**Conceptualization:** Ke Sheng.

**Data curation:** Jiayi Du.

**Formal analysis:** Jiayi Du.

**Funding acquisition:** Ke Sheng.

**Methodology:** Jiayi Du, Yu Zhou, Lihua Jin, Ke Sheng.

**Software:** Jiayi Du.

**Supervision:** Ke Sheng.

**Visualization:** Jiayi Du, Yu Zhou.

**Writing – original draft:** Jiayi Du.

**Writing – review & editing:** Yu Zhou, Lihua Jin, Ke Sheng.

**References**

1. Lowengrub JS, Frieboes HB, Jin F, Chuang YL, Li X, Macklin P, et al. Nonlinear modelling of cancer: bridging the gap between cells and tumours. Nonlinearity. 2009 Dec; 23(1):R1–91.

2. Metzcar J, Wang Y, Heiland R, Macklin P. A Review of Cell-Based Computational Modeling in Cancer Biology. JCO Clin Cancer Inform. 2019 Feb 4;(3):1–13. https://doi.org/10.1200/CCI.18.00069 PMID: 30715927

3. Karolak A, Markov DA, McCawley LJ, Rejnia K. Towards personalized computational oncology: from spatial models of tumour spheroids, to organoids, to tissues. J R Soc Interface. 2018 Jan 31; 15 (138):20170703. https://doi.org/10.1098/rsif.2017.0703 PMID: 29367239
4. Grant MR, Mostov KE, Tlsty TD, Hunt CA. Simulating Properties of In Vitro Epithelial Cell Morphogenesis. PLoS Comput Biol [Internet]. 2006 Oct [cited 2020 May 20]; 2(10). Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1592363/ https://doi.org/10.1371/journal.pcbi.0020129 PMID: 17029556

5. Tang J, Enderling H, Becker-Weimann S, Pham C, Polyzos A, Chen CY, et al. Phenotypic transition maps of 3D breast acini obtained by imaging-guided agent-based modeling. Integr Biol. 2011 Apr 1; 3(4):408–21. https://doi.org/10.1039/c0ib00092b PMID: 21373705

6. Poplawski NJ, Shirinifard A, Agero U, Gens JS, Swat M, Glazier JA. Front Instabilities and Invasiveness of Simulated 3D Avascular Tumors. PLoS ONE [Internet]. 2010 May 26 [cited 2020 May 20]; 5(5). Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2877086/ https://doi.org/10.1371/journal.pone.0010641 PMID: 20520818

7. Perfahl H, Byrne HM, Chen T, Estrella V, Alarcón T, Lapin A, et al. Multiscale Modelling of Vascular Tumour Growth in 3D: The Roles of Domain Size and Boundary Conditions. PLoS ONE [Internet]. 2011 Apr 13 [cited 2020 May 21]; 6(4). Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3076378/ https://doi.org/10.1371/journal.pone.0014790 PMID: 21533234

8. Rejniak KA, Estrella V, Chen T, Cohen AS, Lloyd MC, Morse DL. The Role of Tumor Tissue Architecture in Treatment Penetration and Efficacy: An Integrative Study. Front Oncol [Internet]. 2013 May 10 [cited 2020 May 21]; 3. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3650652/ https://doi.org/10.3389/fonc.2013.00111 PMID: 23717812

9. Scott JG, Fletcher AG, Anderson ARA, Maini PK. Spatial Metrics of Tumour Vascular Organization Predict Radiation Efficacy in a Computational Model. PLoS Comput Biol [Internet]. 2016 Jan 22 [cited 2020 May 21]; 12(1). Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4723304/ https://doi.org/10.1371/journal.pcbi.1004712 PMID: 26800503

10. Lundin T. Cellular Automaton Modeling of Biological Pattern Formation: Characterization, Applications, and Analysis. Authors: Andreas Deutsch and Sabine Dommann, Birkhäuser, 2005, XXVI, 334 p., 131 illus., Hardcover. ISBN:0-8176-4281-1, List Price: $89.95. Genet Program Evolvable Mach. 2007 Mar 1; 8(1):105–6.

11. Chopard B, Ouared R, Deutsch A, Hatzikirou H, Wolf-Gladrow D. Lattice-Gas Cellular Automata Models for Biology: From Fluids to Cells. Acta Biotheor. 2010; 58(4):329–40. https://doi.org/10.1007/s10441-010-9118-5 PMID: 20711745

12. Graner F, Glazier JA. Simulation of biological cell sorting using a two-dimensional extended Potts model. Phys Rev Lett. 1992 Sep 28; 69(13):2013–6. https://doi.org/10.1103/PhysRevLett.69.2013 PMID: 10046374

13. Van Liedekerke P, Palm MM, Jagiella N, Drasdo D. Simulating tissue mechanics with agent-based models: concepts, perspectives and some novel results. Comput Part Mech. 2015 Dec 1; 2(4):401–44.

14. Alt S, Ganguly P, Salbreux G. Vertex models: from cell mechanics to tissue morphogenesis. Philos Trans R Soc B Biol Sci. 2017 May 19; 372(1720):20150520. https://doi.org/10.1098/rstb.2015.0520 PMID: 28348254

15. Ghaffarizadeh A, Heiland R, Friedman SH, Mumenthaler SM, Macklin P. PhysiCell: An open source physics-based cell simulator for 3-D multicellular systems. PLoS Comput Biol [Internet]. 2018 Feb 23 [cited 2020 May 7]; 14(2). Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5841829/ https://doi.org/10.1371/journal.pcbi.1005991 PMID: 29474446

16. Kang S, Kahan S, McDermott J, Flann N, Shmulevich I. Biocellion: accelerating computer simulation of multicellular biological system models. Bioinformatics. 2014 Nov 1; 30(21):3101–8. https://doi.org/10.1093/bioinformatics/btu498 PMID: 25064572

17. Breitwieser L, Hesam A, de Montigny J, Vavourakis V, Iosif A, Jennings J, et al. BioDynaMo: a general platform for scalable agent-based simulation. ArXiv200606775 Cs [Internet]. 2021 Feb 5 [cited 2022 Jan 6]; Available from: http://arxiv.org/abs/2006.06775

18. Germann P, Marin-Riera M, Sharpe J. yaja: GPU-Powered Spheroid Models for Mesenchyme and Epithelium. Cell Syst. 2019: 8(3):261–266.e3. https://doi.org/10.1016/j.cels.2019.02.007 PMID: 30904379

19. Mirans GR, Arthurs CJ, Bernabeo MO, Bordas R, Cooper J, Corrias A, et al. Chaste: An Open Source C++ Library for Computational Physiology and Biology. PLOS Comput Biol. 2013 Mar 14; 9(3): e1002970. https://doi.org/10.1371/journal.pcbi.1002970 PMID: 23516352

20. Mathias S, Coulier A, Hellander A. CBMOS: a GPU-enabled Python framework for the numerical study of center-based models [Internet]. 2021 May [cited 2022 Jan 6] p. 2021.05.06.442893. Available from: https://www.biorxiv.org/content/10.1101/2021.05.06.442893v2

21. Hesam A, Breitwieser L, Rademakers F, Al-Ars Z. GPU Acceleration of 3D Agent-Based Biological Simulations. 2021 IEEE Int Parallel Distrib Process Symp Workshop IPDPSW. 2021 Jun;201–7.

22. Breitwieser L, Hesam A, de Montigny J, Vavourakis V, Iosif A, Jennings J, et al. BioDynaMo: a modular platform for high-performance agent-based simulation. Bioinformatics. 2022; 38(2):453–60. https://doi.org/10.1093/bioinformatics/btab649 PMID: 34529036
23. Baghban R, Roshangar L, Jahanban-Esfahlan R, Seidi K, Ebrahimi-Kalan A, Jaymard M, et al. Tumor microenvironment complexity and therapeutic implications at a glance. Cell Commun Signal. 2020 Apr 7; 18(1):59. https://doi.org/10.1186/s12964-020-0530-4 PMID: 32264958

24. Ghaffarizadeh A, Friedman SH, Macklin P. BioFVM: an efficient, parallelized diffusive transport solver for 3-D biological simulations. Bioinformatics. 2016 Apr 15; 32(8):1256–8. https://doi.org/10.1093/bioinformatics/btv730 PMID: 26656933

25. Povitsky A, Morris PJ. A Higher-Order Compact Method in Space and Time Based on Parallel Implementation of the Thomas Algorithm. J Comput Phys. 2000 Jun 10; 161(1):182–203.

26. Satish N, Harris M, Garland M. Designing efficient sorting algorithms for manycore GPUs. In: 2009 IEEE International Symposium on Parallel Distributed Processing. 2009. p. 1–10.

27. Z-order curve. In: Wikipedia [Internet]. 2022 [cited 2022 Jul 7]. Available from: https://en.wikipedia.org/w/index.php?title=Z-order_curve&oldid=1092826362

28. Kimlin LC, Casagrande G, Virador VM. In vitro three-dimensional (3D) models in cancer research: An update. Mol Carcinog. 2013; 52(3):167–82. https://doi.org/10.1002/mc.21844 PMID: 22162252

29. Zanoni M, Piccinini F, Arienti C, Zamagni A, Santi S, Polico R, et al. 3D tumor spheroid models for in vitro therapeutic screening: a systematic approach to enhance the biological relevance of data obtained. Sci Rep. 2016 Jan 11; 6(1):19103. https://doi.org/10.1038/srep19103 PMID: 26752500

30. Bull JA, Mech F, Quaiser T, Waters SL, Byrne HM. Mathematical modelling reveals cellular dynamics within tumour spheroids. PLOS Comput Biol. 2020 Aug 18; 16(8):e1007961. https://doi.org/10.1371/journal.pcbi.1007961 PMID: 32810174

31. Dorie MJ, Kallman RF, Rapacchietta DF, Van Antwerp D, Huang YR. Migration and internalization of cells and polystyrene microspheres in tumor cell spheroids. Exp Cell Res. 1982 Sep 1; 141(1):201–9.

32. Mao X, McManaway S, Jaiswal JK, Patel PB, Wilson WR, Hicks KO, et al. An agent-based model for drug-radiation interactions in the tumour microenvironment: Hypoxia-activated prodrug SN30000 in multicellular tumour spheroids. PLOS Comput Biol. 2018 Oct 24; 14(10):e1006469. https://doi.org/10.1371/journal.pcbi.1006469 PMID: 30356233

33. Doctor A, Seftert V, Ulrich M, Hauser S, Pietzsch J. Three-Dimensional Cell Culture Systems in Radio-pharmaceutical Cancer Research. Cancers. 2020 Oct; 12(10):2765. https://doi.org/10.3390/cancers12102765 PMID: 32993034

34. Burststein HJ, Polyak K, Wong JS, Lester SC, Kaelin CM. Ductal Carcinoma in Situ of the Breast. N Engl J Med. 2004 Apr 1; 350(14):1430–41. https://doi.org/10.1056/NEJMra031301 PMID: 15070793

35. Grimm LJ, Rahbar H, Abdelmalak M, Hall AH, Ryser MD. Ductal Carcinoma in Situ: State-of-the-Art Review. Radiology. 2022 Feb; 302(2):246–55. https://doi.org/10.1148/ radiol.211839 PMID: 34931856

36. Macklin P, Edgerton ME, Thompson AM, Cristini V. Patient-calibrated agent-based modelling of ductal carcinoma in situ (DCIS): From microscopic measurements to macroscopic predictions of clinical progression. J Theor Biol. 2012 May 21; 301:122–40. https://doi.org/10.1016/j.jtbi.2012.02.002 PMID: 22342935

37. Franks SJ, Byrne HM, Underwood JCE, Lewis CE. Biological inferences from a mathematical model of comedo ductal carcinoma in situ of the breast. J Theor Biol. 2005 Feb 21; 232(4):523–43. https://doi.org/10.1016/j.jtbi.2004.08.032 PMID: 15588633

38. Franks SJ, Byrne HM, King JR, Underwood JCE, Lewis CE. Modelling the early growth of ductal carcinoma in situ of the breast. J Math Biol. 2003 Nov 1; 47(5):424–52. https://doi.org/10.1007/s00285-003-0214-x PMID: 14605857

39. Rejniak KA, Dillon RH. A single cell-based model of the ductal tumour microarchitecture. Comput Math Methods Med. 2007 Mar 1; 8(1):51–69.

40. Norton KA, Wininger M, Bhanot G, Ganesan S, Barnard N, Shribot T. A 2D mechanistic model of breast ductal carcinoma in situ (DCIS) morphology and progression. J Theor Biol. 2010 Apr 21; 263 (4):393–406. https://doi.org/10.1016/j.jtbi.2009.11.024 PMID: 20066203

41. Butner JD, Fuentes D, Ozpolat B, Calif GA, Zhou X, Lowengrub J, et al. A Multiscale Agent-Based Model of Ductal Carcinoma In Situ. IEEE Trans Biomed Eng. 2020 May; 67(6):1450–61. https://doi.org/10.1109/TBME.2019.2938485 PMID: 31603768