Heart and Blood Flow Simulation using Position Based Dynamics

Thanandon IMAROMKUL

Graduate School of Library, Information and Media Studies

University of Tsukuba

September 2019

Contents

List of Figures

Chapter 1

Introduction

According to the report from World Health Organization or WHO, despite the growth of medical technology nowadays, CVDs are the number one cause of death globally, approximately 17.9 million people died in 2016 and 85% of the deaths were due to heart attacks and strokes [World Health Organization 2017]. Another recent statistic report form the American Heart Association had shown that in America, people die from CVDs roughly every 42 seconds [Mozaffarian et al. 2016].

Following the WHO definition, "Cardiovascular disease refers to the group of disorders in heart and blood vessels" [American Heart Association 2018; Pruthi 2018; World Health Organization 2017]. These diseases can lead to a heart attack or stroke if left untreated. There are several common cases of CVDs. For example, a peripheral vascular disease which refers to when a blood vessel becomes blocked, narrowed or spasmed [American Heart Association 2016a]. Rheumatic heart disease is when the heart valves were damaged causing the blood to flow backward or blocked [Rheumatic Heart Disease Australia 2019]. Cardiomyopathy refers to the conditions that commonly affect the heart muscle, for example, hypertrophic or dilation. These heart muscle abnormality will interrupt blood flow and can easily cause heart failure [American Heart Association 2016b]. Furthermore, [Doost et al. 2016] pointed out that CVDs are mainly caused by the malfunctions or defects discovered in the left ventricle chamber of the heart and it is vital for the practitioner to give it the highest inspection priority.

Recognizing the conditions requires an extensive amount of knowledge and experience from the doctors. Also, in the case of American, the need for the new generations of cardiologists is substantially high because of the growing demands and the geographic maldistribution of the services in the underserved area [Narang et al. 2016]. In the same manner with the rest of the world, it is crucial to rapidly train new medical trainees and students to become professionals as fast and as efficient as possible [Townsend et al. 2016]. Heart muscle and blood flow analysis is vital for doctors and cardiologists. Even though there are tools for such analytical operations, most of those tools are aiming for advanced users which makes it challenging for early in-training students to access.

Hence, an educational oriented tool that is moderately easy to use will definitely help the students grasp the medical concepts significantly faster. To investigate the matter, a one-hour interview was arranged in order to consult with a doctor professor about a tool that can help the medical students improve their understanding and strengthen their learning capability. The doctor is currently teaching and positioning in a famous medical

school in Bangkok, Thailand. During the interview, the professor mentioned that it is challenging for the students to grasp the physiology concepts with just the traditional 2D diagrams from the textbooks. 3D materials are extremely useful and are proven to help them understand the lessons better. However, MRI and CT-scan are too expensive and technologically challenging to be integrated into the classes and even if the students have access to the 3D models, there are no movement or flow to be observed. Hence, a tool that can offer the benefits of both worlds would undoubtedly assist the students in their learning experiences.

Therefore, the purpose of this research is to explore the possibility of developing such a tool. The system is required to be

- **Easy to use**: Medical students must be able to intuitively use the software with ease.
- **Interactive**: The software must respond to the students' configuration in real time.
- **Moderately accurate**: Since education is the aim of this system, the results must be realistic enough to not mislead the students.

Hence the implementation technique must be considered accordingly. The rendering in this environment is composed of two primary systems (1) elastic model and (2) fluid flow simulation. There are numerous techniques that can be used for both domains, but the most suitable method for this particular setting is "Position Based Dynamics" or PBD due to its stability, controllability, speedy nature and above all, PBD can unify both systems into a singular module. More details would be discussed in the next section.

The contribution of this research is to investigate the feasibility of using this modern simulation technology, PBD, to create an educational heart and blood simulation system for medical students. The rest of this paper is organized as follows. Related works and PBD are reviewed in section 2 and 3. The proposed method and the results are presented in sections 4 and 5. Finally, expert interview and discussions will be drawn in section 6 and 7.

Chapter 2

Related Work

Heart and blood simulation is one of the active fields in Computer Graphics and continues to gain more popularity. Still, most of the researches are either focusing on the heart model simulation or the blood flow simulation, research that emphasis on combining the two are yet to be found.

Heart model simulation essentially falls under the domain of elastic simulation in the field of Computer Graphic. Elastic or deformable objects are referring to objects that will deform when a certain amount of force was applied but can return itself back to its original state. Numerous techniques had been developed for this particular problem, but perhaps the most dominating techniques are Finite Element Method (FEM), and Finite Volume Method (FVM) as they were used in [Baillargeon et al. 2014; Berberoğlu et al. 2014; Gonzales et al. 2013; Klepach et al. 2012]. Though their purposes were different, all of the cited papers were using FEM and/or FVM to construct a deformable human heart model to investigate the cardiac structural properties. [Baillargeon et al. 2014] aims to create a heart model with all four chambers and valves with electrical and mechanical simulation. [Gonzales et al. 2013] offer an alternative solution to map cubic Hermite finite elements and improve the accuracy of the human atrial model. [Berberoğlu et al. 2014] coupled the heart model with electromechanics to detect possible cardiac dysfunctions while [Klepach et al. 2012] attempts to inspect cardiac growth of left ventricle in relation to myocardial infarction (MI) and surgical ventricle restoration. Since FEM and FVM were originally created to solve mechanical engineering problems, both of them can provide excellent results, however, there are several drawbacks that made them unsuitable for this research. The methods require a lot of configurations and parameters settings from the user in order to influence the simulation and also requires a certain amount of time to compute due to an extensive set of physics equations.

Additionally, [Nakashima et al. 2016] proposed an interactive method to construct a deformable heart model. The research offers a solution to simulate the model of a human heart based on an MRI scan of the patient. The constructed model represented the patient's heart very accurately and the system allows the user to interact with the model such as tissue cutting and pinning with ease. Another research [Bender et al. 2015] offers an alternative surgical cutting system for human muscle and tissue. The solution utilizes a contemporary graphic simulation technique called "Position Based Dynamics" or PBD

in order to provide a fast and accurate simulation of electrosurgery. The solution in this paper offers not only the simulation of the deformable tissues but also the simulation of the surgical cutting in an interactive environment. However, these researches still haven't included the integration of blood flow simulation yet.

Blood flow simulation falls under fluid simulation in Computer Graphics. In this area Computational Fluid Dynamics (CFD) seems to be the most respected methods. In the case of CFD, the produced results are really accurate which is why most researches that utilized CFD aim for deployment in the actual medical diagnosis and prognosis. A review paper [Doost et al. 2016] clearly addressed the state of art of heart blood flow simulation. The paper reviewed over 31 highly accurate blood flow simulation techniques that centered around CFD. Almost all of them use an MRI scan while the rest use CT-scan. Despite that, the paper informed that the methods are still not clinically applicable due to the limitations on computational time and the fact that some important phenomenon cannot be integrated into the simulation still. [Selmi et al. 2019] also utilized CFD to simulate the pulsatile blood flow to compare the result in different types of arterial fenestrations while [Tyfa et al. 2018] used the technique to investigate the blood flow in the human aorta.

While most CFD methods offers an accurate but time-consuming solution, with a little tradeoff, Smoothed Particle Hydrodynamics or SPH offers a significantly faster one. The vital aspect that makes SPH quicker than other CFD methods is that SPH is a meshless approach, though the precision might not be as high as the mesh-based approaches [Monaghan 1992; Springel 2010]. [Caballero et al. 2017] used SPH with MRI model to simulate the left ventricular blood flow while [Palyanov et al. 2016] reviewed the application of SPH in simulating the mechanism of biological tissue. It is also notable that all of the mentioned researches presented their result in blood current, blood velocity chart rather than in fluid particles and not interactive.

Position-based Dynamics or (PBD) is a recently created method [Bender et al. 2014]. This method ignores the force and acceleration computational steps which hastened the process but still yield reasonably accurate results. Moreover, the method requires very little configuration and parameters setting form the user to manipulate the simulation and it provides an efficient way to simultaneously combine multiple types of simulation, e.g., deformable objects and fluid simulation together. Additionally, [Müller et al. 2004] had proposed an interactive blood simulation using the combination of SPH and PBD which results in a very interesting and visually pleasing result in 3D. For these reasons, PBD is the most suitable method to be implemented in this particular system.

Chapter 3

Position Based Dynamics

As mentioned in the previous section, "Position Based Dynamics" or PBD is a recently developed graphics simulation technique that is wildly known and praised for its reasonably fast and accurate result [Bender et al. 2015; Macklin et al. 2014; Müller et al. 2003, 2007]

Traditionally, computer graphic simulations are done by using the physics equations to initially calculate the force, acceleration, and velocity of the particle, then use the computed velocity with the integration of time to finally calculate the position corresponding to that particular time. The computational cost can exponentially scale with the level of complexity from the simulation. On the other hand, PBD tries to surpass such limitation by directly compute velocity and position without calculating the force and acceleration. This reduces the computational steps in half and can assuredly accelerated the simulation time.

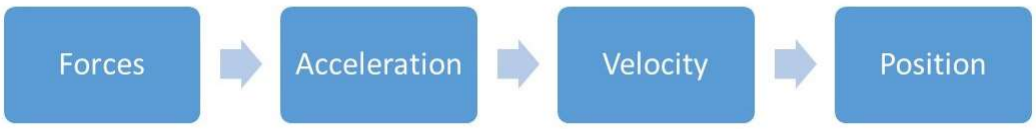


Figure 3.1: Traditional physics-based simulation

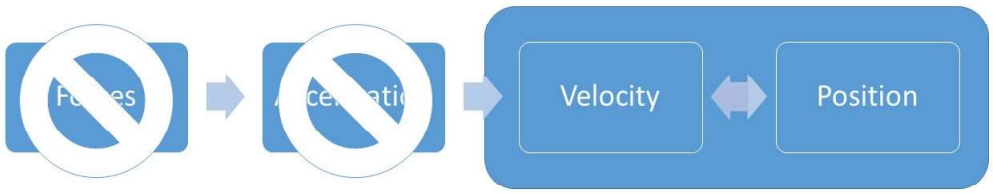


Figure 3.2: Position-based dynamics simulation

In addition, PBD can successfully abbreviate those steps by handling the law of physics implicitly. Instead of directly calculate the force and acceleration, PBD makes sure that the laws of physics are met by adopting a programming technique called "Constraint-Based Programming". The core concept of constraint-based programming is to create a set of constraints that during the operation, the computer has to satisfy. The details of such constraints will be discussed in later sections. Consequently, the PBD technique can indirectly

guarantee that the simulation will still be accurate without calculating the force and acceleration. Fig.3.1 shows the process flow of the traditional physics-based computer graphic simulation and Fig.3.2 shows the process flow of Position Based Dynamics simulation. With these interesting benefits of PBD, this research will be utilizing the technique.

3.1 Constraints

Constraints in PBD are restrictions that typically imitate the laws of physics. They can be defined as an function of positions, e.g., $C(\mathbf{x})$. Traditionally, $C(\mathbf{x})$ would be defined as either equality $C(\mathbf{x}) = 0$ or inequality $C(\mathbf{x}) \geq 0$ equations which refer to bilateral and unilateral constraints respectively. During the simulation, PBD needs to have a position solver and its goal is to find correction vector $\triangle\mathbf{x}$ that satisfy either $C(\mathbf{x} + \triangle\mathbf{x}) = 0$ or $C(\mathbf{x} + \triangle\mathbf{x}) \geq 0$ depends on the type of constraints defined. Additionally, in this specific simulation, the blood fluid will be represented by individual particles and the heart model will be represented by interconnected particles in order to employ the meshless SPH approach.

In this system, a non-linear Gauss-Seidel solver will be used [Bender et al. 2015]. Approximately, the constraint equation can be written as equation (3.1) by using a Taylor expansion. After solving equation (3.1) for $\triangle\mathbf{x}$, and considering the mass of each particle m_i, the correction vector that satisfy the constraints can be computed.

$$C(\mathbf{x} + \triangle\mathbf{x}) \approx C(\mathbf{x}) + \bigtriangledown C(\mathbf{x}) \cdot \triangle\mathbf{x} = 0 \tag{3.1}$$

$$\triangle\mathbf{x} = \lambda\mathbf{M}^{-1} \bigtriangledown (\mathbf{x}) \tag{3.2}$$

where $\mathbf{M}$ is the inverted mass matrix $\mathbf{M} = diag(m_1, \ldots, m_n)$. The correction vector for the particle i can be written as;

$$\triangle\mathbf{x}_i = -\lambda w_i \bigtriangledown_{x_i} C(\mathbf{x}) \tag{3.3}$$

where

$$\lambda = \frac{C(\mathbf{x})}{\sum_j w_j | \bigtriangledown_{x_j} C(\mathbf{x})|^2} \tag{3.4}$$

where $w_i = \frac{1}{m_i}$ is a weight of particle i. The equation can be reformulated using the mass matrix $\mathbf{M}$ as;

$$\lambda = \frac{C(\mathbf{x})}{\bigtriangledown C(\mathbf{x})^T \mathbf{M}^{-1} \bigtriangledown C(\mathbf{x})} \tag{3.5}$$

Finally, after the positions are corrected and velocities will be updated by using a timestep lenght $\triangle t$.

$$\triangle\mathbf{v} = \frac{\triangle\mathbf{x}}{\triangle t} \tag{3.6}$$

3.2 Implementation

Algorithm 1 shows the main process of carrying out PBD which was quoted from [Bender et al. 2015]. The first for loop is for initializing $\mathbf{x}_i$, $\mathbf{v}_i$ and weight w_i which are the old position, velocity and weight of the particle i. Lines (4)-(7) are for estimating the possible new position p_i and velocity v_i given the old position, old velocity, weight, external force, and timestep by using simple symplectic Euler integration. Please note that external force, in this case, is just the gravitational force (9.81 m/s^2). Later, collision constraints and external constraints are generated in line (8)-(9). Toward line(10)-(11), the rest of the constraints will be handled and the possible new positions will be corrected multiple times according to the number of iteration defined by the "solverIteration". Finally, in line (12)-(15), the new corrected positions and velocities will be updated.

Algorithm 1 Position-Based Simulation Pseudocode [Bender et al. 2015]

1: **for** all vertices i **do**
2: initialize $\mathbf{x}_i = \mathbf{x}_i^0$, $\mathbf{v}_i = \mathbf{v}_i^0$, $w_i = \frac{1}{m_i}$
3: **loop**
4: **for** all vertices i **do**
5: $\mathbf{v}_i \leftarrow \mathbf{v}_i + \triangle t \times w_i \times \mathbf{f}_{ext}(\mathbf{x_i})$
6: **for** all vertices i **do**
7: $\mathbf{p}_i \leftarrow \mathbf{x}_i + \triangle t \times \mathbf{v}_i$
8: **for** all vertices i **do**
9: genCollConstraints$(\mathbf{x}_i \rightarrow \mathbf{p}_i)$
10: **for** *solverIteration* **times do**
11: projectConstraints$(C_1, \ldots, C_{M+M_{Coll}}, \mathbf{p}_1, \ldots, \mathbf{p}_N)$
12: **for** all vertices i **do**
13: $\mathbf{v}_i \leftarrow \frac{(\mathbf{p}_i - \mathbf{x}_i)}{\triangle t}$
14: $\mathbf{x}_i \leftarrow \mathbf{p}_i$
15: velocityUpdate$(\mathbf{v}_i, \ldots, \mathbf{v}_N)$

Chapter 4

Proposed Method

In our method, there are 4 main components to be considered; (4.1) Heart model simulation, (4.2) Blood simulation, (4.3) Cardiac cycle simulation and (4.4) Heart and blood interaction simulation, which will all be discussed in this section.

Before advancing into the constraints, this section will explain the utilization of PBD in this research. Fig.4.1 shows the main process of the PBD implementation.

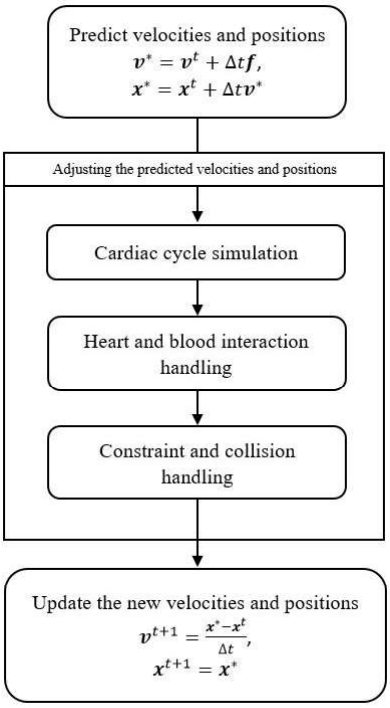


Figure 4.1: Algorithm flow

Initially, the system will try to predict the new position $\mathbf{x}^*$ based on the new velocity $\mathbf{v}^*$ which computed by adding the previous velocity $\mathbf{v}^t$ with the multiplication of the timestep Δt and external force f similarly described in the previous section. In the second step, the movement of the cardiac cycle will be simulated. The heart model will deform according to the cardiac rhythm. If $\mathbf{x}^*$ of any particle penetrates the model, it will be corrected during the second step namely, Heart and blood interaction handling. Thirdly, all collisions and constraints will be managed, $\mathbf{x}^*$ and $\mathbf{v}^*$ will again be updated accordingly. Finally, the

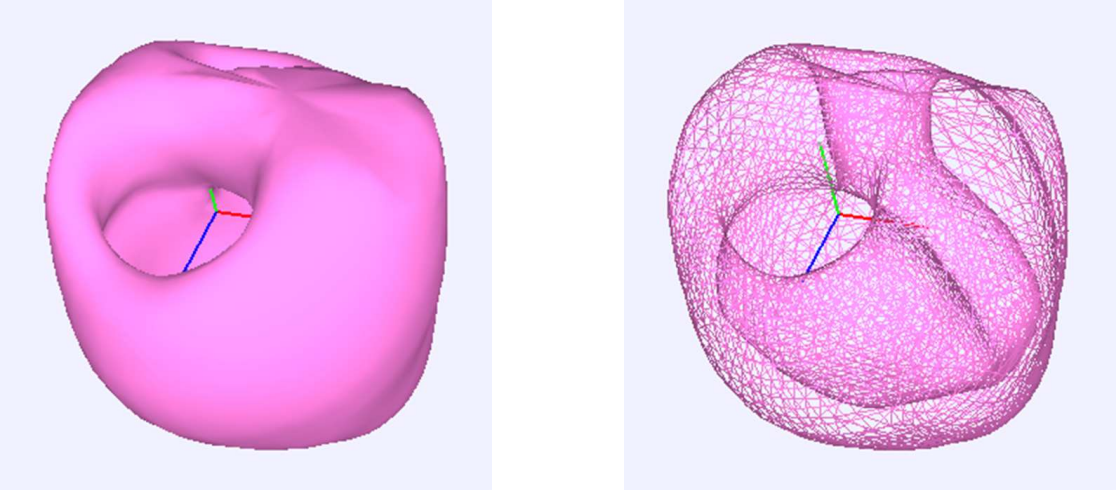
system stores the $\mathbf{x}^*$ and $\mathbf{v}^*$ as the current position $\mathbf{x}^{t+1}$ and velocity $\mathbf{v}^{t+1}$ of the timestep $t+1$.

These steps will be repeated for the whole simulation. In PBD, the animation properties of the models are controlled by employing constraints. Thus, designing them is a very crucial element of this research. The constraints are classified and explained in the next section.

4.1 Constraints for Heart model simulation

The deformable constraints will be applied to the heart model to achieve the soft and non-rigid property of the heart tissue. There are 2 main constraints for deformable objects that will be adopted in this method which are (1) Tetrahedron volume conservation constraints and (2) Triangle face area conservation constraints.

4.1.1 Tetrahedron volume conservation constraint



(a) Sample heart model texture

(b) Sample heart model tetrahedrons

Figure 4.2: Sample heart model

The heart model in this research will be comprised of tetrahedron meshes as shown in Fig. 4.2. During the active movement of the cardiac cycle, the positions of the mesh vertices have to be changed in order to create the deformation effect. For instance, in the case of contraction, the muscle will be contracted together in order to force the blood throughout the entire body and during the release phase, the muscle must be expanded to support the new volume of blood entering the heart.

At the same time, moving the position of the mesh vertices without any restriction can cause misrepresentation of the heart structure. In order to maintain the shape and structural properties of the heart, the volume of the mesh has to be controlled and preserved. Thus, the tetrahedron volume conservation constraint was used.

This constraint aims to conserve and maintain the volume of the mesh in the heart model to hold their structure. The equation is listed below;

$$C(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = \frac{1}{6}(\mathbf{x}_{2,1} \times \mathbf{x}_{3,1}) \cdot \mathbf{x}_{4,1} - V_0 \tag{4.1}$$

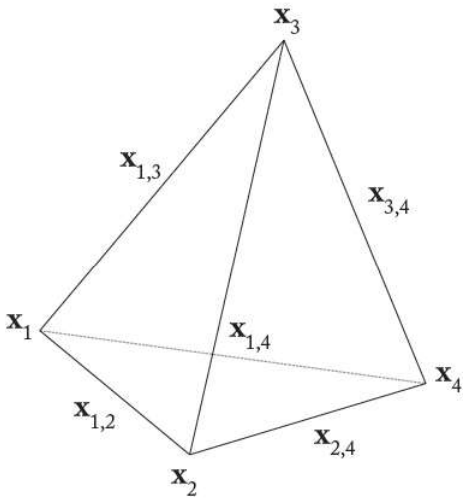


Figure 4.3: Tetrahedron vertices and edges

$$V_0 = \frac{1}{6}(\mathbf{x}_{rest_{2,1}} \times \mathbf{x}_{rest_{3,1}}) \cdot \mathbf{x}_{rest_{4,1}} \tag{4.2}$$

where $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$ are the four vertices or corners of the tetrahedron and $\mathbf{x}_{i,j}$ represent the edge vector for the vertex i and j (ex, $\mathbf{x}_{2,1} = \mathbf{x}_2 - \mathbf{x}_1$) as shown in Fig.4.3. V_0 is the volume of the mesh at rest state which can be calculated by using equation (4.2) where $\mathbf{x}_{rest_1}, \mathbf{x}_{rest_2}, \mathbf{x}_{rest_3}, \mathbf{x}_{rest_4}$ are the four vertices of the tetrahedron at rest state. This constraint is a bilateral constraint which means it will be satisfied when $C(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = 0$ [Macklin et al. 2014]. The position correction $\triangle\mathbf{x}$ for this constraints can be computed by substituting equation (4.1) into equation (3.3).

4.1.2 Triangle face area conservation constraints

Besides the mesh volume, the surface area of the triangles that are representing the mesh has to also be maintained. The constraint is listed below.

$$C(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = \frac{1}{2}|\mathbf{x}_{2,1} \times \mathbf{x}_{3,1}| - A_0 \tag{4.3}$$

$$A_0 = \frac{1}{2}|\mathbf{x}_{rest_{2,1}} \times \mathbf{x}_{rest_{3,1}}| \tag{4.4}$$

where $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ are the three vertices of the triangle and A_0 is the area of the triangle at rest state which can be calculate by using equation (4.4). This constraint is also a bilateral constraint [Bender et al. 2014] and the position correction is calculated by also substituting equation (4.3) into equation (3.3).

4.2 Constraints for blood simulation

This research uses particles to represent the blood fluid and adopts 3 main constraints including (1) Collision with environment constraint, (2) Collision with particle constraint, (3) Density constraints. Besides, this research also utilizes XSPH with an implicit solver to introduce the viscosity effect or the stickiness of the blood [Springel 2010].

4.2.1 Collision with environment constraint

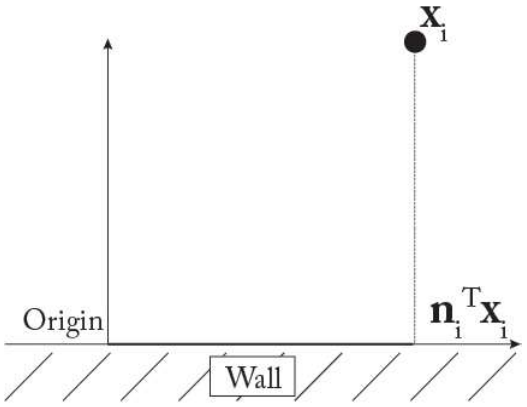


Figure 4.4: Tetrahedron vertices and edges

This constraint aims to maintain the distance between the fluid particles and the surrounding environment, including the heart muscle, to support the non-penetrable property. The equation is displayed below.

$$C(\mathbf{x}_i) = \mathbf{n}_i{}^T\mathbf{x}_i - d_{rest} = 0 \tag{4.5}$$

where $\mathbf{n}_i$ is the normal vector of the normalized contact plane for each particle i, $\mathbf{x}_i$ represent the position of the particle i in the coordinate system with its origin on the contact plane and d_{rest} is the distance to maintain at the rest state.

4.2.2 Collision with particle constraint

The purpose of this constraint is to make sure that the fluid particles will not penetrate each other by using a similar mechanism as the previous constraint.

$$C(\mathbf{x}_i, \mathbf{x}_j) = |\mathbf{x}_{ij}| - (\mathbf{r}_i + \mathbf{r}_j) \geq 0 \tag{4.6}$$

where $\mathbf{x}_{ij}$ is the distance between particle i and j. Also, r_i and r_j are the radii of the 2 particles.

4.2.3 Density Constraints

In order to accomplish realistic fluid behavior, density conversation is also a notable aspect to consider. Assuming that the mass of each particle m_i is constant, in that case, a constant density ρ represents the volume conservation. Hence, a constraint for the fluid density ρ can be introduced as;

$$C(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \frac{\rho_i}{\rho_0} - 1 \tag{4.7}$$

where ρ_0 is the density of the fluid at rest state and ρ_i is the volume at the particle i which defined as the sum of smooth kernels of the surrounded neighbor particles [Monaghan 1992].

$$\rho_i = \sum_j m_j W(\mathbf{x}_i - \mathbf{x}_j, h) \tag{4.8}$$

Then, in order to solve these constraints derivative of equation (4.8) is needed and can be calculated by using the gradient of SPH kernel.

$$\nabla_{\mathbf{x}_k} C_i = \frac{1}{\rho_0} \begin{cases} \sum_j \nabla_{\mathbf{x}_k} W(\mathbf{x}_i - \mathbf{x}_j, h) & \text{if } k = i \\ -\nabla_{\mathbf{x}_k} W(\mathbf{x}_i - \mathbf{x}_j, h) & \text{if } k = j \end{cases} \tag{4.9}$$

Finally, the correction position $\triangle \mathbf{x}_i$ for particle i can be calculated by;

$$\triangle \mathbf{x}_i = \frac{1}{\rho_0} \sum_j (\lambda_i + \lambda_j) \nabla W(\mathbf{x}_i - \mathbf{x}_j, h) \tag{4.10}$$

4.2.4 Viscosity acceleration (XSPH)

Due to the fact that blood is thicker than normal fluid, viscosity is needed to be applied in order to visualize the stickiness of the blood. Viscosity could be added implicitly by multiplying the viscosity constant along with the SPH kernel [Peer et al. 2015].

$$\mathbf{v}_i^* = \mathbf{v}_i - \sum_{j=1}^{k} [\gamma \frac{m_i}{\rho_i} (\mathbf{v}_i - \mathbf{v}_j) W(\mathbf{x}_i - \mathbf{x}_j, h)] \tag{4.11}$$

$\mathbf{v}_i^*$ represent the new velocity vector while γ is the viscosity constant, $\mathbf{v}_i$ and $\mathbf{x}_i$ are the current velocity and position of the particle i respectively and m_i is the particle's mass which assumed to be 1. Also, k refers to the total neighbor of particle i within a support radius h. Unlike the constraints described before, viscosity is applied as an external force in line (5) of algorithm 1.

4.3 Cardiac cycle

There are two phases in human cardiac cycle [Betts 2013]. The contraction phase where the heart shoots the blood throughout the body is called "Systole" and the relaxation phase which suggests to the moment when the muscle relaxes to allow the blood to fill the chambers is called "Diastole". These two stages can be expanded into more aspects including blood valves behavior, pumping rhythm and so forth, however for this study, only contraction and relaxation motion will be focused. Moreover, as discussed in the introduction section, the left ventricle chamber of the heart is the most essential component to be inspected in CVDs diagnosis, hence solely the systole and diastole phase of the left ventricle chamber will be focused.

4.3.1 Ventricular Systole

In systole, the ventricular muscles contract, starting to increase the blood pressure inside the ventricle chambers and finally the pressure forces the blood to begins opening the valves which direct the blood to the atriums. Generally, the left ventricle will generate more pressure than the right ventricle. However, both still deliver the same quantity of blood to the atriums [Betts 2013].

In order to imitate the muscle contraction, the surface particles of the heart must be explicitly shifted toward a certain point which is known as the center of the contraction. This center point is assumed to be the center of the heart model itself. Then the accurate

number of animation step should be calculated based on the actual cardiac cycle to best embody the motion. Fig.4.5 visualizes the process, while d represents the position of the surface particles at rest state and d^* represents the newly calculated position toward the center of the contraction.

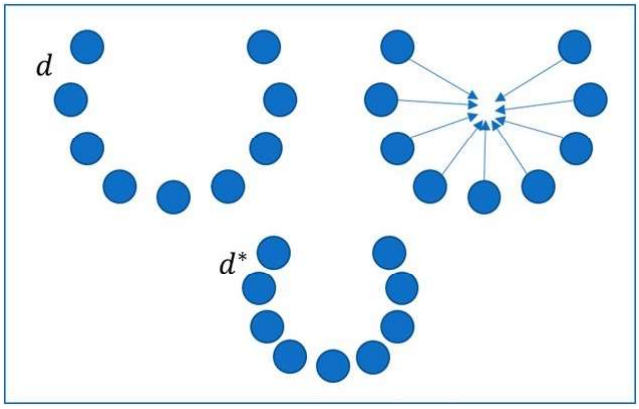


Figure 4.5: Systole motion

4.3.2 Ventricular Diastole

In diastole, the ventricular muscle relaxes causes the pressure of the blood inside the ventricles to drop. When the blood pressure inside the ventricles is dropped below the blood pressure inside in the atriums, the blood from the atriums will flow passes through the valves into the ventricles [Betts 2013].

To recreate the phenomenon, the surface particles have to be relaxed from the contraction. This can be archived by stop applying position changes to the particles and let the programmed constraints to reset the position of the particles into their resting stage.

4.3.3 Simulate cardiac cycle motion

This subsection discusses the actual implementation of the cardiac cycle. As discussed, there are 2 phases of the cardiac cycle; systole and diastole. The heart will only spend a certain amount of time in each phase. If the duration is long the heart rate will be low and vice versa. Naturally, this "beating duration" is only a split second, therefore this simulation model will be referring to this duration as a multiple of timestep $\triangle t$ rather than second. Moreover, the model also has to know the length that the heart muscles will move in each timestep which will be defined as "beating displacement". This length can be regarded as the difference between d and d^* as shown in Fig.4.5. Additionally, in a learning environment, medical students should be able to explore multiple cardiac cycle cases. Naturally, the whole ventricular chambers will actively contract, however, there are cases where only part of the chambers actually move. Hence in order to offer an interactive feature to also simulate those cases, the system will allow the students to explicitly select the part of the model that will participate in the systole phase. These parts can be selected according to the specific cycle that the students want to see. Since the model is representing by particles, these selected parts will be regarded as "selectedBeatingParticles" in Algorithm 2. In conclusion, there

are three elements that the system needs to know in order to simulate the cardiac cycle including the beating duration, beating displacement and selected beating particles.

The cardiac cycle can be simulated as follow. For the systole phase, the heart muscles or the selected beating particles will be shifted toward the center of the heart by the value equal to the defined displacement for an amount of time equal to the defined beating duration. For example, if the beating duration is defined as 10 timesteps and the beating displacement is defined as 0.01 $[m]$. In the systole phase, the heart muscles will be shifted toward the center by 0.01 $[m]$ for 10 timesteps which means at the end of this systole phase the heart muscles will be contracted by 0.1 $[m]$ from its rest state. For the diastole phase, the model will stop shifting the muscles and allow the Position Based Dynamics to manage the deformable constraints which will eventually return the muscles into its original state.

In order to actually implement this, the system needs to keep track of the current phase and the amount of time that the model has remained in that phase. The current phase of the model is referred to as "State" and the amount of time is referred to as "Steps". Also, two constant parameters have to be defined namely, BEATING_DURATION and BEATING_DISPLACE which refer to the beating duration and the beating displacement mentioned in the previous paragraph. During the simulation, the muscles will be shifted and the steps will be incremented each round. If the steps have exceeded the beating duration, the steps will be reset to 0 and the current state will be switched as shown in Fig.4.6. The algorithm is presented in Algorithm 2. Line (1)-(3), the constant parameters will be defined along with variables such as "steps" and "state". Originally, the state will be initialized as "systole" and steps as 0. During the simulation, the first step is to simulate the deformable properties of the heart model by calling the function "SimulateDeformableHeartModel" in line (5), then the algorithm will check if the current state is "systole", if so it will continue to move the beating particles toward the center of the heart according to the defined beating displacement by calling function "MovingParticles" as shown in Algorithm 3. "MovingParticles" function essentially calculate the directional vector *dir* from the current particle position pointing toward the center of the heart and multiply it with the beating displacement before adding it to the current position. The system can edit the positions directly by using PBD. The process will be repeated to every particle in selectedBeatingParticles.

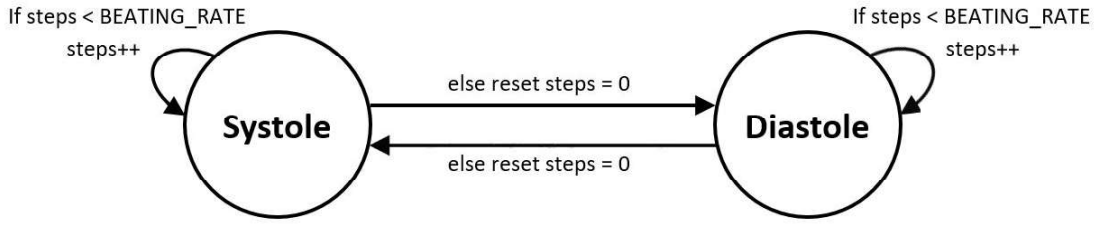


Figure 4.6: Cardiac cycle simulation state diagram

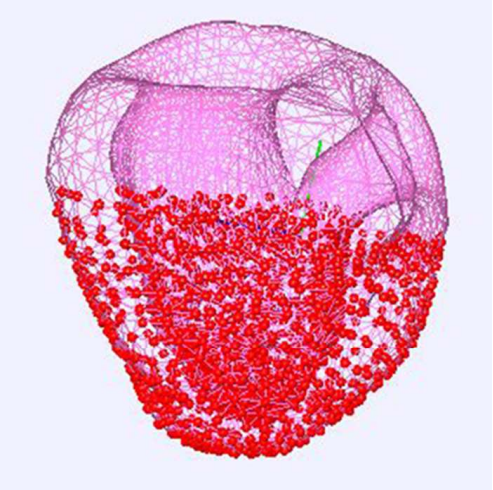
Algorithm 2 Cardiac Cycle Pseudocode

```
1: define BEATING_DURATION
2: define BEATING_DISPLACE
3: steps = 0, state = Systole
4: while true do
5:     SimulateDeformableHeartModel( )
6:     if state == Systole then
7:         MovingSelectedParticles(heartcenter, selectedBeatingParticles,
   BEATING_DISPLACE)
8:     steps++
9:     if steps >= BEATIONG_DURATION then
10:        steps = 0
11:        SwitchState( )
```

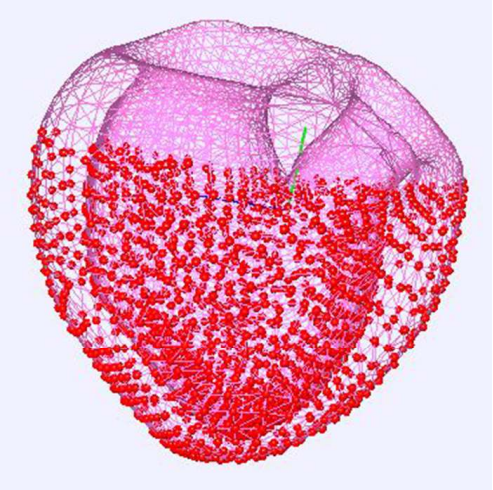
Algorithm 3 Moving Selected Particles

```
1: function MovingParticles(heartcenter, selectedBeatingParticles,
   BEATING_DISPLACE)
2:     for particlePos in selectedBeatingParticles do
3:         dir = heartcenter − particlePos
4:         Normalize(dir)
5:         UpdatePosition(particlePos + (dir ∗ BEATING_DISPLACE))
```

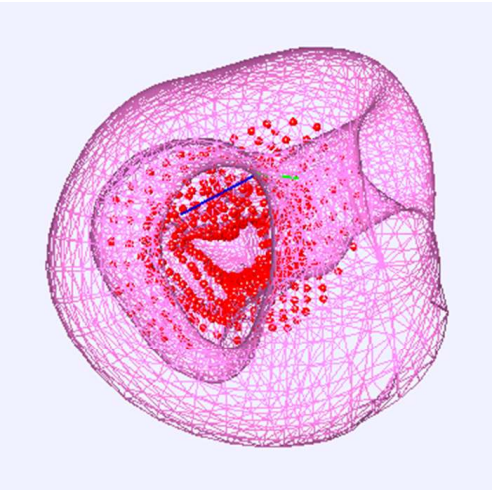
Fig.4.7 shows the result of the described algorithms. Fig.4.7(a) and Fig.4.7(b) demonstrating the end result of simulating the systole and diastole phase repectively. Similarly, Fig.4.7(c) and Fig.4.7(d) also demonstrating the same state, but in the top view. Note that the red circles refer to the selected beating particles that the user have selected.



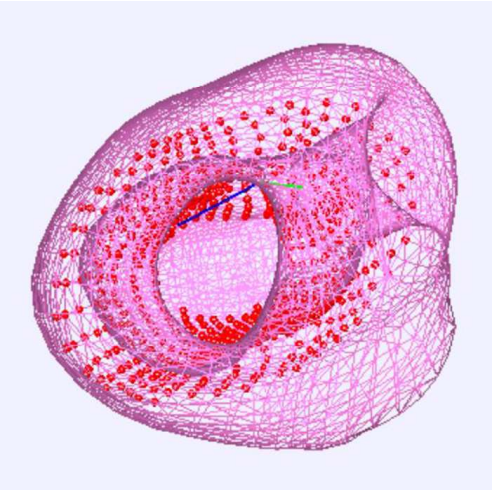
(a) The heart model at the end of systole phase



(b) The heart model at the end of diastole phase



(c) The heart model at the end of systole phase top view



(d) The heart model at the end of diastole phase top view

Figure 4.7: Result of the cardiac cycle simulation

4.4 Heart and blood interaction simulation

Since the three simulations described so far are based on the same technique which is PBD, they can be effortlessly combined together into a singular simulation that meets all of the mentioned constraints. However, in order to thoroughly realize the blood flow simulation, an interaction between heart and blood should be cautiously considered.

There are three steps involved in the interaction which is shown in Fig.4.8 Initially, the heart simulation model has to simulate the cardiac cycle motion described in the previous section. Then, the system will attempt to identify if the blood particles have penetrated the heart model. If the penetration is detected, the system will adjust the position. Finally, the predicted positions and velocities will be updated.

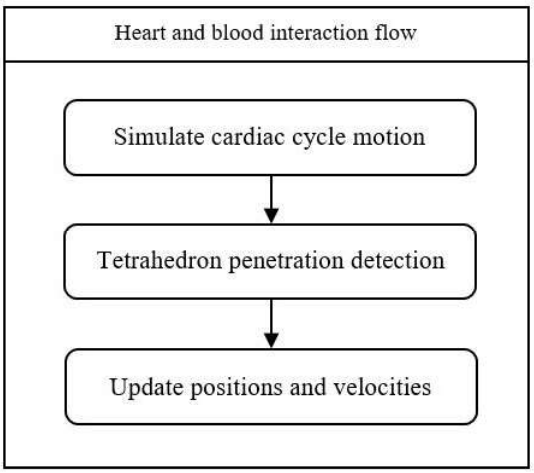


Figure 4.8: Heart and blood interaction flow

4.4.1 Tetrahedron penetration detection

The heart model is represented by a tetrahedral model as shown in Fig 4.10. The model composed of several tetrahedrons each with 4 faces. In order to test the penetration of a particle against a tetrahedron, the system follows 2 steps.

1. Check if the particle is inside a certain tetrahedron.
2. Use Plücker coordinates to calculate the face that the particle enters the tetrahedron and the intersection point [Platis et al. 2003].

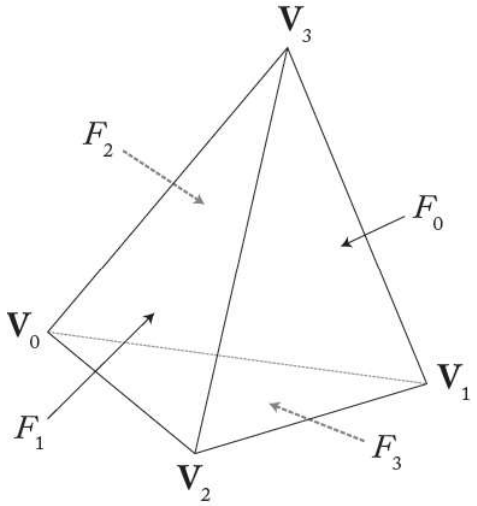


Figure 4.9: Tetrahedron vertices and faces

The first step, a particle can only be considered inside a tetrahedron when it lies under all the faces that hold the tetrahedron. Suppose a tetrahedron got $\mathbf{V}_0, \mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3$ as vertices and $F_3(\mathbf{V}_0, \mathbf{V}_1, \mathbf{V}_2)$, $F_2(\mathbf{V}_1, \mathbf{V}_0, \mathbf{V}_3)$, $F_1(\mathbf{V}_2, \mathbf{V}_3, \mathbf{V}_0)$, $F_0(\mathbf{V}_3, \mathbf{V}_2, \mathbf{V}_1)$ are the faces of the tetrahedron as displayed in Fig.4.9. While $\mathbf{V}_0^i$, $\mathbf{V}_1^i$, $\mathbf{V}_2^i$ refer to the vertices of face F_i; for example for F_2 above $\{\mathbf{V}_0^2 = \mathbf{V}_1, \mathbf{V}_1^2 = \mathbf{V}_0, \mathbf{V}_3^2 = \mathbf{V}_3\}$. Let $\mathbf{X}$ be a point that lie on the

face F_i, $\mathbf{P}_j$ be the position of the particle j that is going to be tested. If $\mathbf{N_i}$ is a normalized vector of the cross product between $(\mathbf{V}_1^i - \mathbf{V}_0^i)$ and $(\mathbf{V}_2^i - \mathbf{V}_0^i)$ and $\mathbf{U}$ is a vector pointing from $\mathbf{X}$ to $\mathbf{P}$. Let θ_i be the dot product between $\mathbf{N_i}$ and $\mathbf{U}$, the position $\mathbf{P}_j$ lies under the face F_i if and only if θ_i is less than 0.

$$\theta_i = \mathbf{N_i} \cdot \mathbf{U} = |(\mathbf{V}_1^i - \mathbf{V}_0^i) \times (\mathbf{V}_2^i - \mathbf{V}_0^i)| \cdot (\mathbf{X} - \mathbf{P}) \tag{4.12}$$

$$\theta_i < 0 \iff \text{position } \mathbf{P} \text{ lies under the face } F_i \tag{4.13}$$

With equation (4.12) and (4.13), the particle would be considered inside a tetrahedron when $\theta_i < 0$ for all F_i, $i \in \{0, 1, 2, 3\}$.

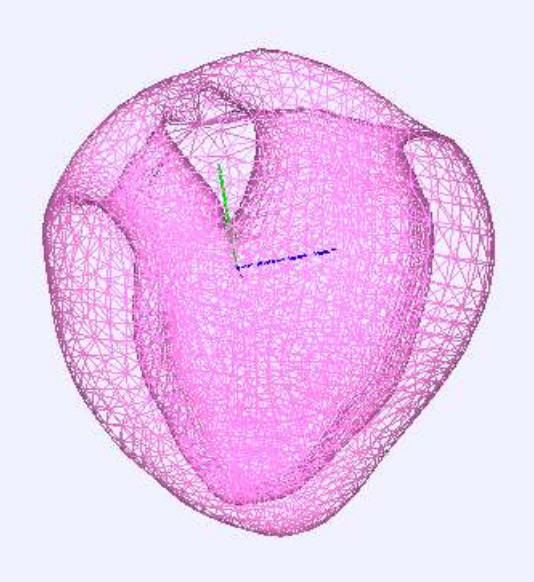


Figure 4.10: Tetrahedral mesh

For the second step after the particle has been verified to be inside the tetrahedron, a ray-tetrahedron intersection algorithm was adopted to detect the intersection point and the entering face [Platis et al. 2003]. Plücker coordinates is an alternative way to represent a line in 3-dimensional space by using 6-dimensional vectors. Given a ray r with a direction vector $\mathbf{L}$ and determined by a point $\mathbf{P}$, the Plücker coordinates π_r are defined as followed;

$$\pi_r = \{\mathbf{L} : \mathbf{L} \times \mathbf{P}\} = \{\mathbf{U}_r : \mathbf{V}_r\} \tag{4.14}$$

Also, given two rays r and s, the permuted inner product between them can be defined as;

$$\pi_r \odot \pi_s = \mathbf{U}_r \cdot \mathbf{V}_s + \mathbf{U}_s \cdot \mathbf{V}_r \tag{4.15}$$

therefore valuable information can be derived as;

$$\begin{aligned} \pi_r \odot \pi_s > 0 &\iff s \text{ goes counterclockwise around } r \\ \pi_r \odot \pi_s < 0 &\iff s \text{ goes clockwise around } r \\ \pi_r \odot \pi_s = 0 &\iff s \text{ intersects or is parallel to } r \end{aligned} \tag{4.16}$$

By using this information, the algorithm can identify the entering and leaving faces which are the face that the particle intersects when it begins to penetrate the tetrahedron

and the face that the particle will intersect when it omits the tetrahedron. The ray r will be held entering a face $F_i(\mathbf{V}_0^i, \mathbf{V}_1^i, \mathbf{V}_2^i)$ when it goes counterclockwise around all edges of F_i specifically, $\mathbf{e}_0(\mathbf{V}_1^i, \mathbf{V}_2^i)$, $\mathbf{e}_1(\mathbf{V}_2^i, \mathbf{V}_0^i)$ and $\mathbf{e}_2(\mathbf{V}_0^i, \mathbf{V}_1^i)$ and leaving a face when it goes clockwise around them.

$$\begin{aligned} r \text{ enters a face} &\iff \pi_r \odot \pi_{e_i} \geq 0 \;\forall\; i \text{ AND } \exists \text{ j} : \pi_r \odot \pi_{e_j} \neq 0 \\ r \text{ leaves a face} &\iff \pi_r \odot \pi_{e_i} \leq 0 \;\forall\; i \text{ AND } \exists \text{ j} : \pi_r \odot \pi_{e_j} \neq 0 \\ r \text{ is coplannar with a face} &\iff \pi_r \odot \pi_{e_i} = 0 \;\forall\; i \end{aligned} \tag{4.17}$$

The algorithm to obtain the entering face and the intersection point is illustrated in Algorithm 4 and 5 sequentially. In Algorithm 4, δ_j^i is the sign of $(\pi_r \odot \pi_j^i)$;

$$\delta_j^i = sign(\pi_r \odot \pi_j^i) = \begin{cases} 1, & \text{if } \pi_r \odot \pi_j^i > 0 \\ 0, & \text{if } \pi_r \odot \pi_j^i = 0 \\ -1, & \text{if } \pi_r \odot \pi_j^i < 0 \end{cases} \tag{4.18}$$

It is notable that side product function in Algorithm 5 is the permuted inner product of the Plücker coordinate which is also demonstrated in equation (4.19).

Algorithm 4 Ray-Tetrahedron intersection pseudocode [Platis et al. 2003]

```
F_enter = nil
F_leave = nil
for i = 3, 2, 1, 0 do
    Compute δ_0^i, δ_1^i and δ_2^i
    if (δ_0^i ≠ 0) OR (δ_1^i ≠ 0) OR (δ_2^i ≠ 0) then
        if (F_enter == nil) AND (δ_0^i ≥ 0) AND (δ_1^i ≥ 0) AND (δ_2^i ≥ 0) then
            F_enter = F_i
        else if (F_leave == nil) AND (δ_0^i ≤ 0) AND (δ_1^i ≤ 0) AND (δ_2^i ≤ 0) then
            F_leave = F_i
```

Algorithm 5 Compute Cartesian Coordinate pseudocode

```
s_0 = SideProduct(ray, e_0)
s_1 = SideProduct(ray, e_0)
s_2 = SideProduct(ray, e_0)
u_0 = s_1 / (s_1 + s_2 + s_3)
u_1 = s_2 / (s_1 + s_2 + s_3)
u_2 = s_3 / (s_1 + s_2 + s_3)
pos = (u_0 × V_2^i) + (u_1 × V_0^i) + (u_2 × V_1^i)
dist = p − ||origin||
```

$$\begin{aligned} SideProduct(L_1, L_2) = (L_1[0] \times L_2[4]) + (L_1[1] \times L_2[5]) + (L_1[2] \times L_2[3]) \\ + (L_1[3] \times L_2[2]) + (L_1[4] \times L_2[0]) + (L_1[5] \times L_2[1]) \end{aligned} \tag{4.19}$$

Consequently, Algorithm 5 will yield the intersected point *pos* in Cartesian coordinates along with the scalar distance value *dist* indicating how far the particle is inside the tetrahedron. With these values, the system can now correct the position of the identified particles.

This can be arranged by moving the particle along the ray r by the scalar $dist$ plus the particle radius $radius$ to ensure that the particle is outside of the tetrahedron, the equation can be written as follow.

$$p_i^* = p_i - \{\mathbf{r} \times (dist + radius)\} \tag{4.20}$$

where p_i^* is the corrected position and p_i is the old position of the detected particle i.

Chapter 5

Results

This chapter will discuss the environment used in the implementation process and the actual results of the simulation.

5.1 Implementation Environment

Table 5.1 shows the computational resources used to implement this system while table 5.2, 5.3 and 5.4 demonstrating the constant parameters for blood, heart and cardiac cycle simulation respectively. Note that timestep $\triangle t = 0.005$ for these experiments.

Table 5.1: Implementation Environment

CPU	Intel(R) Core(TM) i7-7700K 4.20 GHz
RAM	16 GB
GPU	NVIDIA GeForce GTX Titan
VRAM	6 GB
OS	Window 10
Graphic API	OpenGL

Table 5.2: Blood simulation constants

Simulation type	Constants	Value
Blood	Fluid viscosity	0.04 $kg/(s \cdot m)$
	Particle Radius	0.08 m
	Mass of fluid particle	0.001 kg
	Number of particle	1,000
	Rest Density	1,000 kg/m^3
	Kernel support radius	0.32 m

Table 5.3: Heart simulation constants

Simulation type	Constants	Value
Heart	Stiffness parameter	0.5
	Gravitational acceleration	9.81 m/s^2
	Number of vertices	1061
	Number of faces	2,118
	Number of tetrahedrons	3,345

Table 5.4: Cardiac Cycle constants

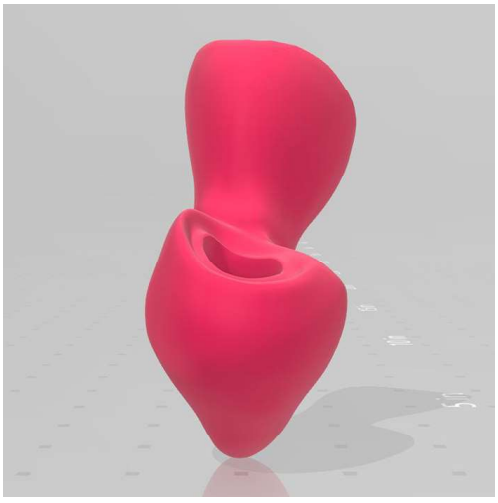
Simulation type	Constants	Value
Cardiac Cycle	BEATING_DURATION	8
	BEATING_DISPLACE	0.08

5.2 Heart Model

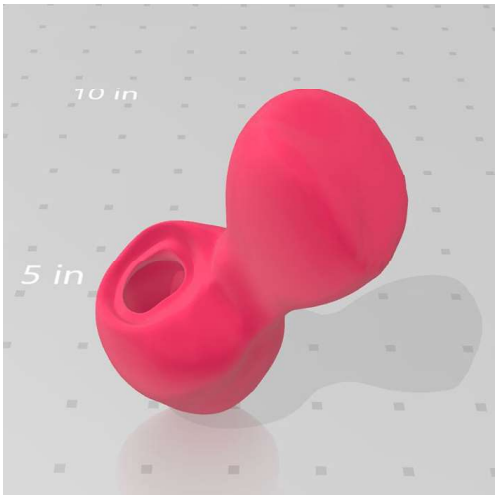
This subsection shows the input 3D heart model that was manually drawn, by using a 3D computer graphics software called Blender, to primarily verify the algorithms. Afterward, it can be replaced with any 3D heart model. The input heart model is shown in Fig.5.1. Note that in this research, only left ventricle with two valves and a blood vessel has been simulated and modeled.

(a) Front view of the input heart model

(b) Right side view of the input heart model



(c) Left side view of the input heart model



(d) Top view of the input heart model

Figure 5.1: Input heart model

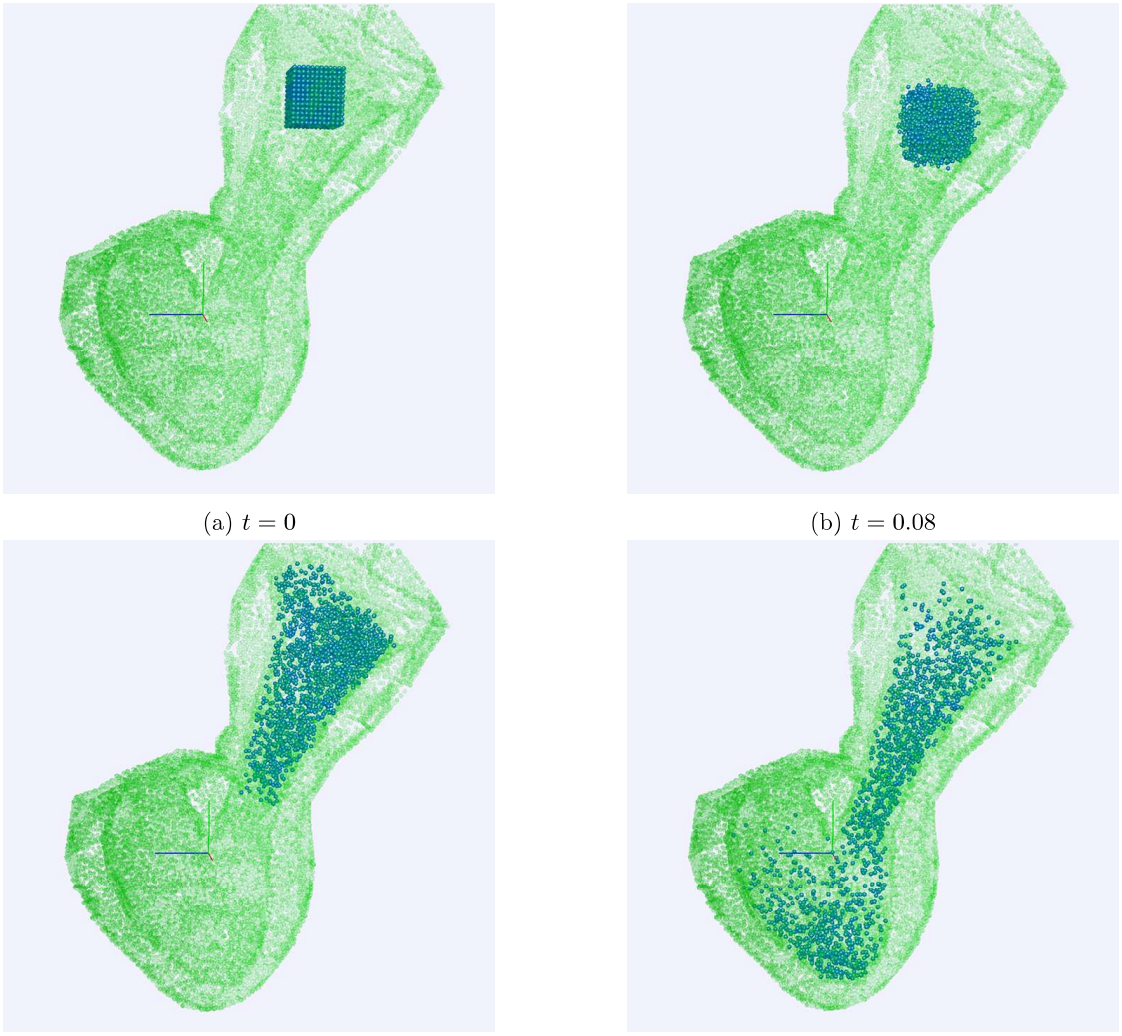
5.3 Simulation results

This section will compare the result from initializing the position of blood fluid in the blood vessel and the result from initializing it inside the heart chamber directly and also the differences between integrating and not integrating the cardiac cycle. These four situations will be simulated for the evaluation of this research as shown in Table 5.5.

The results will be presented as follow. For each experiments, there will be four sets of images captured from the simulation in different time $t = \{0, 0.08, 1.0, 2.0\}$. A set of images composed of four different views including front, right, left and top views represented by sub figure (a), (b), (c) and (d) respectively. The corresponding results, figure numbers and computational time are also shown in Table 5.5.

Table 5.5: Simulated result

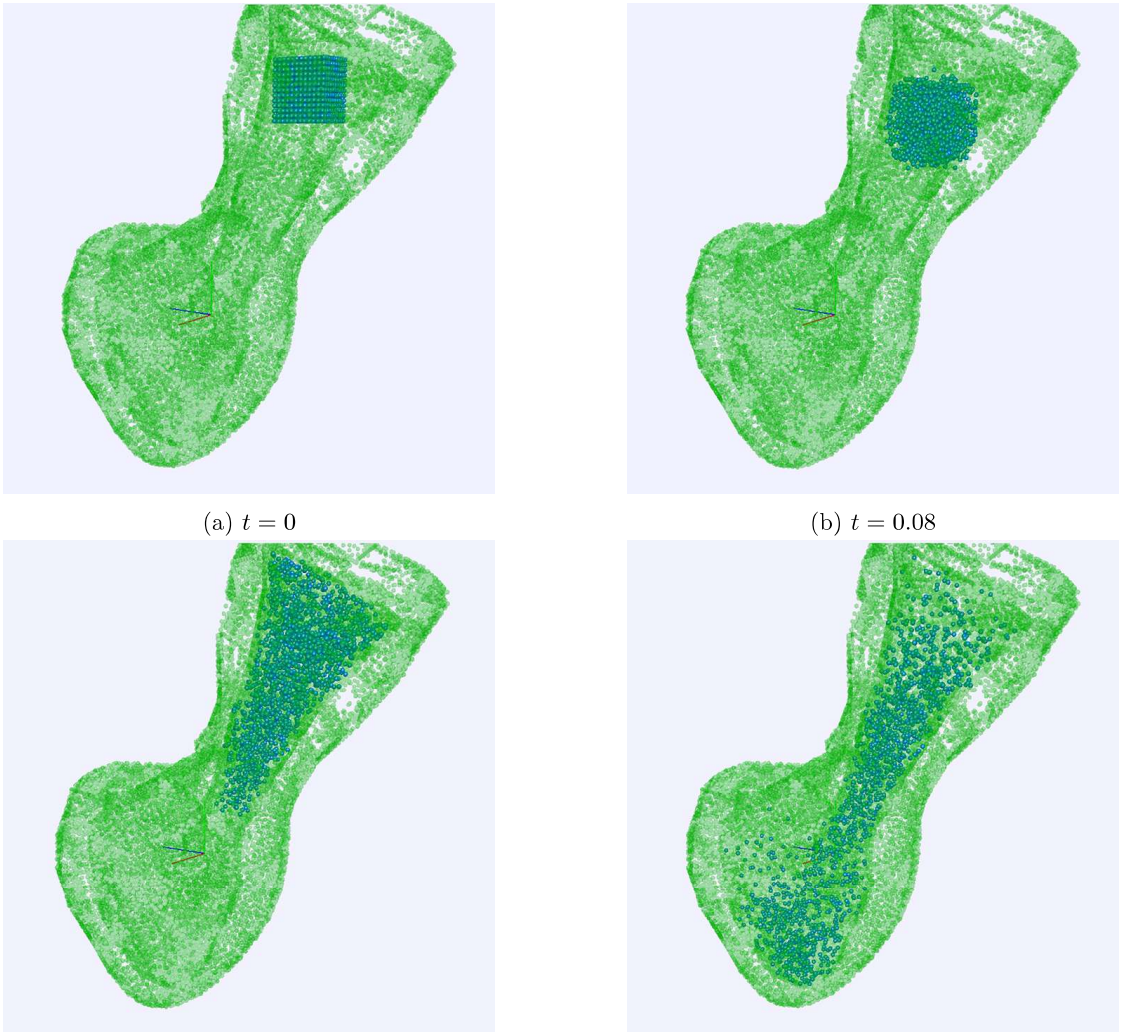
	Start position of blood fluid	Cardiac cycle	Figures	Computational time
1	Inside the blood vessel	No	Fig.5.2 to Fig.5.5	1 m 19 s 08
2	Inside the blood vessel	Yes	Fig.5.6 to Fig.5.9	1 m 49 s 29
3	Inside the heart chamber	No	Fig.5.10 to Fig.5.13	1 m 21 s 35
4	Inside the heart chamber	Yes	Fig.5.14 to Fig.5.17	1 m 51 s 17



(a) $t = 0$ (b) $t = 0.08$

(c) $t = 1.00$ (d) $t = 2.00$

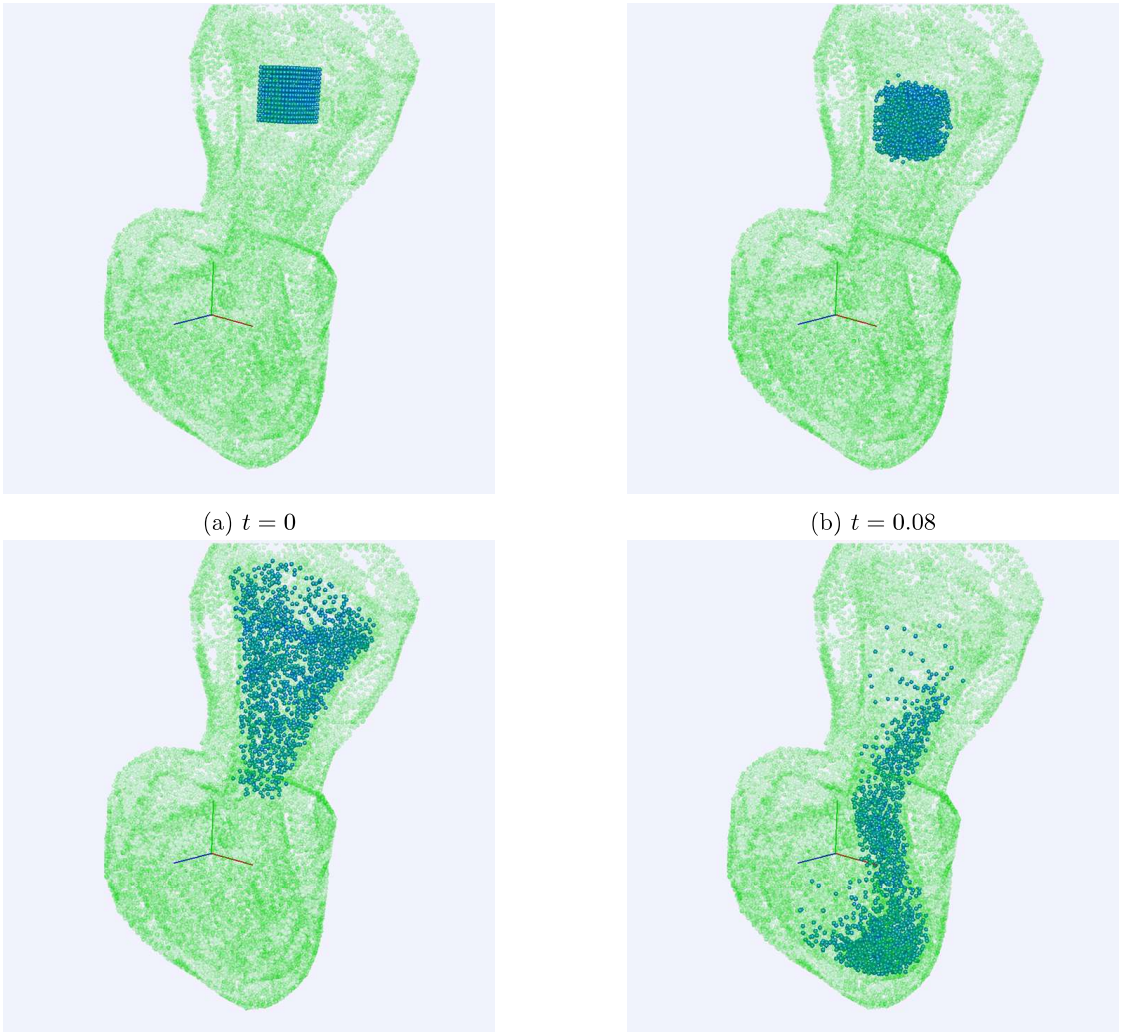
Figure 5.2: Blood flow simulation in the vessel without cardiac cycle front view



(a) $t = 0$ (b) $t = 0.08$

(c) $t = 1.00$ (d) $t = 2.00$

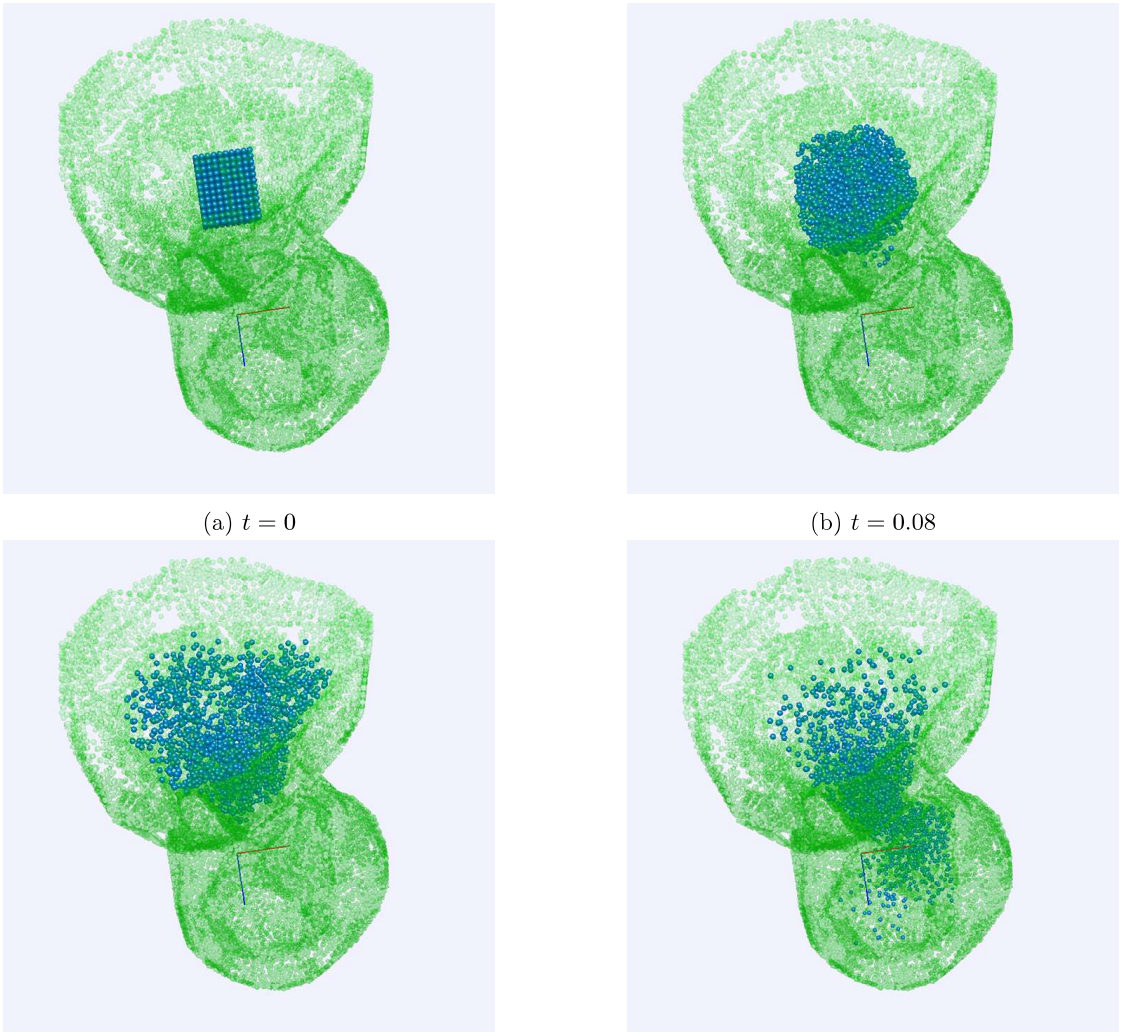
Figure 5.3: Blood flow simulation in the vessel without cardiac cycle left view



(a) $t = 0$ (b) $t = 0.08$

(c) $t = 1.00$ (d) $t = 2.00$

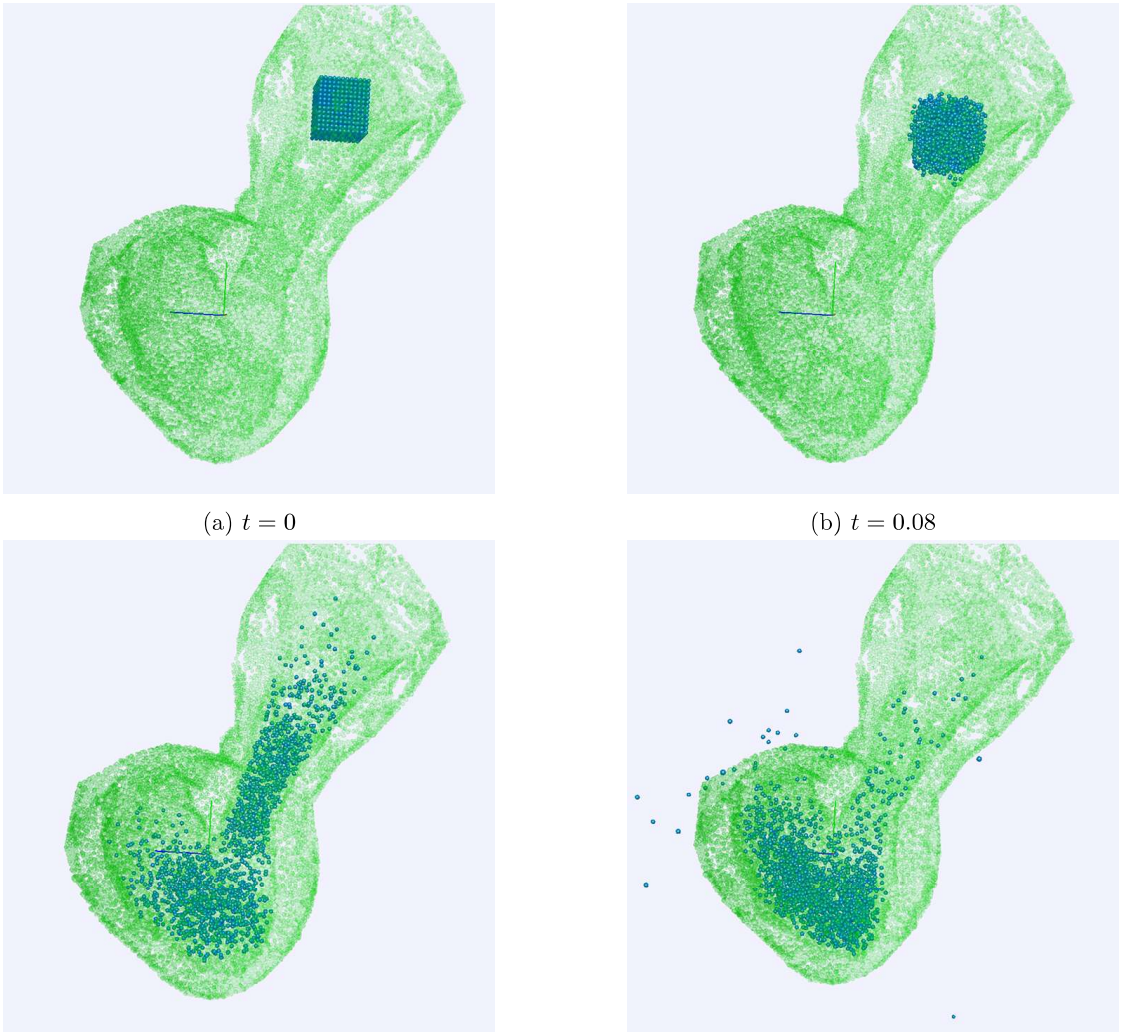
Figure 5.4: Blood flow simulation in the vessel without cardiac cycle right view



(a) $t = 0$ (b) $t = 0.08$

(c) $t = 1.00$ (d) $t = 2.00$

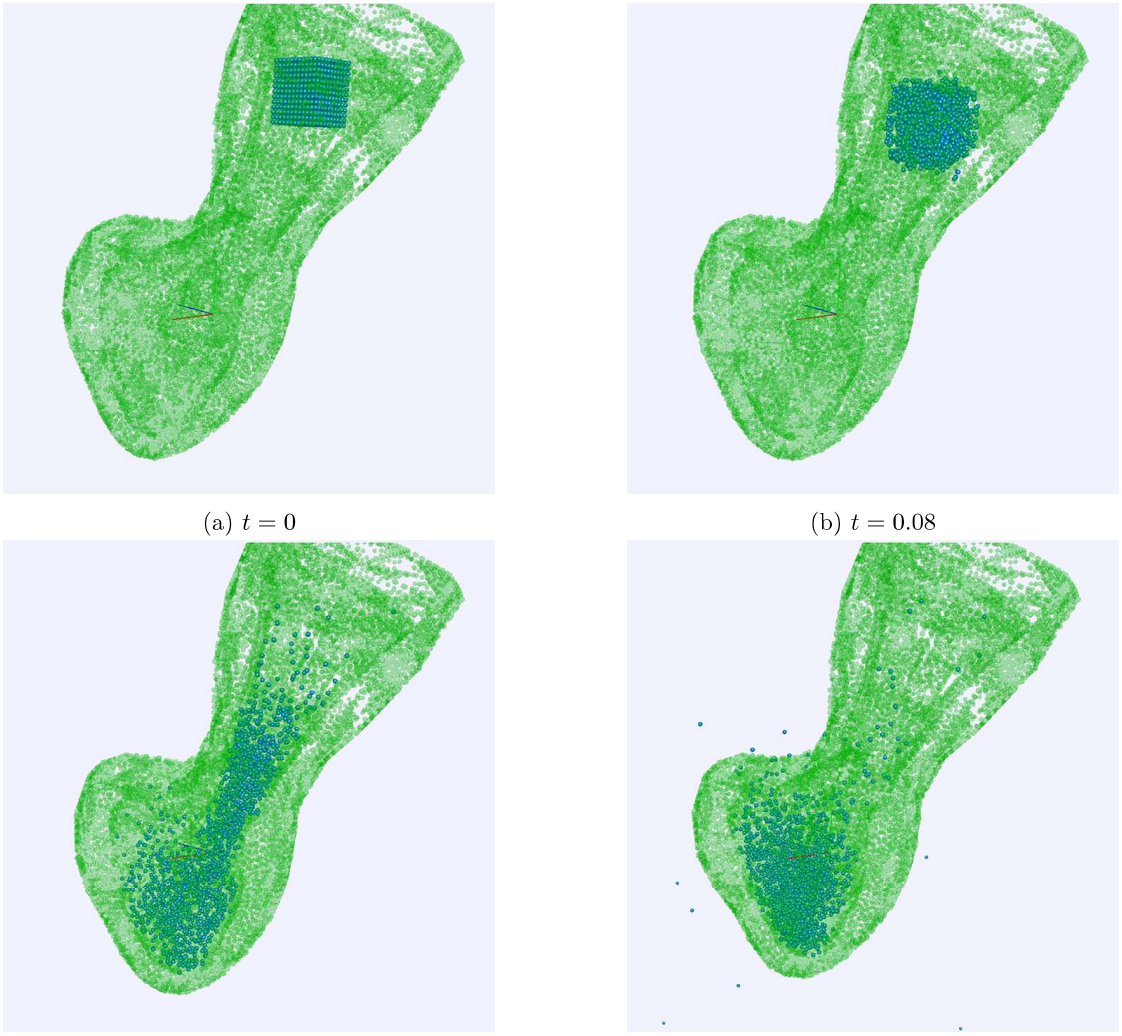
Figure 5.5: Blood flow simulation in the vessel without cardiac cycle top view



(a) $t = 0$ (b) $t = 0.08$

(c) $t = 1.00$ (d) $t = 2.00$

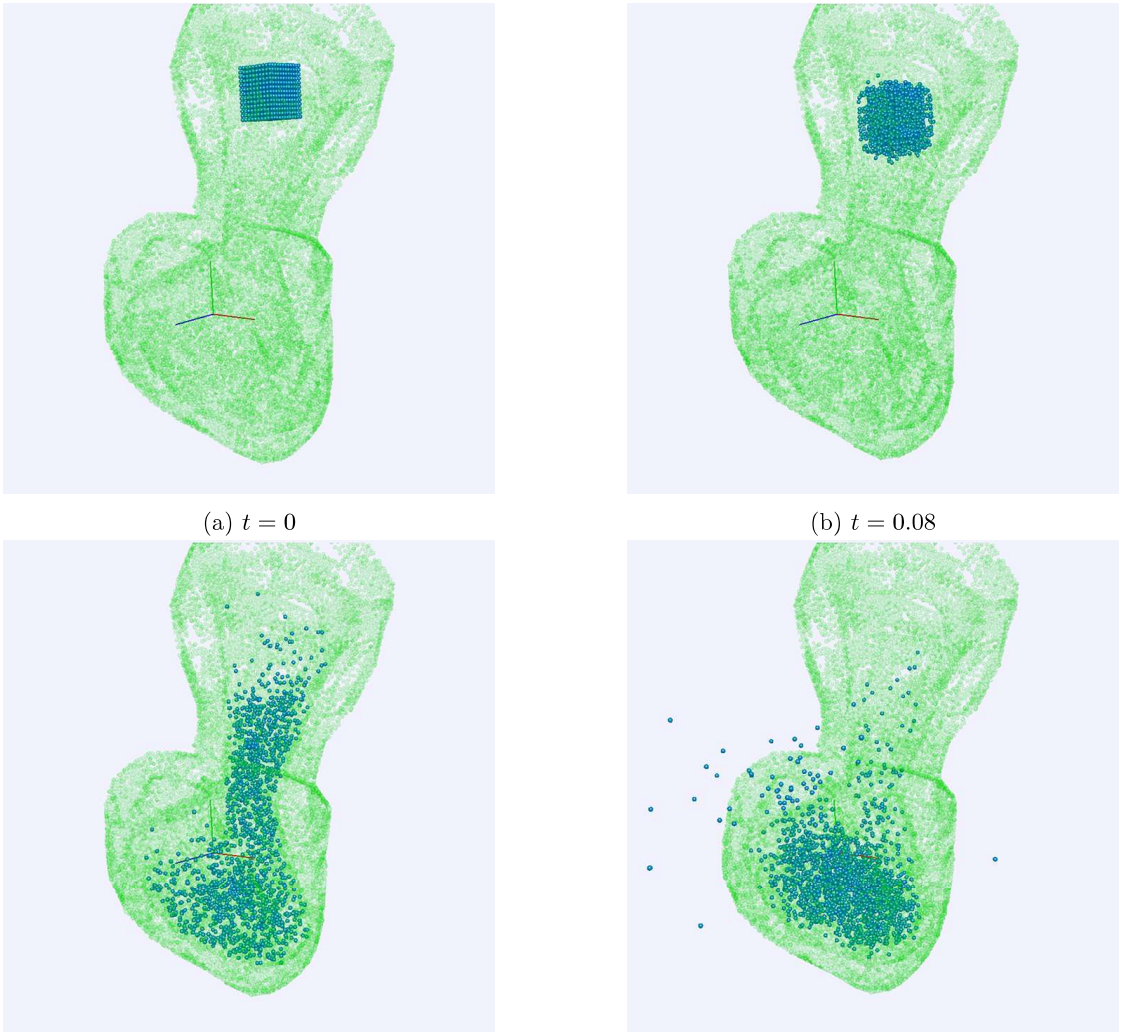
Figure 5.6: Blood flow simulation in the vessel with cardiac cycle front view



(a) $t = 0$ (b) $t = 0.08$

(c) $t = 1.00$ (d) $t = 2.00$

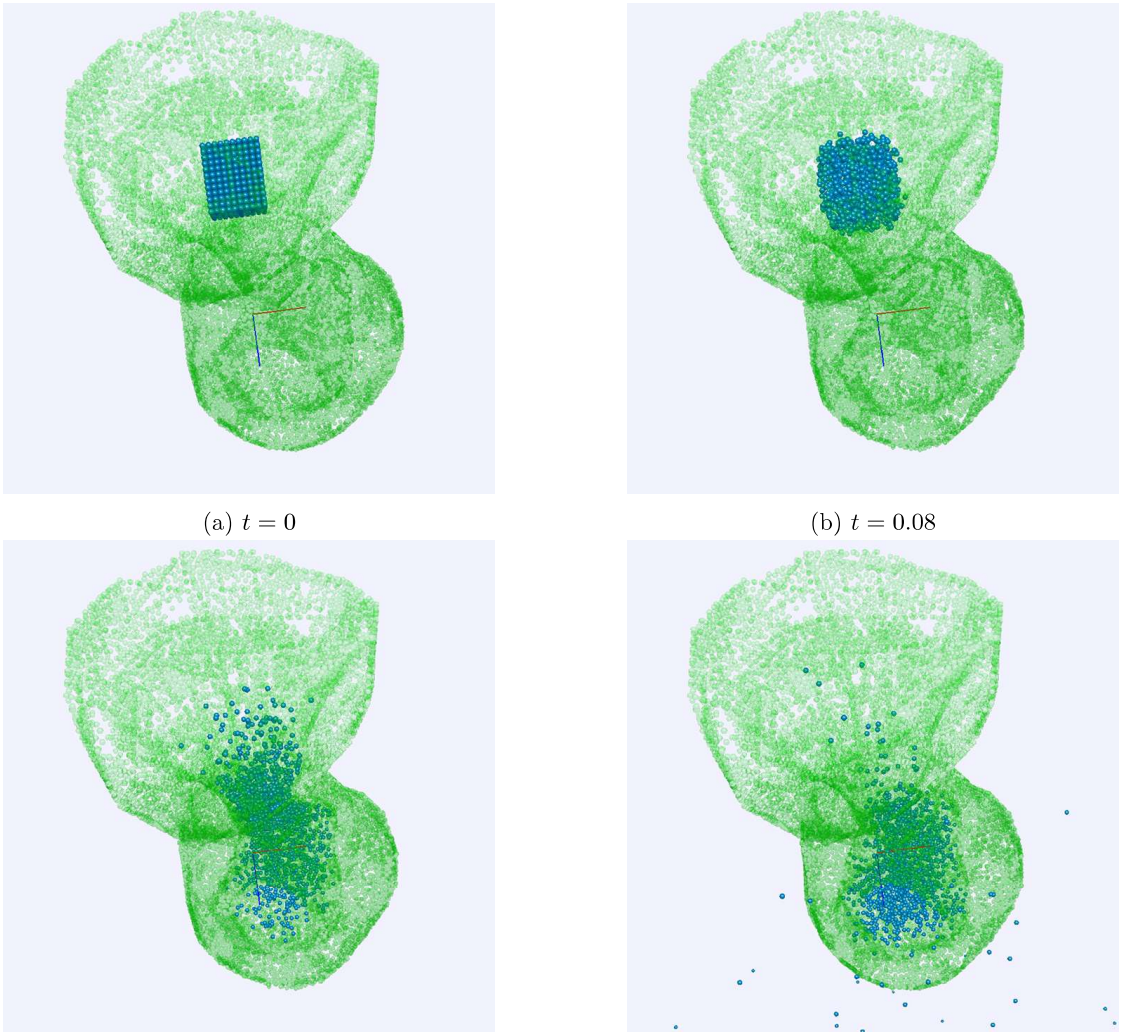
Figure 5.7: Blood flow simulation in the vessel with cardiac cycle left view



(a) $t = 0$ (b) $t = 0.08$

(c) $t = 1.00$ (d) $t = 2.00$

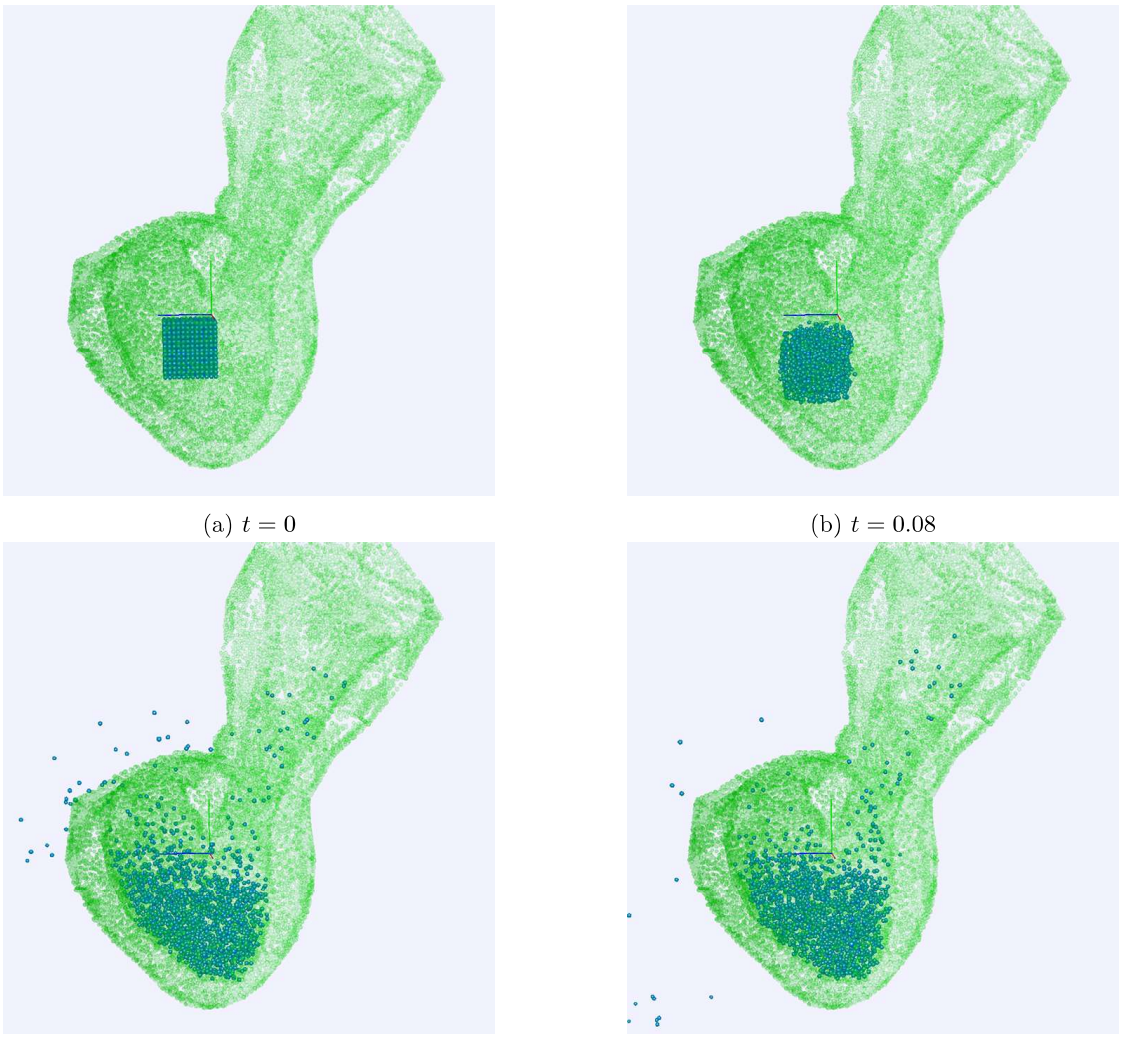
Figure 5.8: Blood flow simulation in the vessel with cardiac cycle right view



(a) $t = 0$ (b) $t = 0.08$

(c) $t = 1.00$ (d) $t = 2.00$

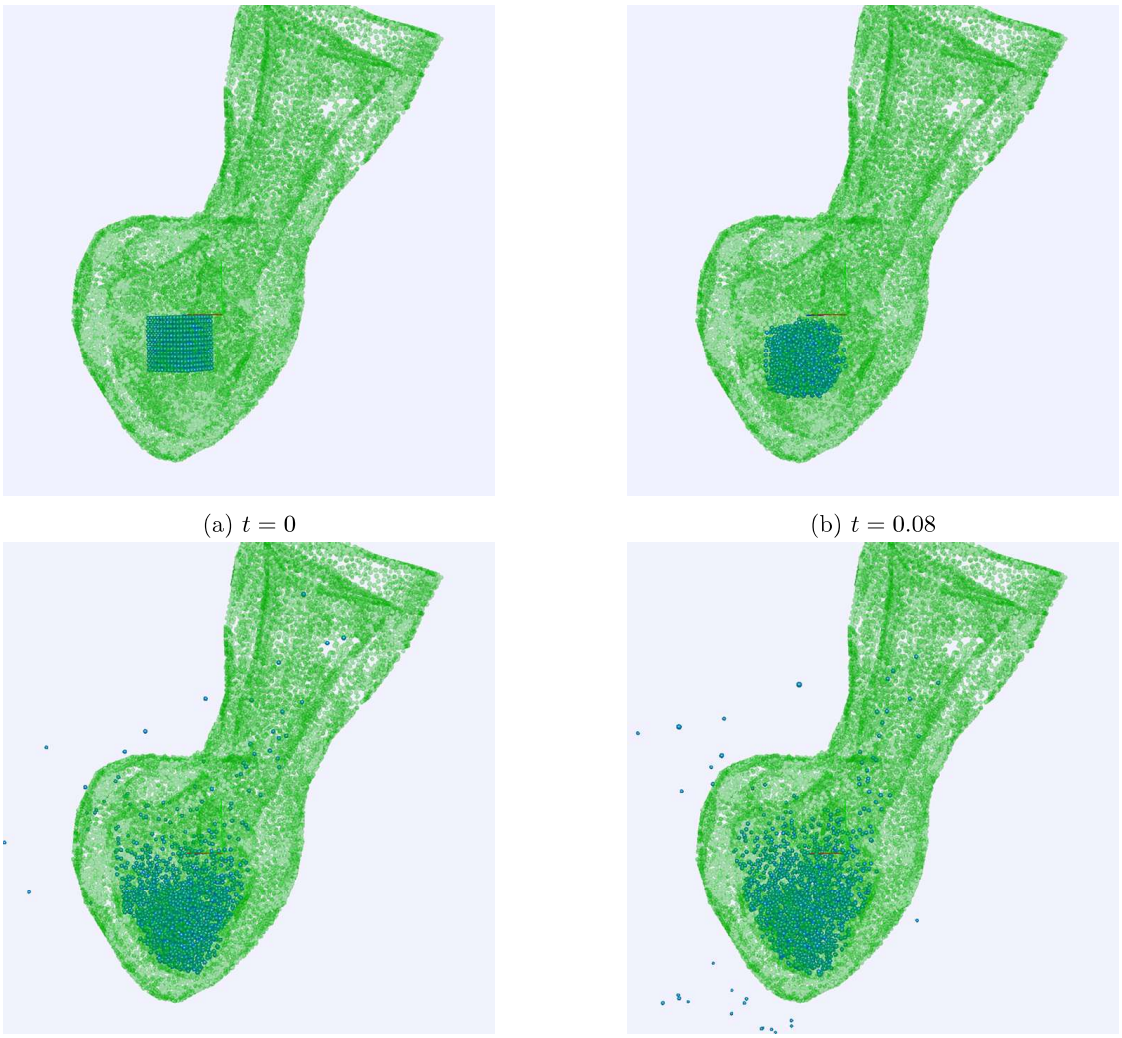
Figure 5.9: Blood flow simulation in the vessel with cardiac cycle top view



(a) $t = 0$ (b) $t = 0.08$

(c) $t = 1.00$ (d) $t = 2.00$

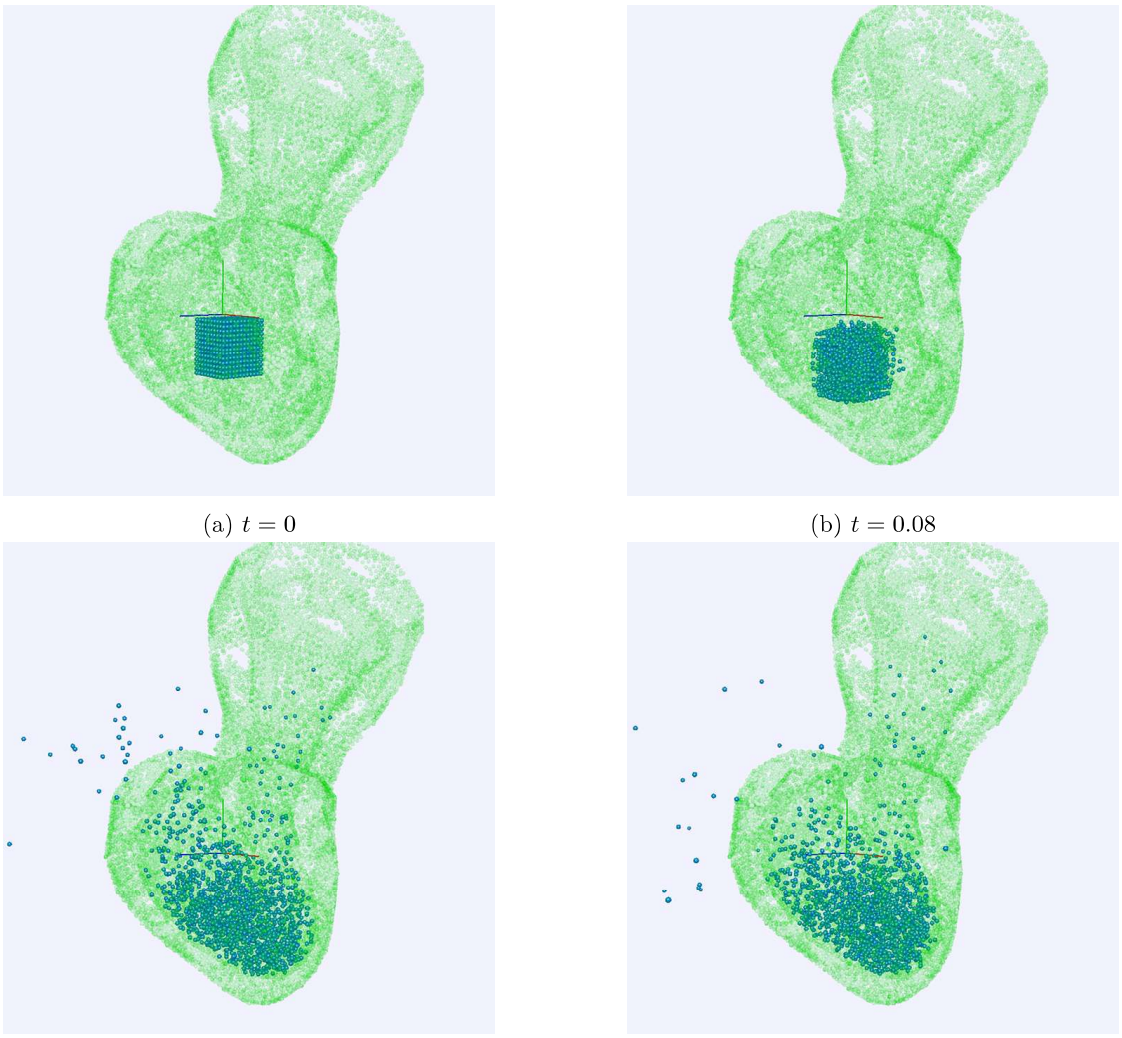
Figure 5.10: Blood flow simulation inside the heart model without cardiac cycle front view



(a) $t = 0$ (b) $t = 0.08$

(c) $t = 1.00$ (d) $t = 2.00$

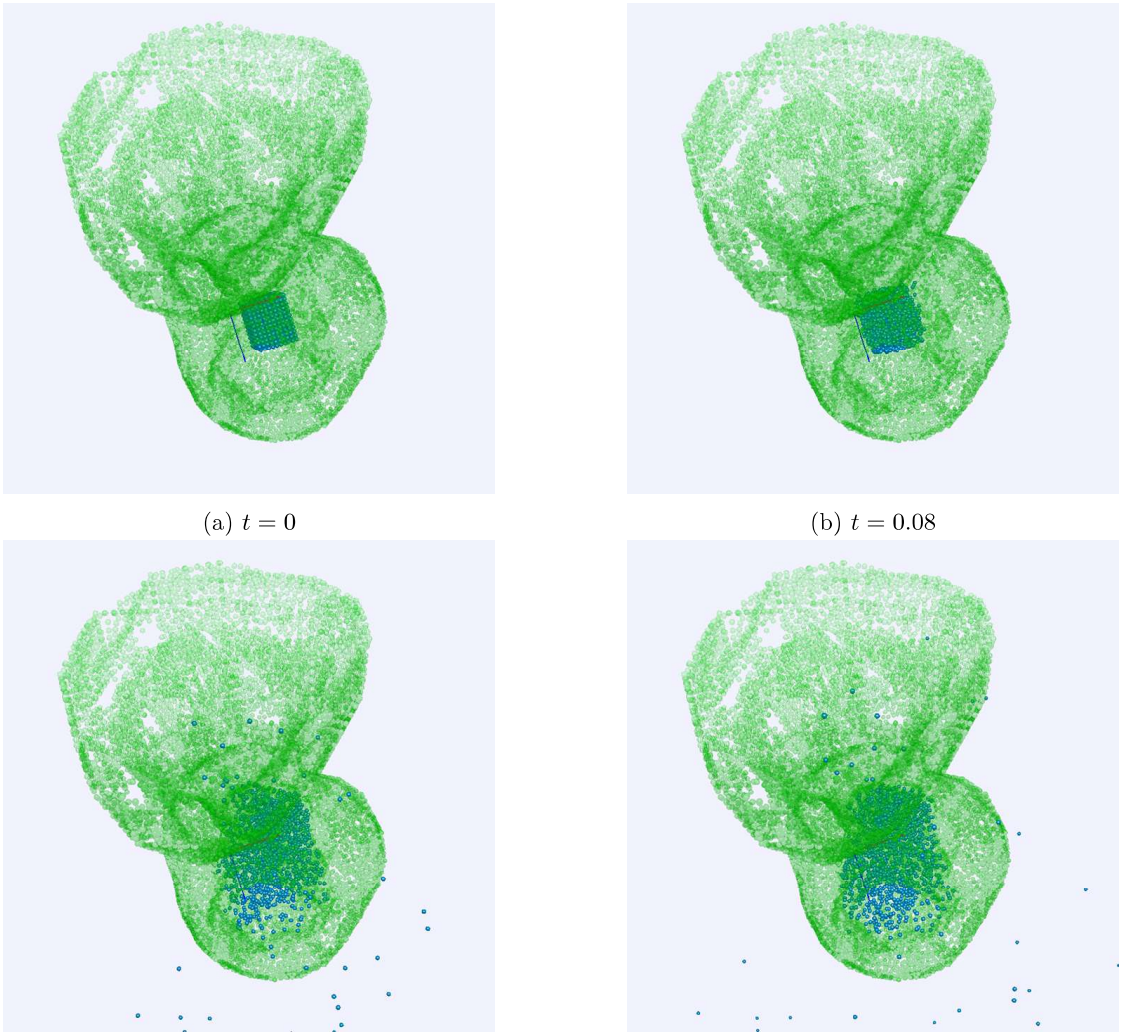
Figure 5.11: Blood flow simulation inside the heart model without cardiac cycle left view



(a) $t = 0$ (b) $t = 0.08$

(c) $t = 1.00$ (d) $t = 2.00$

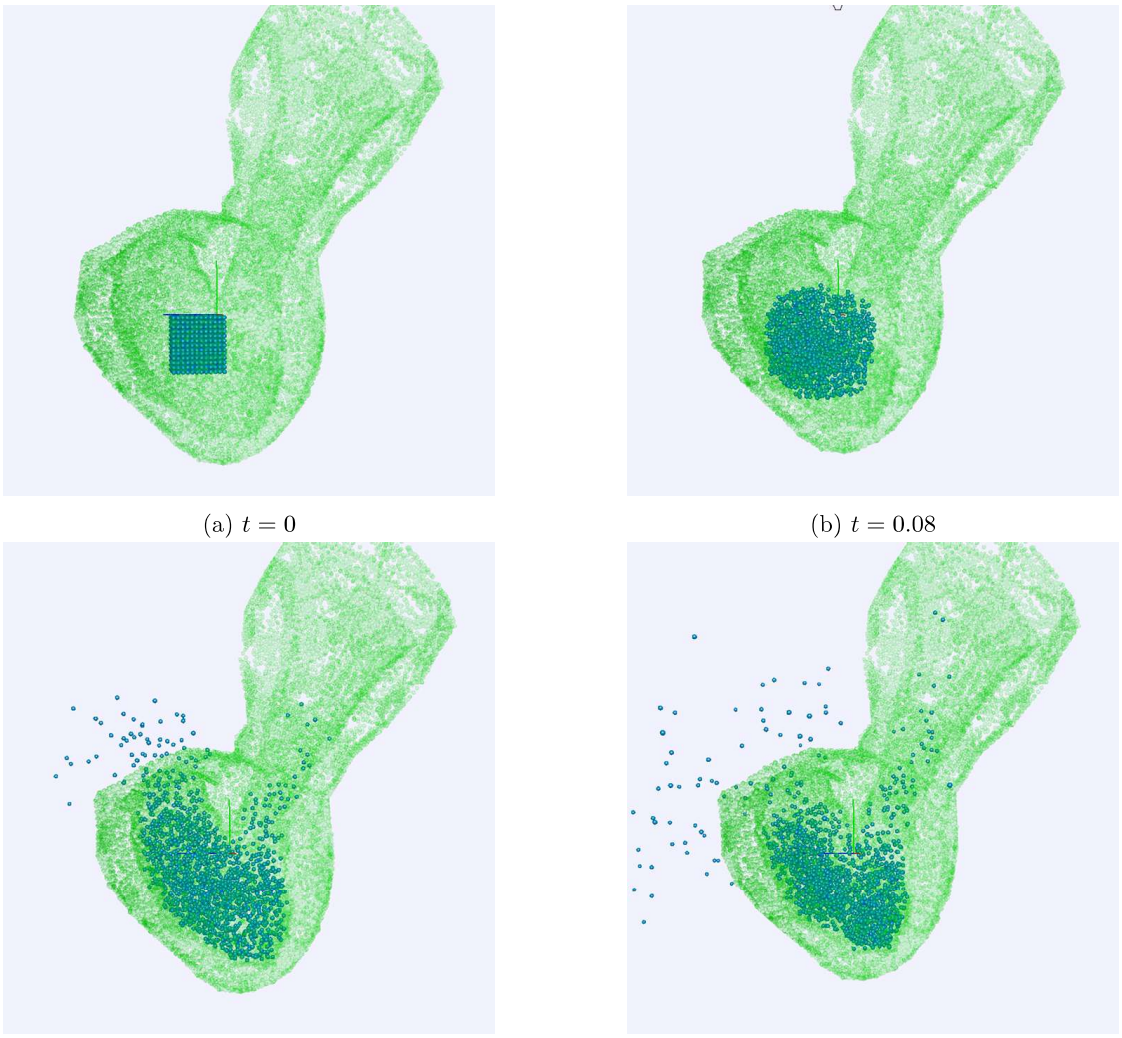
Figure 5.12: Blood flow simulation inside the heart model without cardiac cycle right view



(a) $t = 0$ (b) $t = 0.08$

(c) $t = 1.00$ (d) $t = 2.00$

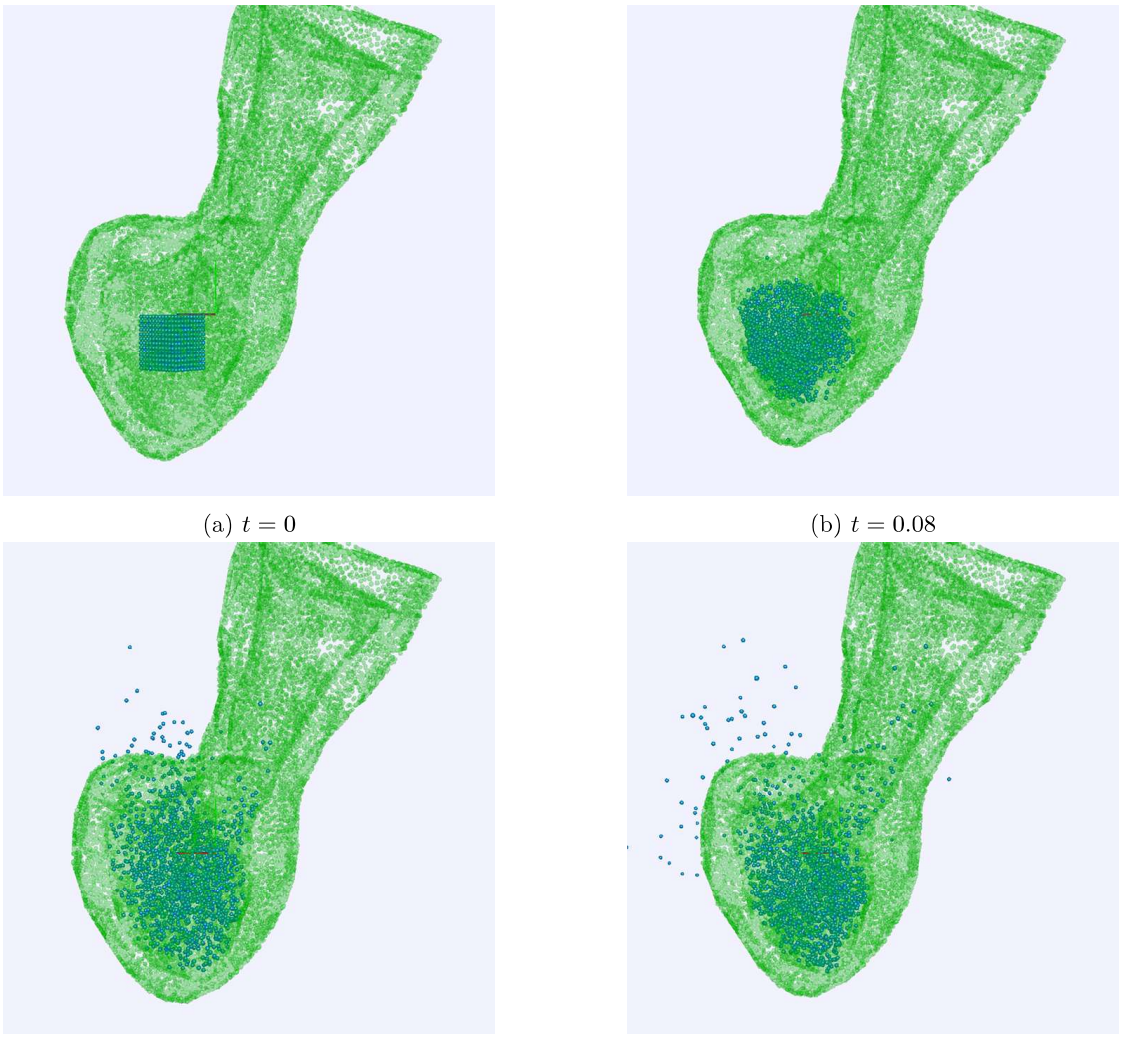
Figure 5.13: Blood flow simulation inside the heart model without cardiac cycle top view



(a) $t = 0$ (b) $t = 0.08$

(c) $t = 1.00$ (d) $t = 2.00$

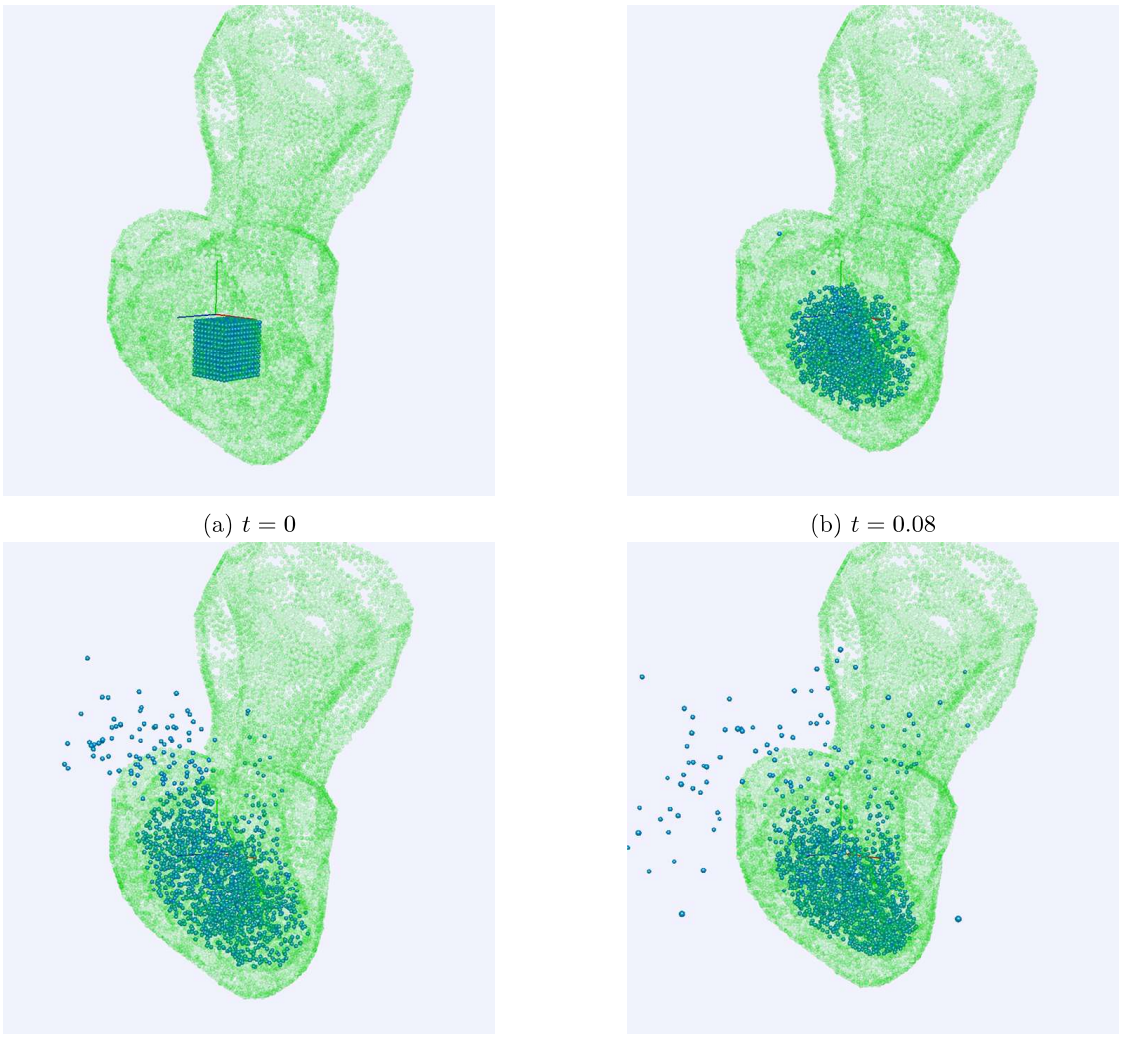
Figure 5.14: Blood flow simulation inside the heart model with cardiac cycle front view



(a) $t = 0$ (b) $t = 0.08$

(c) $t = 1.00$ (d) $t = 2.00$

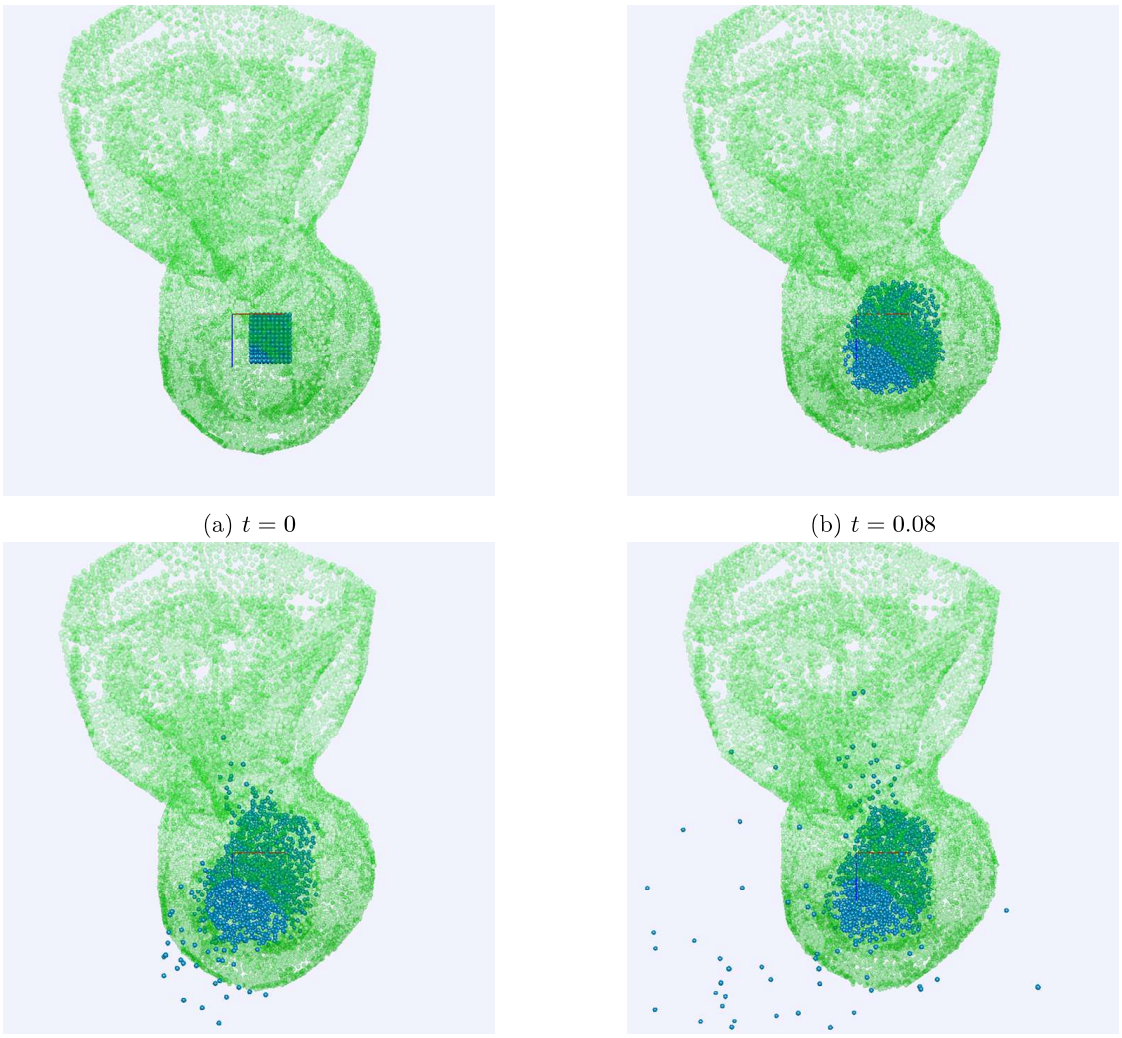
Figure 5.15: Blood flow simulation inside the heart model with cardiac cycle left view



(a) $t = 0$ (b) $t = 0.08$

(c) $t = 1.00$ (d) $t = 2.00$

Figure 5.16: Blood flow simulation inside the heart model with cardiac cycle right view



(a) $t = 0$ (b) $t = 0.08$

(c) $t = 1.00$ (d) $t = 2.00$

Figure 5.17: Blood flow simulation inside the heart model with cardiac cycle top view

Chapter 6

Evaluation

After implementing the proposed method, another 40 minutes interview was conducted. The interviewee was the same medical professor who provided the consultation about this educational tool in the preliminary section. As mentioned, the professor is currently teaching in a medical school in Thailand namely, Faculty of Medicine, Siriraj Hospital, and also positioning in the hospital. The interview was conducted online via Skype video call. The professor was presented with the simulation in real-time and was explained about the process and flow of the simulation, then he was inquired about his evaluation with a set of guided questions as shown in Table 6.1.

Table 6.1: Interview questions

	Questions
1	What do you think about the simulation in term of the accuracy of the mechanism?
2	What do you think about the feature where the students specify the beating parts in systole phase?
3	What potential do you think this simulation model has to be integrated into medical education?
4	What features do think should be added in order to better serve the educational purpose?

Conclusively, the discussion points that he provided can be divided into 3 main categories; (1) Virtual accuracy, (2) Functionalities, and (3) Educational integration.

6.1 Virtual accuracy

The professor mentioned that the heart model and beating motion are quite genuine. In systole phase, the heart will actually contract toward the center as shown in the presented model. However, from the front view, the blood should enter the ventricular chamber from the left vessel instead of the right. He also mentioned that the heart vessels or veins should be thinner.

In addition, the doctor mentioned that it will be beneficial to see the blood flow enters and leaves the heart in the vessels. That way the students can observe the blood pressure in the vessels as well. Furthermore, the vessels conveying the blood into the heart should be bigger than the one conveying the blood from the heart to the whole body.

6.2 Functionalities

In term of functionalities, he stated that it would be more promising to observe real patient's heart models from CT or MRI scans. Moreover, there are two cases of CVD, namely irregular and inconsistent beating, that he believed would be helpful for the students to visualize on this platform. Irregular beating when the heart begins contracting without fully released, is preventing the blood from filling the heart completely. If this happens the amount of blood being pumped out will be less than normal. Meanwhile, Inconsistent beating is when some sections of the heart aren't beating at the same rate of the rest causing inconsistent blood pressure. With the feature that allows the students to select the beating parts, these cases can be simulated easily.

6.3 Educational Integration

As for the aspect of integrating this module into medical education, he agrees that the platform got potentials and can be very helpful. Especially, if the module has been proven to be capable of simulating the normal and abnormal cases, i.e. irregular and inconsistent beating, the students will benefit greatly.

Chapter 7

Discussion and future work

As a result, the 3D human heart and blood simulation proof-of-concept engine proposed in this research has shown a considerable amount of potentials. The platform can simulate the deformable heart model, blood flow and the interaction between the two in an interactive environment, additionally, with a feature to manipulate how the heart should behave in the cardiac cycle. With all of these in mind, the concept of using PBD to create an educational oriented system has been proven plausible. Thus, with time and effort, such an innovative tool is absolutely possible.

There are still a number of limitations in this research. Further study has to be done to elevate the result to be even more realistic and educative, i.e. utilizing the CT or MRI scan of the patient' heart for a more detailed and precise model, and simulating abnormal cases in comparison with the normal one as stated in the previous section. More noticeably, the interaction between the heart model and blood flow is still one-sided. The interaction presented in this research only considers the forces that the deformable object applies to the fluid particles, but the forces that the fluid particles apply to the object is still overlooked. This fluid and deformable object interaction issue is still ongoing in the field of PBD.

Furthermore, a well designed graphic user interface and parallel computing should be included in the program. Since the main users of this program are the medical students and professors, the interface should be designed in compliance with their needs and specifications. Also, to improve their experiences, better rendering methods, for instance, shading, or texture mapping, should be added to the interface to deliver a more realistic simulation. Besides, even though PBD is already a prompt method and this system has already implemented CPU parallelism, utilizing GPU will still dramatically accelerated the computation time.

Acknowledgment

First and foremost, I would like to express my deep and sincere gratitude to my research supervisor, Professor Makoto FUJISAWA, head of Physics-based Computer Graphics or PBCG lab for his guidance, support, and warm welcome. Especially, his patience and understanding had helped me in so many ways and for that, I'm extremely grateful.

Furthermore, I would like to express my gratitude to Professor Mashiko MIKAWA for his encouragement and valuable feedbacks he gave in our joint seminar, my completion in this research cannot happen without him.

Moreover, I would also like to extend my gratitude to Professor Thanongchai SIRIAPISITH, Medical Professor, Department of Radiology, Faculty of Medicine, for all of his precious opinions and suggestions he gave in both of the interviews conducted in this research. He participated in this research out of his generosity without asking anything in return.

Finally, I am extremely grateful for my friends and family for their love and caring. My heartfelt thanks.

References

American Heart Association (Sept. 2016a). *Peripheral Vascular Disease: Types, Causes, and Risk Factors*. URL: https://www.heart.org/en/health-topics/cardiomyopathy/what-is-cardiomyopathy-in-adults.

— (Mar. 2016b). *What Is Cardiomyopathy in Adults?* URL: https://www.heart.org/en/health-topics/cardiomyopathy/what-is-cardiomyopathy-in-adults.

— (May 2018). *What is Cardiovascular Disease?* URL: https://www.heart.org/en/health-topics/consumer-healthcare/what-is-cardiovascular-disease.

Baillargeon, Brian, Nuno Rebelo, David D Fox, Robert L Taylor, and Ellen Kuhl (2014). "The living heart project: a robust and integrative simulator for human heart function". In: *European Journal of Mechanics-A/Solids*, 48, pp. 38–47.

Bender, Jan, Matthias Müller, and Miles Macklin (2015). "Position-Based Simulation Methods in Computer Graphics". In: *EG 2015 - Tutorials*. Ed. by M. Zwicker and C. Soler. The Eurographics Association. DOI: 10.2312/egt.20151045.

Bender, Jan, Matthias Müller, Miguel A Otaduy, Matthias Teschner, and Miles Macklin (2014). "A survey on position-based simulation methods in computer graphics". In: *Computer Graphics Forum*, 33(6), pp. 228–251.

Berberoğlu, Ezgi, H Onur Solmaz, and Serdar Göktepe (2014). "Computational modeling of coupled cardiac electromechanics incorporating cardiac dysfunctions". In: *European Journal of Mechanics-A/Solids*, 48, pp. 60–73.

Betts, J.G. (2013). *Anatomy & Physiology*. OpenStax College, Rice University. URL: https://books.google.co.jp/books?id=st7ZvAEACAAJ.

Caballero, Andrés, Wenbin Mao, Liang Liang, John Oshinski, Charles Primiano, Raymond McKay, Susheel Kodali, and Wei Sun (2017). "Modeling left ventricular blood flow using smoothed particle hydrodynamics". In: *Cardiovascular Engineering and Technology*, 8(4), pp. 465–479.

Doost, Siamak N., Dhanjoo Ghista, Boyang Su, Liang Zhong, and Yosry S. Morsi (Aug. 2016). "Heart blood flow simulation: a perspective review". In: *BioMedical Engineering OnLine*, 15(1), pp. 101:1–101:28. ISSN: 1475-925X. DOI: 10.1186/s12938-016-0224-8. URL: https://doi.org/10.1186/s12938-016-0224-8.

Gonzales, Matthew J, Gregory Sturgeon, Adarsh Krishnamurthy, Johan Hake, René Jonas, Paul Stark, Wouter-Jan Rappel, Sanjiv M Narayan, Yongjie Zhang, W Paul Segars, et al. (2013). "A three-dimensional finite element model of human atrial anatomy: new methods for cubic Hermite meshes with extraordinary vertices". In: *Medical Image Analysis*, 17(5), pp. 525–537.

Klepach, Doron, Lik Chuan Lee, Jonathan F Wenk, Mark B Ratcliffe, Tarek I Zohdi, Jose L Navia, Ghassan S Kassab, Ellen Kuhl, and Julius M Guccione (2012). "Growth and remodeling of the left ventricle: a case study of myocardial infarction and surgical ventricular restoration". In: *Mechanics Research Communications*, 42, pp. 134–141.

Macklin, Miles, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim (2014). "Unified particle physics for real-time applications". In: *ACM Transactions on Graphics (TOG)*, 33(4), pp. 153:1–153:12.

Monaghan, Joe J (1992). "Smoothed particle hydrodynamics". In: *Annual Review of Astronomy and Astrophysics*, 30(1), pp. 543–574.

Mozaffarian, Dariush, Emelia J Benjamin, Alan S Go, Donna K Arnett, Michael J Blaha, Mary Cushman, Sandeep R Das, Sarah de Ferranti, Jean Pierre Després, Heather J Fullerton, et al. (2016). "Heart disease and stroke statistics-2016 update a report from the American Heart Association". In: *Circulation*, 133(4), pp. e38–e48.

Müller, Matthias, David Charypar, and Markus Gross (2003). "Particle-based fluid simulation for interactive applications". In: *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, pp. 154–159.

Müller, Matthias, Bruno Heidelberger, Marcus Hennix, and John Ratcliff (2007). "Position based dynamics". In: *Journal of Visual Communication and Image Representation*, 18(2), pp. 109–118.

Müller, Matthias, Simon Schirm, and Matthias Teschner (2004). "Interactive blood simulation for virtual surgery based on smoothed particle hydrodynamics". In: *Technology and Health Care*, 12(1), pp. 25–31.

Nakashima, Kazutaka, Yuki Koyama, Takeo Igarashi, Takashi Ijiri, Shin Inada, and Kazuo Nakazawa (2016). "Interactive deformation of structurally complex heart models constructed from medical images". In: *Proceedings of Eurographics 2016*, pp. 49–52.

Narang, Akhil, Shashank S Sinha, Bharath Rajagopalan, Nkechinyere N Ijioma, Natalie Jayaram, Aaron P Kithcart, Varsha K Tanguturi, and Michael W Cullen (2016). "The supply and demand of the cardiovascular workforce: striking the right balance". In: *Journal of the American College of Cardiology*, 68(15), pp. 1680–1689.

Palyanov, Andrey, Sergey Khayrulin, and Stephen D Larson (2016). "Application of smoothed particle hydrodynamics to modeling mechanisms of biological tissue". In: *Advances in Engineering Software*, 98, pp. 1–11.

Peer, Andreas, Markus Ihmsen, Jens Cornelis, and Matthias Teschner (July 2015). "An Implicit Viscosity Formulation for SPH Fluids". In: *ACM Transactions on Graphics (TOG)*, 34(4), pp. 114:1–114:10. ISSN: 0730-0301. DOI: 10.1145/2766925. URL: http://doi.acm.org/10.1145/2766925.

Platis, Nikos and Theoharis Theoharis (2003). "Fast ray-tetrahedron intersection using plucker coordinates". In: *Journal of Graphics Tools*, 8(4), pp. 37–48.

Pruthi, Sandhya (2018). *Heart disease*. URL: https://www.mayoclinic.org/diseases-conditions/heart-disease/symptoms-causes/syc-20353118 (visited on 03/22/2018).

Rheumatic Heart Disease Australia (Mar. 2019). *What is Rheumatic Heart Disease?* URL: https://www.rhdaustralia.org.au/what-rheumatic-heart-disease.

Selmi, Marwa, Hafedh Belmabrouk, and Abdullah Bajahzar (2019). "Numerical Study of the Blood Flow in a Deformable Human Aorta". In: *Applied Sciences*, 9(6), pp. 1216:1–1216:11.

Springel, Volker (2010). "Smoothed particle hydrodynamics in astrophysics". In: *Annual Review of Astronomy and Astrophysics*, 48, pp. 391–430.

Townsend, Nick, Lauren Wilson, Prachi Bhatnagar, Kremlin Wickramasinghe, Mike Rayner, and Melanie Nichols (2016). "Cardiovascular disease in Europe: epidemiological update 2016". In: *European Heart Journal*, 37(42), pp. 3232–3245.

Tyfa, Zbigniew, Damian Obidowski, Piotr Reorowicz, Ludomir Stefańczyk, Jan Fortuniak, and Krzysztof Jóźwik (2018). "Numerical simulations of the pulsatile blood flow in the different types of arterial fenestrations: Comparable analysis of multiple vascular geometries". In: *Biocybernetics and Biomedical Engineering*, 38(2), pp. 228–242.

World Health Organization (May 2017). *Cardiovascular diseases (CVDs)*. URL: https://www.who.int/en/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds).