TECHNISCHE UNIVERSITÄT MÜNCHEN

Computer Graphics and Visualization Group
Computer Games Laboratory WS 17

Blood Invaders

Authors: Markus Ansorge, Philipp Holl, Christian Schnelzer, Robin Otto
Date: January 9, 2018

Contents

1 Game Proposal 3
1.1 Game Description 3
1.1.1 Artstyle 3
1.1.2 Gameplay 3
1.2 Technical Achievement 5
1.2.1 Correct Representation of the Immune System 5
1.2.2 Asynchronous Multiplayer 5
1.3 Schedule 6
1.3.1 Layered Task Breakdown 6
1.3.2 Development Schedule 9
1.4 Assessment 11
2 Physical Prototype 12
2.1 Prototype 12
2.1.1 General 12
2.1.2 Players 12
2.1.3 Enemies 13
2.2 Experience 14
2.3 Design Revisions 15
2.3.1 Prototype 15
2.3.2 Critiques 16
3 Interim Report 17
3.1 Progress 17
3.2 Game description 17
3.2.1 Overview 17
3.2.2 Characters 18
3.2.3 Fluid simulation 19
3.3 Challenges 21
3.3.1 Minor Challenges 21
3.3.2 Major Challenges 21
4 Alpha Report 23
4.1 Progress 23
4.1.1 What is missing? 23
4.1.2 What is done? 23
4.1.3 What's next? 24
4.2 Game description 24
4.2.1 Immune System & Level Progression 24
4.2.2 Improved Models 25
4.2.3 Custom and procedural shaders 26

1. Game Proposal

1.1. Game Description

As this year's topic of the Computer Games Laboratory course is T̈ogether, we decided to make a coop game. At its core, it is a Shoot Ém Up that is played by two players locally on the same computer. The first player assumes the role of a bacterium while the second player assumes the role of a virus. The players find themselves in a human body and it is their goal to fight against the body's immune system. The immune system as an enemy is a core element of our game. The enemies will react to player actions like a real body's immune system would react to a bacterium or virus and so create a dynamic challenge that can be different every time you play it. The bacterium and the will also differ from each other visually and functionally. While the bacterium is bigger and bulkier and therefore a bit slower, the virus is a swarm of multiple smaller viruses and therefore faster. The goal of the game is to infect the whole body. In order to achieve this, every level has to be beaten by defeating the boss at the end of every level. If both players die during a level, they can replay the level until they succeed. There won't be a G̈ame Over T̈hat forces you to start completely from the beginning.

1.1.1. Artstyle

The game is situated in the human body and will be divided into separate levels. These levels correspond to different body parts or organs. At the end of each level, there is a boss, which needs to be defeated in order to complete the level. The overall art style of our game is not photo-realistic, but the players, as well as the enemies and the environments, resemble their real-world counterparts. The game will be 2.5D meaning that the assets and environments have 3 dimensions, while the gameplay will be restricted to two dimensions.

1.1.2. Gameplay

In general, the gameplay takes place in a typical Platformer or Shoot Ém Up environment, where the players can traverse the screen either from left to right or from top to bottom. So the gameplay will be restricted to those 2 dimensions. After a short introduction video on how the bacterium and the virus enter the body, the players gain control over their characters and start to progress through the level. Some parts of the game feature an autoscroll mechanic that forces the players to progress through the level at a certain speed by automatically moving the environment forward. If the players can't keep up, they lose health. Other sections of the game allow the players to progress at their own speed.
The bacterium and the virus will not only look different but will also differ in their behavior and abilities.
The bacterium is relatively large compared to the virus. Therefore it has a bit slower movement which it compensates with a higher hp pool. Its size is also an indicator of how much

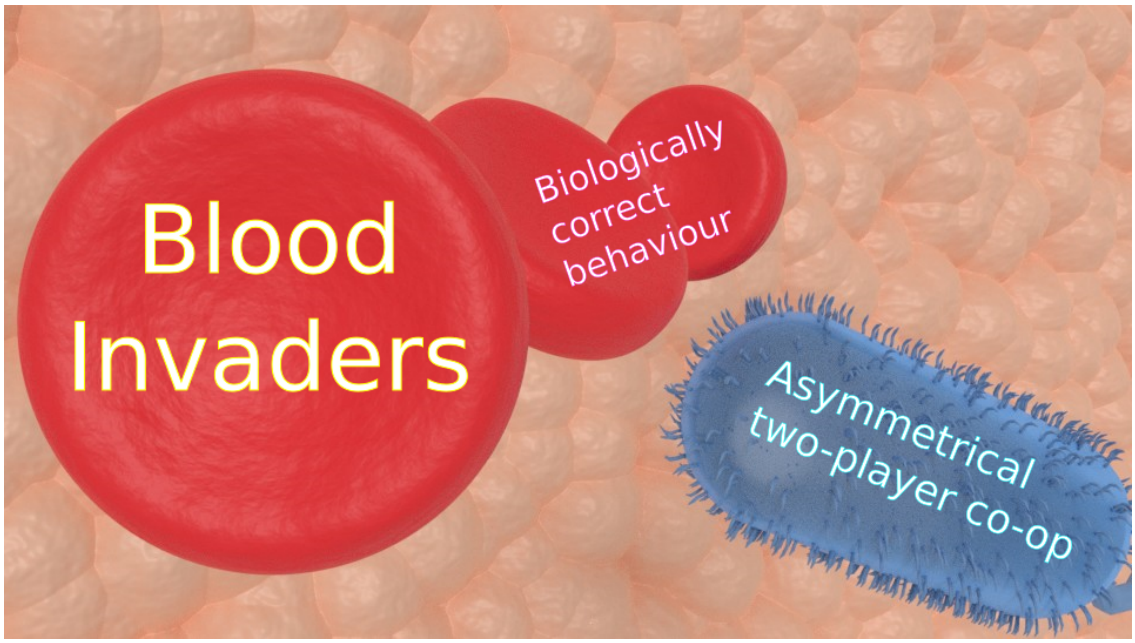
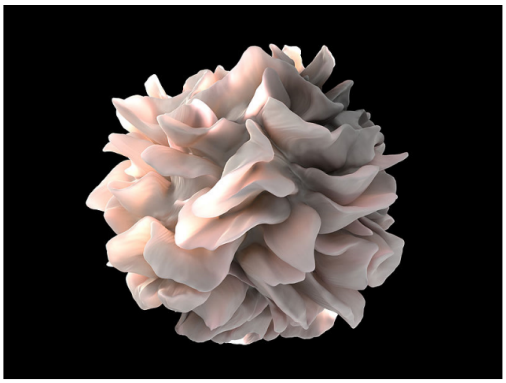


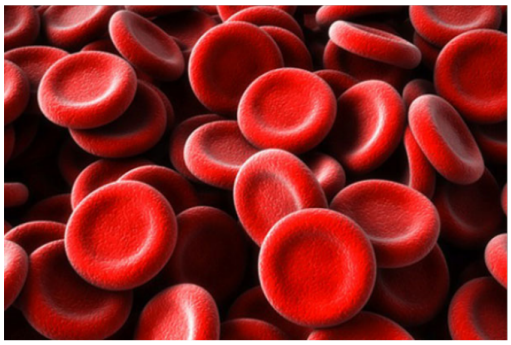
Figure 1 *Big Idea Bullseye* - Main concepts of the game

health it has remaining. So if the bacterium loses hit points, it will shrink in size. The bacterium's first ability is to eat certain cells or parts of certain cells in order to restore some health. The second ability is a dash, that allows him to quickly get out of critical situations.
The virus, on the other hand, behaves differently. Since a single virus is relatively small compared to a bacterium, the virus actually consist of a swarm of viruses. The number of viruses in the swarm indicates the health of the virus. If there are no more viruses left, the virus is dead. The virus has the ability to infect healthy body cells in order to create new viruses and therefore restoring health. The virus' second ability allows him to temporarily mutate and so gain temporal immunity against certain enemies. The abilities of the virus and the bacterium will be on a cool down before they can be used again.
Killing enemies will also charge the duo's ultimate ability which can be activated once enough enemies have been killed. There are two ultimate abilities. The first one creates a p̈owerlinkb̈etween the virus and the bacterium which kills any enemy that touches it. The second one allows the bacterium and the virus to temporarily merge and create a sort of ẗankẅhere the bacterium is responsible for steering and the virus is responsible for the shooting.
Enemies resemble cells that the body's immune system uses in order to fight bacteria or a virus. Their combat behavior is inspired by their behavior in a real human body. Cells that eat bacteria or viruses try to collide with the players. Other cells that produce antibodies, shoot those anti-bodies towards the players. Other enemies will serve as messengers that alert other parts of the immune system and therefore call forth additional enemies.

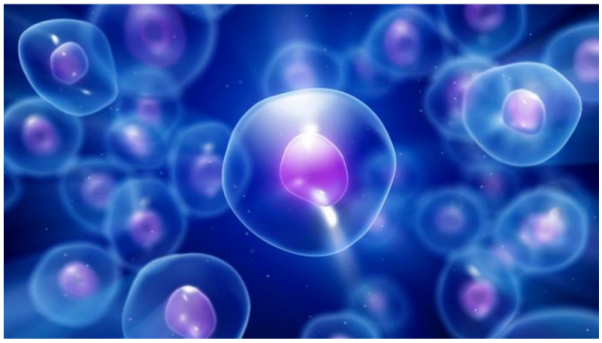
(a) Bacterium



(b) Dendridic cell



(c) Red blood cells



(d) Cells to invade

Figure 2 The art style of the game is *Scientific visualization*, similar to the CG depiction in documentaries. Simplified shapes, false colors and PBR shaders are going to be used.

1.2. Technical Achievement

The technical achievement describes the challenges arising with the implementation of the game idea.

1.2.1. Correct Representation of the Immune System

In case of Blood Invaders, one technical difficulty is to implement a precise and accurate representation of the immune system of the human body. It should be recognizable, how the Adaptive Immune System reacts to intruders like viruses and bacteria that are trying to harm the body. I.e. which cells defend in what kind of situation and how are they try to remove the attackers. In addition to this, what are the virus' and bacterium's way to win the fight against these defenders, which abilities help them to reach their goal.

1.2.2. Asynchronous Multiplayer

The second technical difficulty in Blood Invaders will be the implementation of a smoothly working and fun-to-play asymmetric multiplayer game. One player plays the virus with his individual abilities and look, the other takes control of the bacterium, which has quite a different behavior.

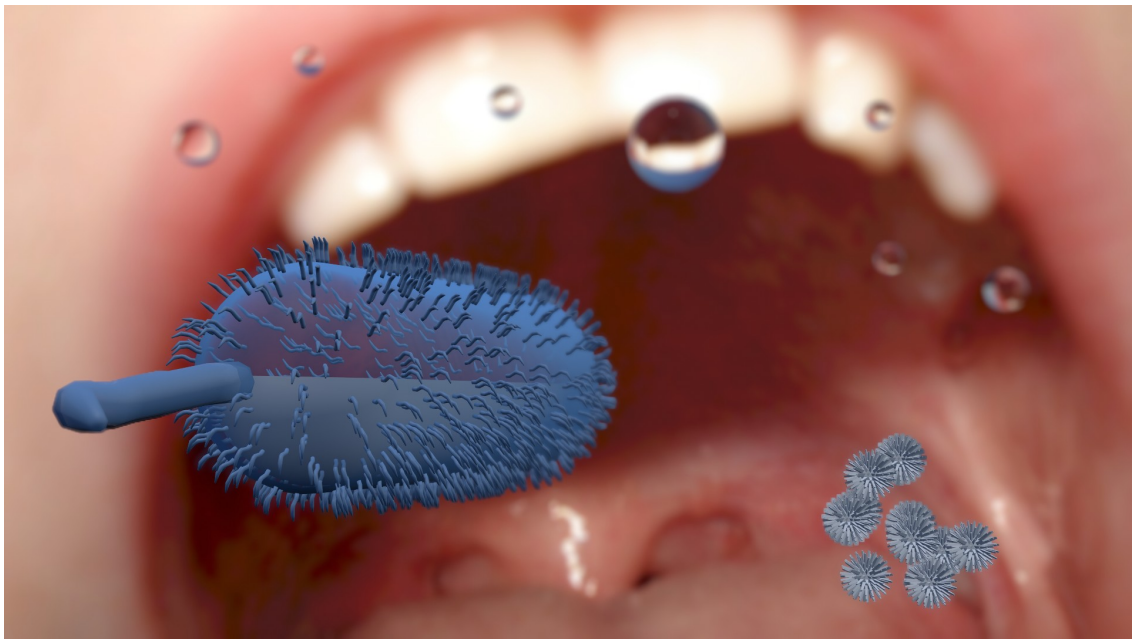


Figure 3 *Mockup:* Short intro sequences show how the invaders entered the body.

1.3. Schedule

1.3.1. Layered Task Breakdown

This section provides an overview of the different features the game should contain. The features are clustered associated to how crucial they are with respect to the game itself. For the separation we need to differentiate between five topics of features. Each functionality can either be associated with the player, the enemies, the UI, design and general issues.

Functional Minimum

The functional minimum describes the essential features needed to call the application a game.

- Player:
  - Basic ability: The players should be able to shoot some kind of projectile.
  - Basic health property: If hit by an enemy, a player can die.
- Enemy:
  - Basic behavior: the first enemies appearing in the game will be neutrophils with no abilities. They fly around randomly.
  - Basic damage chart: If hitting a player, the collision kills both, the player and the enemy.
- UI:
  - There will be no main menu at this point, only the gamescene with the current level.
- Design:
  - Thus far, the players and enemies both have a placeholder model, like a cube, and no

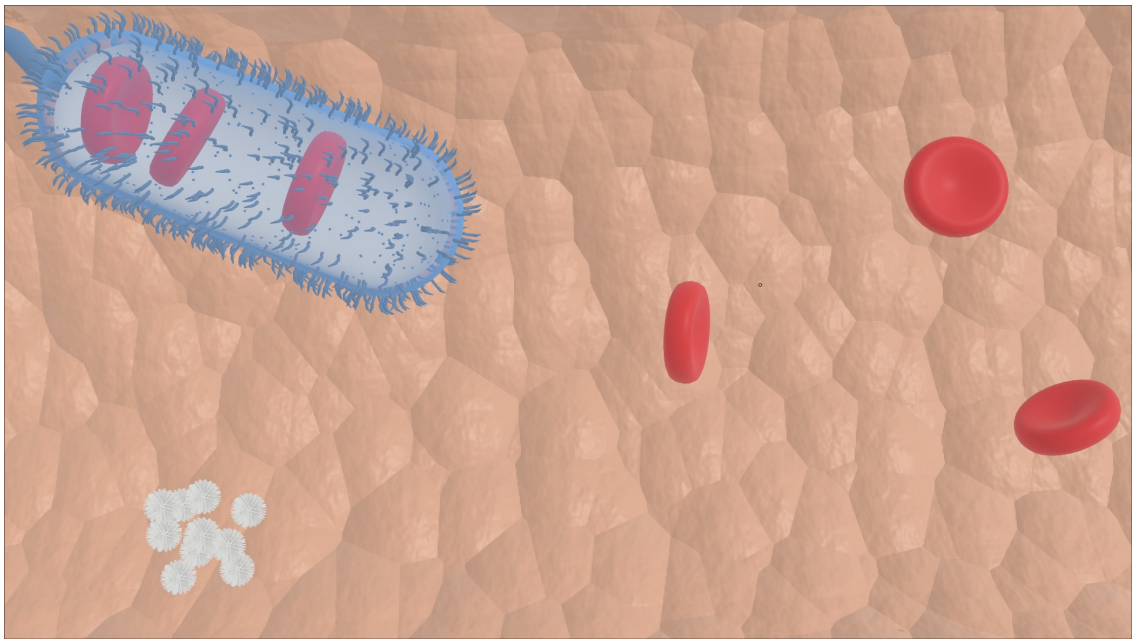


Figure 4 *Mockup:* Example scene of the game showing the bacterium and the virus swarm inside the blood stream.

animations.

- General:
  - Multiplayer: The game should be playable by two players on one machine.
  - Controls: Both players play share the same keyboard.

Low Target

The low target represents the least acceptable state of the application that can be submitted as a final result.

- Player:
  - First skill: The virus and the bacterium both have a first ability that focuses on regaining a portion of the health.
  - Different players: Both characters are distinguishable with respect to looks and abilities.
- Enemy:
  - Two kinds of enemies: Macrophages are added to the enemy spectrum. They are bigger than Neutrophils and survive longer.
  - Infectable cells: Cells, which can be infected by the virus are added to the game.
- UI:
  - Main menu: The application has a main menu which contains the options to start and end the game.
  - Level entry and exit: the existing level will have a defined start point as well as an end point, which brings you back to the main menu.
- Design:

    - Low-polygon models: Some self made character models in low resolution are available for the players. The virus will be a swarm of little viruses and the bacterium a big blob.
    - Background: the game scene has a simple background that scrolls from top to bottom or from right to left.
- General:
    - No new features for this layer.

Desirable Target

If the game complies with the requirements defined in the desirable target section, the developers have reached their predetermined goals for this project.

- Player:
    - Bacterium: Is bigger and slower than than the virus, also has more health.
    - Virus: Smaller and faster, more squishy.
    - Ultimate ability: A chargeable, synchronized super ability is available, forcing both players to interact.
- Enemy:
    - New types: Natural killer cells, who kill cells infected by the virus, as well as dendritic cells, who inform the adaptive immune system about intruders.
    - Adaptive immune system: B-Lymphocites are added to the game. These shoot loads of anti-bodies that hurt the characters.
- UI:
    - Main menu: Level selection is available in the main menu.
    - Implicit indicators: there will be hints on different status attributes, e.g. an ultimate progress indicated by a glowing bacterium or health indicated by the virus' size.
- Design:
    - Self designed characters: Higher resolution and self made animations.
    - Self designed enemies: Some enemies have self designed models with basic animations.
    - Different levels: So far, there are up to two levels with different settings. The levels have a polished design.
- General:
    - Sound: Sound effects for attacks and background music is added to the game.
    - Controls: The game is playable with controller and/or keyboard.

High Target

The requirements from the high target represent the result of the application if things take course better than expected.

- Player:
  - Quick time event: A synchronized ability will be added that is started by one player and finished by the other through a quick time event.
  - Virus: Can mutate in order to gain temporal immunity against anti bodies.
  - Bacterium: Has a new dash ability to dodge enemies.
- Enemy:
  - Boss fight: At the end of each level, there will be a boss fight.
- UI:
  - Main menu: Settings section will be added.
  - Customization options: Difficulty settings, different character models to choose from.
- Design:
  - Cutscenes: Amongst others, an intro cutscene will be added.
  - Fluid animations: Character and enemy animations will be more polished and fluid.
  - Levels will contain scripted events.
- General:
  - Sound: Different sound tracks for the different levels.
  - Perspective: Multi-task sequences where both players need to contribute. These sequences will be in 3D perspective.

Extras

The extras section contains features that would be nice-to-have as addons and future work, but which do not fit in the scope of this project.

- Player:
  - No additional features.
- Enemy:
  - Biological correctness: The game will contain are more accurate and detailed representation of the immune system.
- UI:
  - No new features.
- Design:
  - Random level generator: levels are automatically created for the different environments.
  - Level editor: Blueprints and a level editor will be available.
- General:
  - Publishment: The game will be published on Steam.

1.3.2. Development Schedule

This schedule (Fig. 5 and 6) presents how we plan to implement the game. At the end of the implementation phase our target is to have all desirable target items to be finished. The

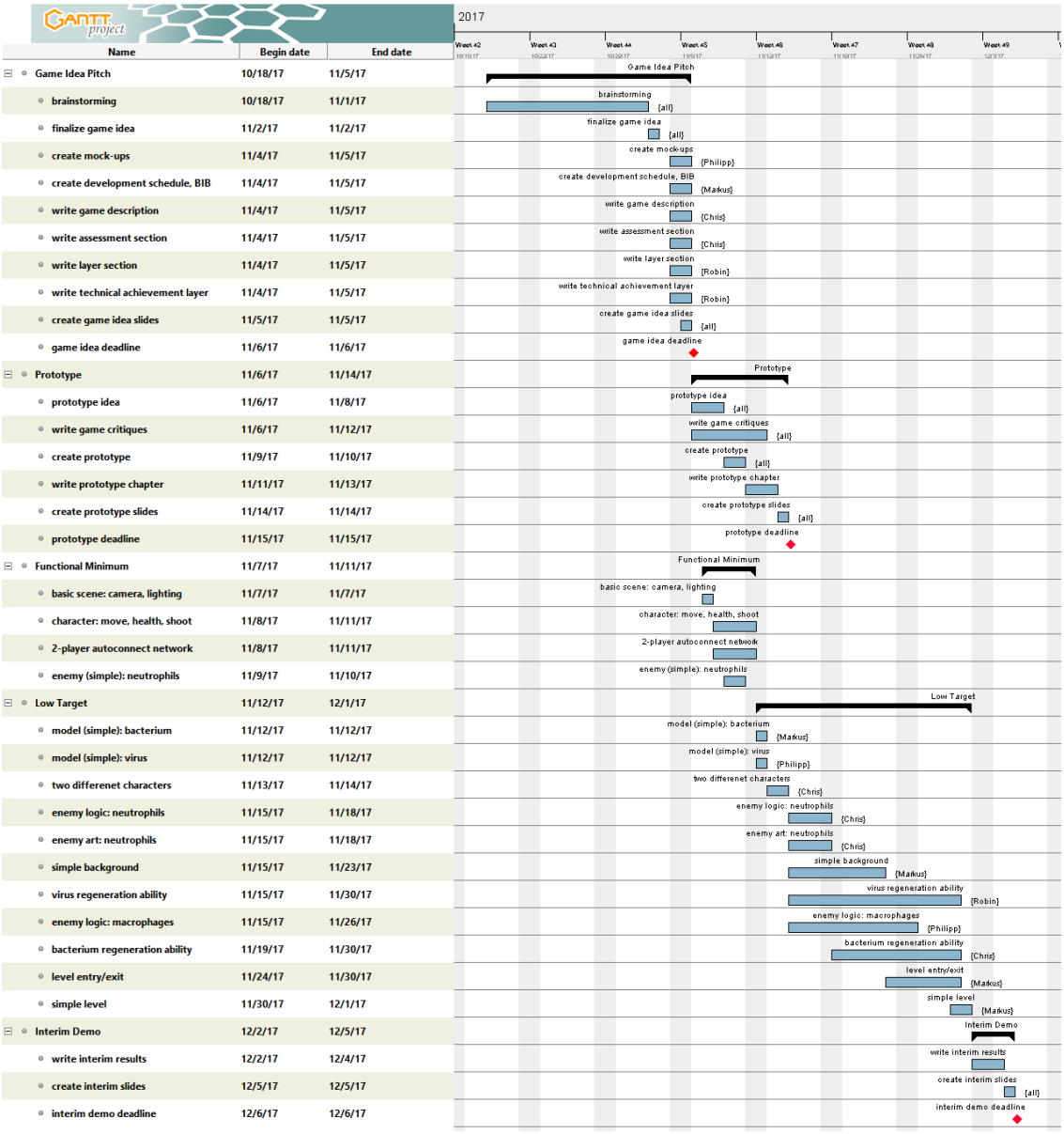


Figure 5 Part one of the development schedule

schedule includes the assignment of all four team members to different tasks. This is still work-in-progress and will certainly change during the implementation phase. Therefore, this is intended as a guideline that will help us to keep track of our work and rather not as a binding and final schedule.

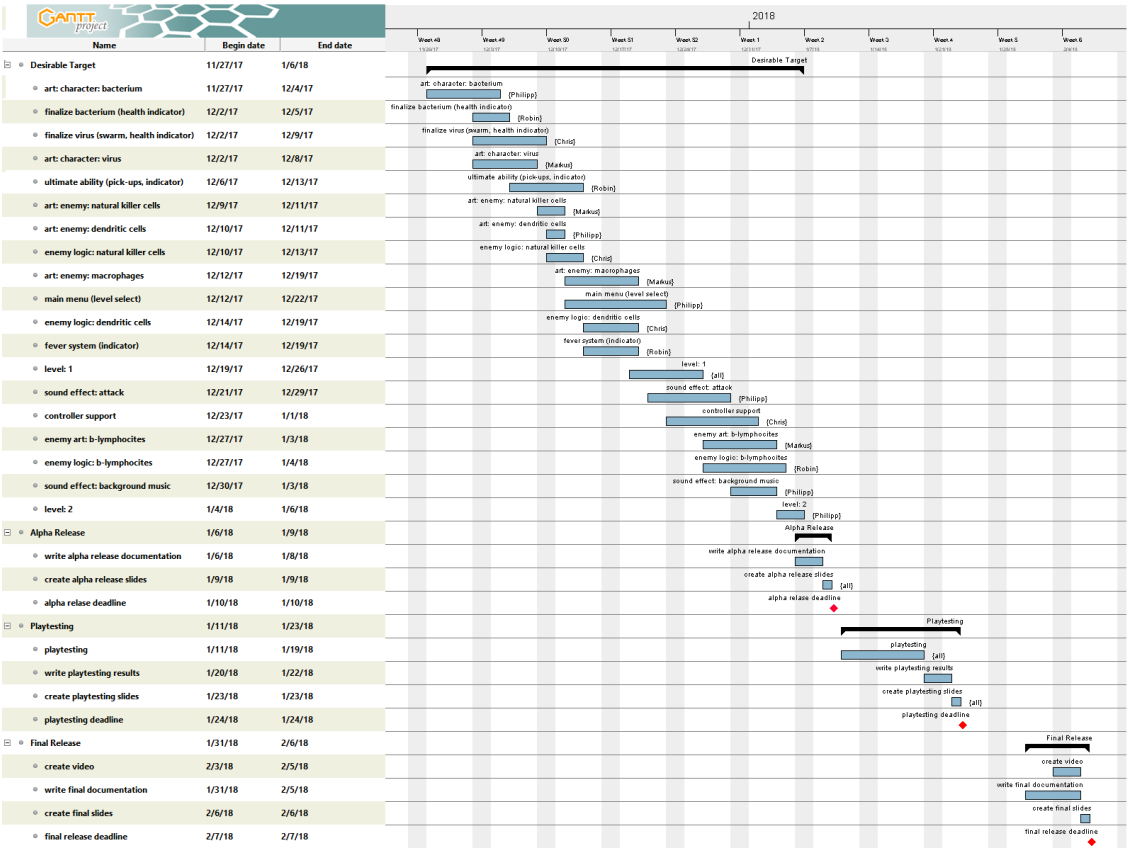


Figure 6 Part two of the development schedule

1.4. Assessment

Generally speaking, we want to create a Shoot 'Em Up game that looks and plays reasonably well, considering the time we have for the Computer Games Laboratory Course. The aspects that make our game unique and stand out from the ocean of Shoot 'Em Up games out in the wild are the following: With a bacterium and a virus as coop partners, we create a unique pairing that offers possibilities for interesting game mechanics and player-player interactions inspired by actual biology and chemistry.

Our setting inside a human body is already naturally different from the vast majority of Shoot 'Em Up games that are at least in some form space or spaceship themed. This allows for interesting and unique visuals that can make our game unique and stand out without the need to look perfectly polished.

Last but not least our enemy system that is inspired by the immune system of a real body provides a dynamic gameplay experience that can produce different scenarios depending on how the players play, without any need of additional scripting for us.

We are confident that our game will offer a fun an unique experience if we succeed to execute these three core aspects well.

2. Physical Prototype

2.1. Prototype

The main reason for creating a physical prototype for any kind of application is to test its core idea. In case of a game, the prototype should carve out the core gameplay and test its functionality. Applied to our project "Blood Invaders", it is important to try out whether our main element, the enemy system inspired by the real immune system, works with respect to our game idea. Therefore, we designed and build a board game whose mechanics and rules will be described in the following section.

2.1.1. General

The game is designed for two players. One player plays the virus, the other one plays the bacterium. Together they have to survive against the immune system until the boss of the level appears. Once they have beaten the boss, the players win. If both players die or the fever level of the body becomes too high, the players lose. The enemies behave according to a set of rules and are not controlled by a player. The game is played on a board consisting of hexagons. The players start on one side while new enemies approach them from the other side of the board. The game is turn based. It starts with the players' turn, where the virus and the bacterium get to perform their actions. After that, it's the enemies' turn. The players' and the enemies' turn keep alternating until the game is over. Players are able to shoot at enemies, however they can only shoot horizontally on the board. A player can only target enemies that are located on the same horizontal line as the enemy. We simulate the fact that enemies that are further away, are harder to hit, with dice rolls. Enemies that are further away require the players to roll higher in order to hit.

2.1.2. Players

This section describes the two player characters and their different traits and abilities.

The bacterium is the bigger one of two player characters and therefore occupies two spaces on the board and has 3 hit points. Every turn it can perform 2 actions. For the first action, the player that controls the bacterium has 3 possibilities:

- move up to two spaces forward in the direction he is facing
- move up to two spaces backward in the opposite direction
- turn by 60 degrees

For the second action he shoot once. If he is standing diagonally on the board, he has the possibility to shoot at either of the lines he is occupying. The bacterium has a range of up to 12 hexagons. If he decides to shoot he has to roll a dice. For every "eye" he rolls, the range

of the shot will increase by 2, meaning that if he rolls a "1" he will hit an enemy that is two or less hexagons away. If he rolls a "2" he will enemy that is four or less hexagons away and so on. If he hits he decreases the hit points of the targeted enemy by one.

The virus is smaller and only occupies one hexagon on the board and also has 3 hit points. It can also perform actions. For the first action it can move 1 space in any direction. For the second action he can shoot at enemy that is on the same horizontal line as the virus. The virus has a range of up to 6 hexagons. He also has to roll a dice if he tries to shoot an enemy. Every eye that he rolls increases the range of his shot by one, meaning that if he rolls a 1 he will hit an enemy that is 1 hexagon away. If he rolls a 2 he will hit an enemy that is 2 or less hexagons away and so on. If the hits, the targeted enemy loses 1 hit point The virus also has a special ability. He can infect infectable body cells if he stands on the same hexagon as the infectable cell. If he does that, two different things happen. First of all, the virus can no longer perform his normal actions during his turn, but his actions replaced by a dice roll. Once he reaches a cumulative "eyecount" of 10, the infection was successful. The infected cells gets destroyed and the virus regains 2 health points. While the virus is infecting a cell, he can only be damaged by natural killer cells and macrophages stop targeting until the infection is completed. The second thing that happens is that 2 natural killer cells appear on the board. Their functionality will be explained in the enemy section.

2.1.3. Enemies

This section describes the different kinds of enemies, how they behave and when they spawn.

Our most basic enemies are the neutrophils, which have one hit point. They’re movement is pseudo-random is determined by a dice roll:

- if the roll is a 1 or 2: move one hexagon forward and one hexagon upward
- if the roll is a 3 or 4: move two hexagons forward
- if the roll is a 5 or 6: move one hexagon forward and one hexagon downward

A single dice rolls determines the movement for all neutrophils one the board. If the neutrophils are directly next to player at the start of the enemy turn, they always move on the space that the player is standing on, independent of the dice roll. If a neutrophil is on the same space as a player the neutrophil does 1 damage to the respective player and dies afterwards. In the first turn of the game 3 neutrophils spawn and are randomly distributed in the last third of the board. From the second turn on, a new neutrophil spawns every enemy turn. There are six different spawn positions. Where the new neutrophil spawns is determined by a dice roll.

Our next enemies are the macrophages, which have 2 hit points. They always move 2 hexagons in the direction of the nearest player. Macrophages can damage players even if

they 1 field away from the player. However they can only damage each player once a turn. In turn 4, two macrophages spawn. Where they spawn is determined by a dice roll. From there on out 1 new macrophage spawns every other turn. Once the mast cell appears on the board macrophages stop spawning. However already existing macrophages stay on the board.

Another enemy are the infectable cells, which have 1 hit point. They always move 2 spaces forward horizontally. They can not damage the player. If they collide with the bacterium the move either upwards or downwards. If they collide with virus, the virus can infect them if he wants to. In turn 6, 3 infectable cells spawn. Their spawnposition is determined by a dice roll. From there on out, 2 infectable cells spawn every 3 turns until the game is over.

A special enemy are the natural killer cells, which have 1 hit point. They always move 3 spaces in the direction of an infected cell. They have to move around the bacterium. If they hit the infected cell before the infection is completed, the infection stops, the infected cell gets destroyed and the virus loses 2 health points. Also all natural killer cells are removed from the field. If the infection is completed before the natural killer cells can reach the infected cell, all natural killer cells are removed from the field. Natural killer cells can only spawn when the virus infects an infectable cell. In this occasion two natural killer cells spawn and their spawn location is determined by a dice roll.

The boss of the game is the mast cell, which has 3 hit points. Every turn, the mast cell spawns a chemical. Afterwards it moves two hexagons vertically. If it reaches the end of the board it changes direction until it hits the other end of the board. It spawns in turn 12 in the corner of the board.

Our last enemies are the chemicals, which have 1 hit point. They always move 2 hexagons forward horizontally. If they leave the board, the fever level of the body increases by 1. If they fever level of the body reaches 2, the players lose. Chemicals can only be spawned by the mast cell.

2.2. Experience

Our overall impression of testing our prototype was pretty positive. In comparison to our expectation, it was much easier to make the game fun and balanced. Both the virus' as well as the bacterium's role and influence in winning the level were approximately the same. This made it possible for us to test the crucial components for both characters.
We learned, that a big challenge will be to design the levels and the immune system in a way, that the spawning behavior together with the damage chart of the enemies is both fun and challenging. In our prototype, it took us a lot of iterations until we figured out a decent spawning sheet. Our final result (see Fig. 7) gave the players the opportunity to approach the enemies as well as the boss in different ways but still demanded a great deal from the virus and bacterium. This represents exactly what we want to achieve in the real game: A

challenging but fun experience for everyone!
Another thing we learned while playing our prototype was, that a small number of different abilities for the bacterium and the virus are sufficient, since the immune system will be complex enough to still ensure varying experiences for the player. In our setup, we used one individual ability for each player together with a sequence that required both characters to cooperate, which was enough to create a lot of variation and at the same time prevented the players from losing track. Therefore, when implementing the game, we will focus on getting few items right which fit the individual characters well and can be used in a reasonable way against the immune system.

	Number of enemies				
Round	Neutrophil	Macrophage	Infect. Cells	Mast Cell	Chemicals
1	4				
2	1				
3	1				
4		2			
5	1				
6	1	1	3		
7	1				
8	1	1			
9	1		2		
10	1	1			
11	1				
12	1		2	1	
13	1				1
14	1				1
15	1		2		1
16	1				1
17	1				1
18	1		2		1
19	1				1

Figure 7 Picture playing the prototype.

2.3. Design Revisions

In this section we will explain which design revisions and changes we have derived from playing our physical prototype and receiving critiques from the other groups participating in the Computer Games Laboratory.

2.3.1. Prototype

For the most part, we were rather surprised how well our ideas worked out as a physical prototype. The asymmetric gameplay with two different roles and the combat against the immune system with a predefined and rather complex ruleset, resulted in a fun experience right from the first time we tried our prototype. Therefore these two concepts still hold as central design concepts in our game.
During testing the prototype we found out that putting more work into the level details in terms

of course of events and enemy spawn behavior is important to ensure a consistent flow for the players. Because of that we want to focus on at most two levels, maybe even only a single level, that is fleshed out. With this in mind, the random level generation will move even further back in our priority list and development schedule.

2.3.2. Critiques

First of all we want to thank the other teams for their honest and constructive feedback. Overall the asymmetric gameplay seems to be the most liked feature in our game. Furthermore, the other teams seem to like our idea of simulating the humane immune system together with the setting of the game inside the body.

One aspect that was frequently requested, is to constrain the players to cooperate more often, since this is one of the core mechanics of the game. The others suggested to enforce this through environment and/or enemies.

A more detailed implementation of the cooperation component is the cell infection sequence, where the virus has to infect a cell while the bacterium protects it from the natural killer cells. We also tested this component in the prototype in order to validate that the concept we have in mind will actually work out in the game.

3. Interim Report

3.1. Progress

In this section the progress of our game will be described in more detail. This will be based on our goals for each layer which we stated in the first chapter. Overall we managed to finish the largest part of the first two layers and some aspects of the desired target. However, we struggled with some parts (see Chapter Challenges) and will have to invest more work into these parts.

Player The tasks regarding the virus swarm and bacterium proved to be more work than we anticipated. While we managed to realize a shooting mechanic and a health system, we could not complete the low level task of implementing health regain skills.

Enemy Regarding the enemies we finished most of the the first two layers. Neutrophils can spawn, move, deal damage and die. In particular the planned basic movement algorithm was replaced by advanced fluid simulation (see Chapter Fluid simulation). The macrophages currently behave similar except for their movement. They always target the closest player.

UI The progress of the tasks regarding the UI are on point of the second layer. The main menu provides a functional Play and Quit button and a placeholder Options button. One can click on a neutrophil or red blood cell in order to display biological information about them. The level can currently not be completed. However, the death of both players leads back to the main menu.

Design All first and second layer tasks with respect to the Design have been successfully implemented. Both player characters as well as both enemies have low-poly models and we already experimented with some basic animations. The background is a seamless mesh that slowly scrolls from right to left.

General We faced some challenges regarding the General tasks. In particular, we could not manage to have two players play on the same keyboard. However, two players can still play together with the help of a controller. On the other hand a shot sound has already been added, which is part of the desirable target.

3.2. Game description

3.2.1. Overview

The current prototype features self-designed models with very basic animations. Some of the models are already in a final state while others will need some reworking.

On startup, a main menu is displayed from which the game can be started. A screenshot of the main menu is shown in figure 8. Cells and other elements can be seen in the background.

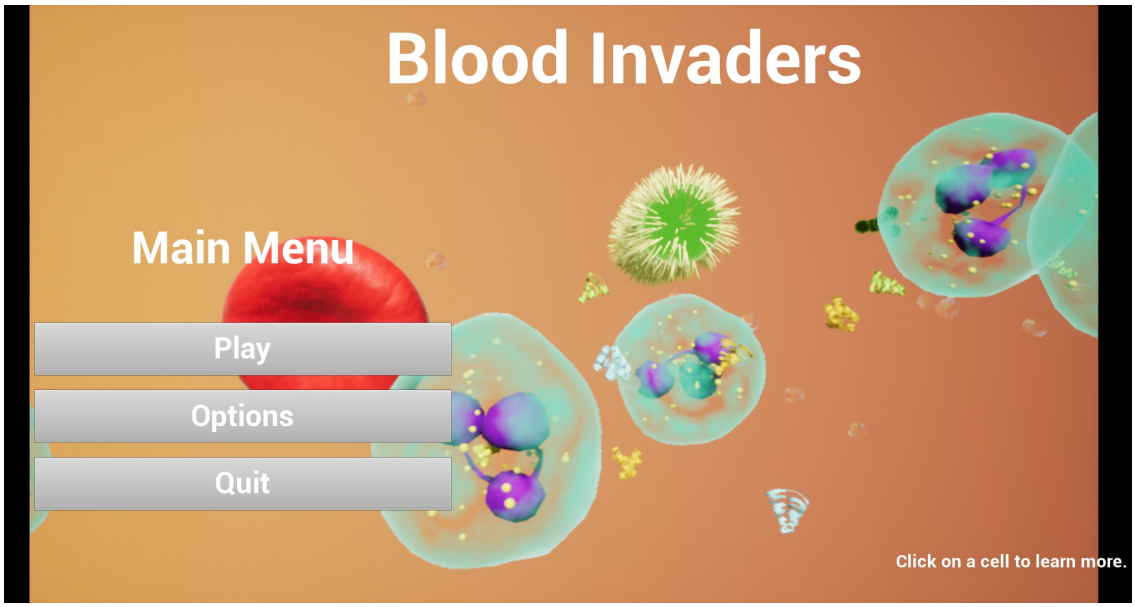


Figure 8 Screenshot of the main menu

By clicking on one, the player is presented with biological information about the object in question. This already embeds some educational possibilities into the game.

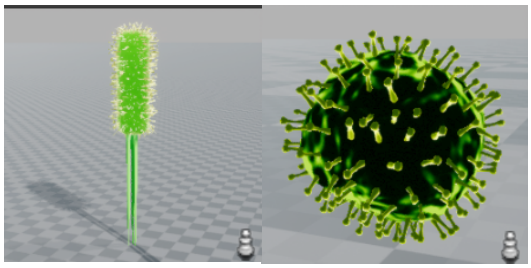
The *play* button loads the default level in which two players play on one computer, one controlling the bacterium with the keyboard and the other controlling the virus swarm with a controller. The controls include movement in four directions and a button for shooting at this point. The movable area is restricted to the bounds of the camera which constantly moves to the right, forcing the players to progress within the level.

The first level takes place in a blood vessel as indicated by the background. Enemies and other cells spawn in intervals from the right and approach the player. The movement of enemies, and all objects in principle, is the combination of two factors.

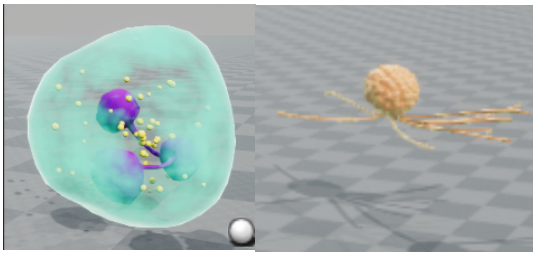
1. Intent: The directed self-propelling in one direction and rotation. Only certain types of enemies have a target they try to reach. For the players, this is defined by the input controls. The intent is implemented as adding forces and torques to the object.
2. Random motion in fluid: We simulate the fluid dynamics inside the blood vessel and apply the resulting forces and torques to all objects.

3.2.2. Characters

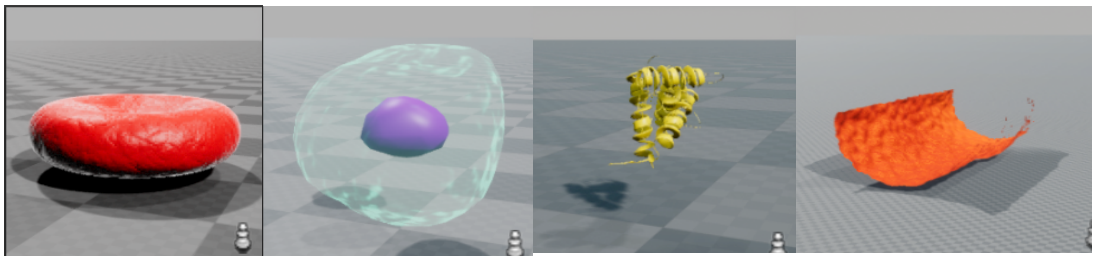
On the player side, there is the bacterium and the virus swarm. For both of them, models are available and basic movement and shooting is implemented. The virus swarm consists of a couple of individual viruses which try to stay together as best as they can given the fluid forces.



Player models



Enemy models



Neutral models and environment

Figure 9 Characters and environment models of the interim demo.

On the enemy front, both neutrophils and macrophages are implemented with models and basic logic. Neutrophils are simple enemies which only drift in the fluid without targeted movement. When they reach a player, they die and the player takes 1 HP damage. Macrophages are larger enemies which actively try to reach the players. They inflict more damage on the player and take more hits to kill. Both enemy classes are spawned in bunches every few seconds.

The game also includes neutral characters which can be seen in the background and serve to make the environment feel more crowded and believable. Red blood cells and different kinds of proteins fall into this category. The models and their behavior already exist but they have not been added yet.

An overview of all current models is shown in figure 9.

3.2.3. Fluid simulation

The movement of both players and NPCs is influenced by a global two-dimensional fluid simulation.

This basis for the simulation of turbulence in a fluid is laid out in the paper *Curl-Noise for Procedural Fluid Flow*. It describes a method to generate a divergence-free vector field based on perlin noise. The idea is simple: by calculating the curl of a scalar field $\Psi(x, y)$, we get a divergence free vector field which resembles isotropic turbulence. For Ψ we use the more efficient simplex noise $S(x, y)$ instead of perlin noise. To make the turbulence more realistic

and detailed, we augment this noise function by adding up multiple octaves.

$$\Psi(x,y) = S^n(x,y) = \sum_{l=1}^{n} \frac{1}{2^l} S(2^l \cdot x, 2^l \cdot y)$$

In practice we find that $n = 3$ gives a good balance between large- and small-scale flow. The fluid velocity field can then be expressed as

$$\vec{V}(x,y) = \nabla \times S^n(x,y) + \vec{L}$$

where $\vec{L}$ is a linear component used to make enemies move to the left. Numerically, the curl is implemented as a finite difference

$$\nabla \times \Psi = \begin{pmatrix} \partial/\partial x \\ \partial/\partial y \\ \partial/\partial z \end{pmatrix} \cdot \Psi = \lim_{\Delta \to 0} \frac{1}{\Delta} \begin{pmatrix} \Psi(x+\Delta, y) - \Psi(x,y) \\ \Psi(x, y+\Delta) - \Psi(x,y) \\ 0 \end{pmatrix}$$

where in two dimensions $\frac{\partial \Psi}{\partial z}$ vanishes. We use $\Delta = 10^{-4}$ for the numerical evaluation.

Boundary conditions ensure that no in- or outflow occurs at specific points. In a typical level of our game, the wall of the blood vessel requires this condition. Boundary conditions are implemented as isocontours in Ψ. This enforces that $\nabla \times \Psi$ is perpendicular to the boundary in two dimensions. We smoothly ramp Ψ down to zero at the boundaries using a fifth order polynomial. Objects outside the fluid domain are pushed inside using a constant acceleration. Optionally, the boundaries can act as hard wall, stopping objects dead in their tracks when they collide. We implemented boundary conditions for boxes and cyllinders which should account for all the obstacles encountered in the game.

For the torque on an object, we calculate the curl of the vector field $\Omega_z = (\nabla \times \nabla \times \Psi)_z$ which yields the angular velocity in the x-y plane. The velocity in the other axes cannot be computed from Ψ directly as the simulation is only two-dimensional. Instead, we use an uncorrelated noise sample $S^n(x', y')$ which can be thought of as the fluid velocity some way above the object. The x and y component of the torque then result from the difference in velocities $\frac{\partial (\nabla \times \Psi)_x}{\partial z}$ and $\frac{\partial (\nabla \times \Psi)_y}{\partial z}$.

The fluid interacts with an object by applying forces and torques as described above. For laminar flow, the force on an object inside a fluid is

$$F = m \cdot a = \gamma \cdot (\vec{v} - \vec{V}(\vec{x})) \cdot A_{\vec{v} - \vec{V}(\vec{x})}$$

where $\vec{x}$, m and $\vec{v}$ denote the location, mass and velocity of the object, $\vec{V}(\vec{x})$ is the fluid velocity at $\vec{x}$ and $A_{\vec{v} - \vec{V}(\vec{x})}$ is the projected area of the object onto the flow direction. The

factor γ measures the friction force between fluid and object surface. For arbitrary shapes, $A_{\vec{v}-\vec{V}(\vec{x})}$ is hard to determine. We therefore approximate it using the scaling law of volume and area $A \approx \alpha\, m^{2/3}$ where α is a constant. Consequently the force becomes

$$F \approx \gamma' \cdot (\vec{v} - \vec{V}(\vec{x})) \cdot m^{2/3}$$

where $\gamma' = \gamma\ \alpha$ is the effective friction or global interaction strength of the fluid which can be found experimentally. For the torque we use a similar relation, finding that

$$\vec{\tau} \approx 4\gamma' \cdot (\vec{\omega} - \vec{\Omega}(\vec{x})) \cdot m^{2/3}$$

where $\vec{\omega}$ is the angular velocity of the object. For fast-moving objects, laminar flow is not applicable anymore. Above a certain speed threshold, we therefore increase the force on an object with the squared velocity.

3.3. Challenges

This section focuses on the problems and handicaps we had during the first implementation phase. It will be divided into 2 parts: minor and major challenges.

3.3.1. Minor Challenges

Minor challenges represent issues that had to be addressed but were on the one hand expected to occur and on the other not too hard to solve. All of the minor issues can be associated to the following two topics: Unreal Engine and project management skills. At the very beginning of the first programming phase, every team member had to familiarize himself with the game engine. As none of us has ever worked with Unreal Engine before, this took quite a few hours and some tutorials on different implementations every now and then. For example, we had to learn how to connect source files in C++ with the Blueprint files of Unreal, since we rely on using both for our features. Furthermore, like in almost every other group project, we had to figure out a way to modularize our task breakdown in order to ensure a fair delegation of tasks among all team members. We solved this issue by introducing a project management tool in form of a so called Trello board[1].

3.3.2. Major Challenges

Major challenges include all issues which arose unexpectedly during the implementation phase. These issues weren't trivial and it took us more time to solve them compared to the minor challenges. We had major difficulties in all modules of our application, the players, enemies, UI & design and general tasks.

[1] https://en.wikipedia.org/wiki/Trello

Players

As we planned to implement a virus swarm and a bacterium as our two main characters in the game, we had to solve the collision in two different ways. For the bacterium, it was straight forward: if the bacterium collides with an enemy, remove a health point. Handling the virus collision required a few more steps. We want to control the whole swarm with our input device while every single virus of that swarm should additionally be influenced by our fluid forces. Furthermore, every virus can collide and also be destroyed by enemies. Implementing this functionality was probably the biggest issue so far.

Enemies

By having implemented an up and running fluid simulation for every object in the scene, a lot of issues concerning random movement of neutrophils were already taken care of. The only challenge in this module so far was to implement and combine a smooth follow mechanic for the macrophage with the fluid simulation and the linear velocity influencing the cell's movement, as the cells are not as free in their movement as the players.

UI & Design

As our artists already had models for the players and enemies in their mind, only one challenge arose in this module: Creating a background mesh that seamlessly moves from right to left.

General

First of all, the biggest challenge talking about general issues was to implement the fluid simulation according to mathematical concepts describing the fluid movement in the real world. One of the problems here was to include boundary conditions to the simulation so that the objects weren't flushed out of the scene. Similar to the minor issues, we also had more difficult problems to cope with concerning Unreal. The usage of a version control system, GitHub[2] in our case, combined with Unreal internal UAsset files was not easy so far. As these files are binary instead of source files, we had several problems regarding merges of different versions and branches. Furthermore, the Git integration by Unreal is only in experimental mode, so that we couldn't come up with a best practice so far.

[2] https://github.com/

4. Alpha Report

4.1. Progress

At this point, most of the implementation work is done. We are happy with the progress we made since the interim report. We finished all the desired targets except the cockpit view of the driver, which we discarded. In retrospective, the project plan and the targets we defined were a good starting point and helped a lot to keep track of our progress. Now we are looking forward to let friends and others play the game and get their feedback. This will be especially helpful in balancing the game and maybe tweak some of the game elements. We hope that they will enjoy the game as much as we do.

4.1.1. What is missing?

Overall, we can say that only few essential features are missing in our game. We implemented most of the key features. On our road to our desirable target we did some design changes and therefore did not stick to our original development plan.

Player We finished our bacterium and virus logic except for one feature: an ultimate ability, where our two players mutate for a short time period. We focused more on the look and feel of the basic controls of our virus and bacterium and decided to discard the ultimate ability for now

Enemy This is the domain where we made the biggest design change. We originally planned to have new enemies introduced in the desirable target, i.e. Natural killer cell that target the cells that are infected by the virus and so called dendritic cells who are notified by macrophages in order to inform the immune system "to send help". As these wouldn't influence our current gameplay, we decided to focus on some high target tasks instead.

UI With respect to the UI, we almost implemented all features on the road to the high target. In our desirable target we lack one feature due to design decisions. Instead of making a second level with a different setting, we focused on optimizing our first level in order to ensure fun and challenge in our game.

Design Here, as a part of our design revision, we skipped the implementation of a second level. We are instead focusing on fine-tuning the animations for different sections in the game.

General We even completed our high target in the general section and therefore do not lack any features.

4.1.2. What is done?

In the last three weeks we finished a lot of work. In the following, we will describe what we achieved in each area. The structure is similar to the progress subsection in the last chapter.

Targets that we already described there, will only be shortly or not mentioned.

Player We finally finished our two players with two different skills and playstyles. The bacterium deals more damage, moves slower and shoots slower projectiles. Our virus is squishy, fast and shoots different projectiles than the bacterium and has less health points.

Enemy So far, we completed five different kinds of enemies. First, the neutrophils. They float around randomly, only influenced by our fluid dynamics and hurt the players on contact. Second, there are macrophages. These cells follow and damage the closest player. Third, we implemented red blood cells, which spawn proteins when they get shot. Fourth, the proteins who can be eaten by the bacterium in order to regain health. Lastly, there are cells that can be infected by the virus for health regeneration.

UI Quite at the beginning we decided to only use implicit indicators for our game. In the current state, we have indicators for the players' health status as well as their abilities. The bacterium's color interpolates from green to red dependent on its health. The virus swarm has one virus for each health point. In case the virus swarm gets damaged, it loses one virus.

Design Our goal for the design topic was to implement high resolution models of our players and enemies. We realized not only the models but also self made animations for all participants in our game.

General We implemented some sound effects for different parts of the game. The heart beat that influences the enemy movement in our fluid simulation as well as health regeneration skill of the bacterium are audible. Also, our game can be played either with a keyboard and a controller or with both players on the keyboard.

4.1.3. What's next?

The next step in our development is now the playtesting. We hope to get a lot of feedback for the balance of our game and also what feedback mechanism are missing. Furthermore, we plan to add some of our high targets and extras. We do not know how much of this we will able to include. But even if we aren't able to include all, we think our game will be a lot of fun to play.

4.2. Game description

At the start of the game, the main menu is displayed.

4.2.1. Immune System & Level Progression

As the players in our game are a virus and a bacterium trying to infect and damage a human body, we developed an enemy system that is inspired by the human immune system. This system adjusts it's parameters as the players progress through a level, modifying values including what enemies to spawn, where to spawn them, how many to spawn etc. In it's current form, the progression through a level can be divided into 4 phases.

In the first phase, the players are still unnoticed by the body's immune system. In this stage only infectable cells and red blood cells spawn, allowing the virus and bacterium to replicate and amass health for the fight that's to come.

The players however don't get by unnoticed for long and the second phase starts. The body sends it's first line of defenses, the Macrophages. Once the body realizes that the infection can not be beaten by the Macrophages alone, the Macrophages start releasing Neutrophil Messengers with increasing frequency. These messengers can be shot by the player in order to delay or prevent a response of the immune system. If they leave the screen however they will alarm the body, demanding reinforcements and triggering the third phase.

In the third phase reinforcements, namely the Neutrophils, join the fray. Macrophages keep producing Neutrophil Messengers. Every time one of them leaves the screen, the chance to spawn Neutrophils increases up to a maximum value. If the combined efforts of the Macrophages and Neutrophils are still not enough to suppress the infection, the Macrophages will start to release Dendritic Messengers. In Reality these messengers bind to dendritic cells, which then make their way to the lymph nodes in order to activate T-Helper Cells and B-Cells. Because this happens away from the site of the infection and does not have a direct influence on our gameplay, we facilitated this process. The dendritic messengers can again be shot by the player, delaying or preventing the body's response. However if they leave the screen, they will notify the body and trigger the fourth phase.

In the fourth phase the body mobilizes his heavy hitters. For every dendritic messenger that leaves the screen, the chance to spawn B-Cells and T-Helper Cells increases up to a maximum values. The point when they reach their maximum spawn chance is the most difficult phase of the game. At this point the players have to endure a certain amount of time. If they manage to survive, they win and are returned to the main menu.

The system features a large set, which have to tuned in order to provide a fun and challenging experience. As this is our first version of the system it is likely subject to some changes before our game is finished. We think that especially the end of the game is currently not satisfying and should feature some kind of climax or different win conditions.

4.2.2. Improved Models

Up until now the quality of our model greatly differed. In order to give our game a consistent look we strove to improve the currently less-appalling models. The main focus was on the rigid and simplistic looking bacterium. We scrapped the current model and recreated it based on a new reference picture. The new model features an drastic increase in the level of detail as shown in figure 10. Additionally, the main body and the flagella are now animated. In particular, the movement of the flagella are physics driven. One flagellum is internally represented by a line of angular constrains that together form some sort of chain.

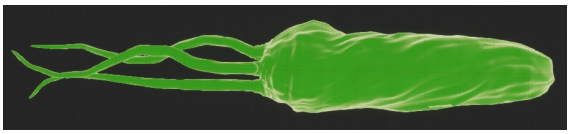


Figure 10 Improved Bacterium model

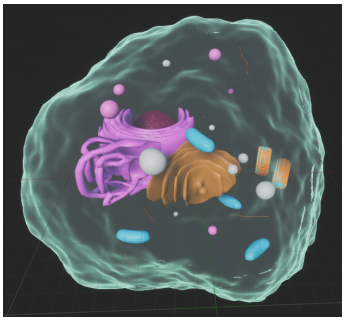


Figure 11 Human cell model

Another major addition was the creation of a human cell model (see fig. 11). In-game it is used to enable the virus to regain some health via infecting the cell and using it for re-population. Since the human cell is generally of great importance we ensured to represent it with a high level of detail. We made the membrane see-through in order to visualize a lot of important components of a typical cell; such as: the nucleus, mitrochondrions, smooth and rough endoplasmic reticulum, and centrioles.

A few minor improvements were the increase of quality of the neutrophils (see fig. 12) and the creation of dedicated models for both player projectiles. The bacterium projectile (see fig. 13) symbolizes a small piece of the bacterium itself, while the virus projectile (see fig. 14) is based on antigen.

4.2.3. Custom and procedural shaders

To fit our style of scientific visualization, we created custom shader code for all of our models. These shaders are based on the physically based shading model and implement additional features such as

Rim shader: Most of our materials include a rim component, fading the outlines of objects towards white. This helps to distance the objects from the background.

Procedural animated texture: In the microscopic world of cells and bacteria, everything is always in motion. To realistically reflect that fact, we included animations in our most prominent materials. We employ a procedural three-dimensional noise generation algorithm to avoid repeating patterns and prevent the animation from looking like a simple texture panning. The semi-transparent cell wall of our neutrophils for example exhibits this behavior as well as rim-shading.

Procedural normal map: For some of the animated materials, e.g. for the red blood cells, we

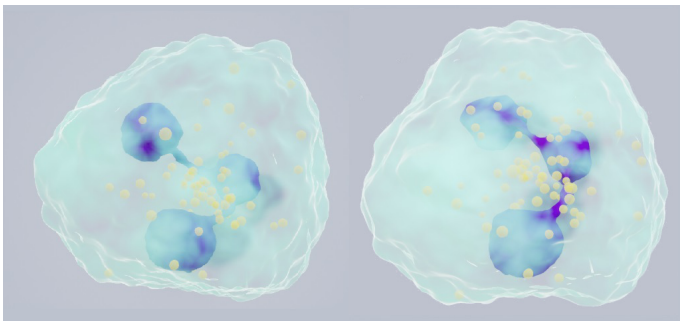


Figure 12 Improved Neutrophil models

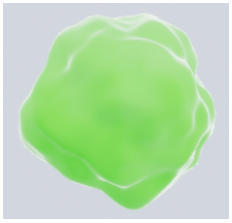


Figure 13 Bacterium projectile model

Figure 14 Virus projectile model

add a normal map to sell the impression of microscopic structure on the surface. As the texture is generated procedurally, the normal map cannot be baked beforehand. Thus, we use an algorithm to calculate it on the fly.

Outline shader: The highlighting of objects in the main menu and in the game, when you can interact with them, is achieved with a post processing shader. This shader makes the outlines of objects flash up periodically. It is implemented based on a custom depth buffer used only for this purpose. Unlike many comparable algorithms, our effect is consistent with the depth of field effect which also plays an important role in graphics of our game. For objects that are out of focus, the outlines get blurred with the object.