# Gameplay

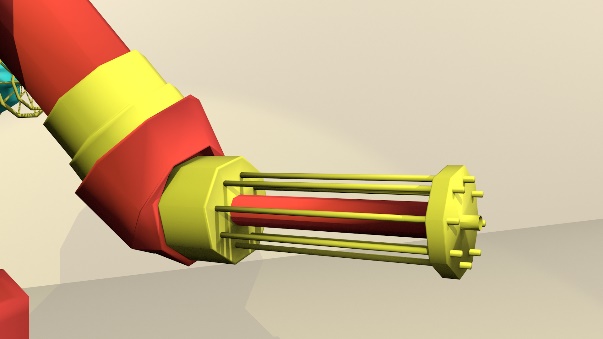
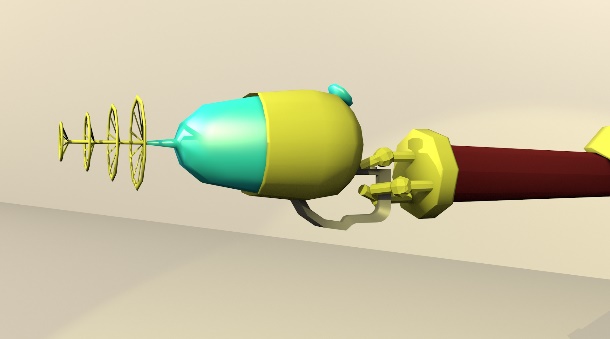
## Summary

“Micro Mayhem” puts the player in control of a technologically advanced form of medication, wittingly named “RoboPill”. RoboPill is a new form of medication that was able to be developed due to recent medical advancements. The player’s job, as RoboPill, is to enter the patients system, and destroy all harmful bacteria whilst saving as many healthy cells as possible. The player will progress through the game in a level-based fashion, earning evolution points at the end of each level depending on how well he was able to help the patient.

## RoboPill

RoboPill is the result of thousands of hours of medical research and micro-bacterial evolution. With the intended result being an extremely potent and multi-purpose medical capsule, RoboPill is able to enter a subject’s system and combat 90% of the world’s most fatal diseases.

RoboPill’s body is divided into two parts; the upper body, and the lower body. The lower body is made up of a rotational thruster, allowing for a full range of motion throughout a subject’s system. The upper part is a more combat ready segment, which can rotate around its centre in a 360 degree motion.

An improvement upon previous capsules, RoboPill’s left arm has been replaced by a mini-gun based weapon, which is always readily available. His right arm, however, features 3 opposable fingers which are able to grip secondary weapons. Shown below is a comparison between RoboPill’s Mini Gun arm and his wielding arm.

RoboPill's wielding arm, shown with The Cleanser equipped

RoboPill’s Built-In MiniGun

## Enemies

Enemies within the game will be mainly based on different variations of real-life bacteria that can be found. Each level, a new enemy will be unlocked, which the user will need to logically understand and defeat. Varying AI logic will be applied to each of the enemies, forcing the player to fight strategically whilst at the same time causing as much carnage as possible.

A wave-based spawning system will be utilised in the game, allowing many variations of enemies, adding replay value and unique gameplay which should further the complexity and immersion of the play-through. Although the wave-based spawning system will increase complexity and the need for multi-tasking, having a basic understanding of the game mechanics and enemy AI should allow the player to make it through each level.

## Level Design

Each level will take place either inside or on top of an organ inside the subject’s body. Some may use the same organ as an environment, however, the following locations of interest should be unique to each level;

1. Spawn Points
   1. Enemy spawn points
   2. Friendly spawn points
2. Environment
   1. Interactive props
   2. Obstacles
3. Pick-ups
   1. Ability pick-up locations
   2. Secondary weapon pick-up locations

Depending on the environment, the player may be constrained to the bounds of the level. If the level takes place inside an organ, the organ walls should stop the player from straying too far from the action, however, as the player progresses through the game, environments may begin to take place on the outer surface of an organ, increasing complexity for the player as the issue of “falling off the edge” may become a factor.

## Evolution & Infection Ratio

Throughout the game, the player will accumulate evolution points which can be used for progression. These points can be used to increase the effectiveness of the player, by increasing damage of certain weapons, unlocking new weapons and increasing the player’s defensive values.

Each level will provide a set value of evolution points to ensure the player is able to unlock certain features during their play-through. Furthermore, bonus evolution points will be provided depending on how well the player performed during the previous level.

The “Infection Meter” is a GUI element used to display the ratio of good and bad cells that are present in the subject’s body. When the meter is closer to the left, the subject is in a stable condition, as more healthy cells are present in the body, however, if it travels to the right, this show the subject is unstable, and is near death. If the meter stays here for too long, the subject will die and the player will fail the level.

The status of the Infection Meter will also be relayed to the player using in-game global illumination, allowing the player to stay focused instead of staring at GUI elements.

At regular intervals, the status of the Infection Meter will be recorded, and used as an average throughout the level to calculate the player’s performance, and subsequently, how many bonus evolution points they are awarded.

## Navigation

The player will control RoboPill from an almost orthographic angled camera, whilst retaining full 3D perspective view.

As can be seen in the reference image, RoboPill’s body is section into two parts; Upper & Lower. This allows for more in-depth control of the character in regards to movement and aiming.

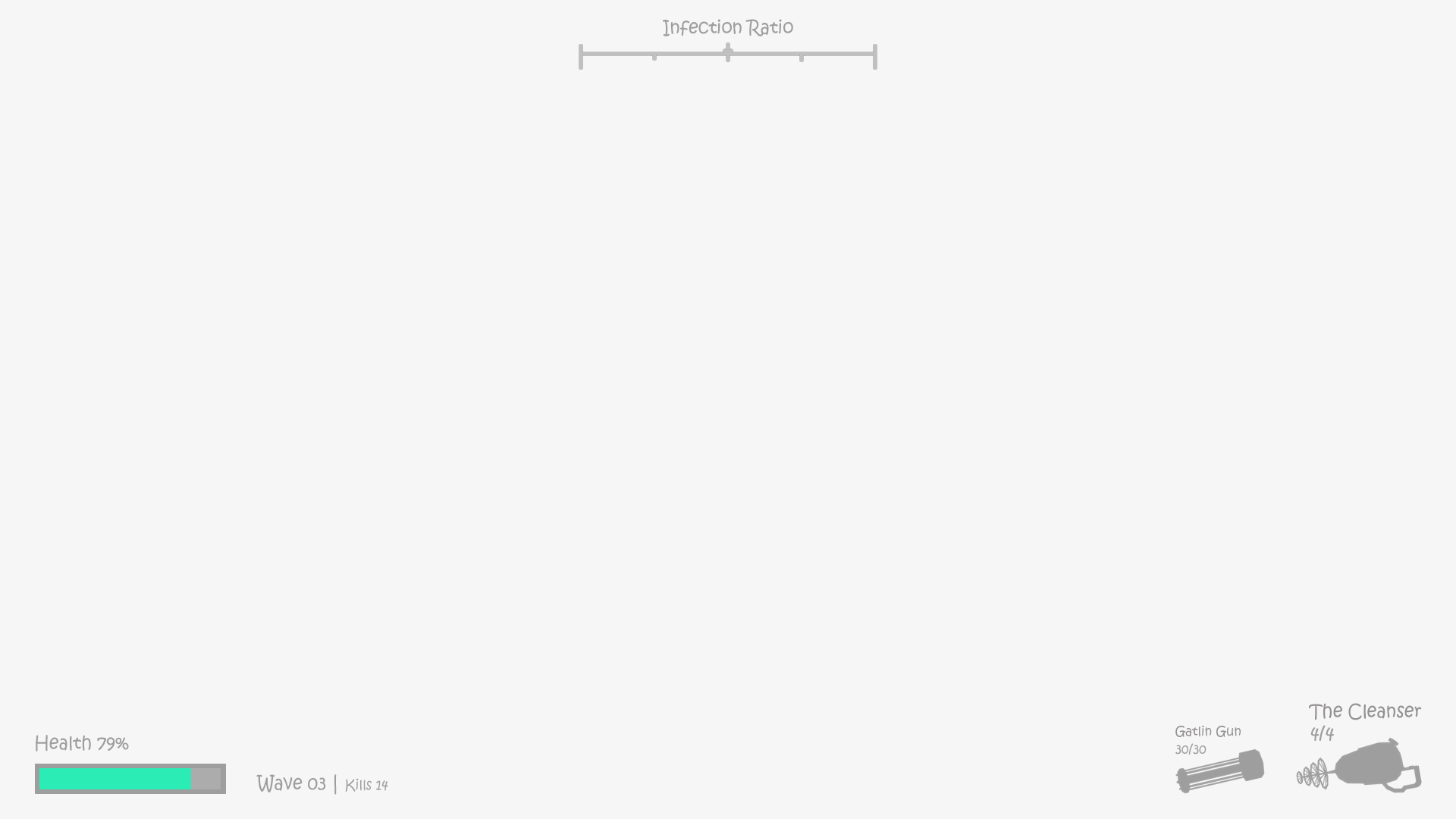


Using WASD or the arrow keys, the player will be able to manipulate RoboPill’s lower-body thruster to move him around the environment using a fixed axis; meaning they will thrust in the same direction regardless of the rotation of RoboPill’s upper-body.

The mouse pointer will be used to rotate the upper-half of RoboPill’s body, giving a full 360 degree range of motion to fire weapons. The upper-half will be rotated depending on the angle between the mouse pointer and the character’s world position.

## UI Design

Aiming for a sleek, minimalistic style, the UI Design will be bare-bones and will only show the player what they need to know. This avoids any clutter on the screen which can be off-putting and distracting, especially in such a high-paced game. Seen below, the user interface will consist of an indicator to show how well the player is doing, their current health and which weapon is equipped. As this is just a sketch, the in-game interface will likely be more aesthetically pleasing.



# Requirements

## Functional

Functional requirements are a collection of constraints that the game must have implemented, or be capable of, the fully function as specified in the design process. Listed below, these are split into technical requirements and in-game requirements.

### Environments & Technical

1. The game will be developed using the C# programming language.
2. Unity 2017.1 will be used as a development environment.
3. Blender v2.79 will be the modelling environment used to develop 3D assets.
4. Final build must run on any modern Windows PC.

### Gameplay

1. The player’s Mini-Gun (Primary Weapon) will fire when the Right Mouse Button is pressed.
2. The player’s secondary weapon/ability will be fired/used when the Left Mouse Button is pressed.
3. Secondary weapons/abilities must be consumed when ammunition runs out.
4. RoboPill’s upper-body must rotate based off of the mouse position.
5. RoboPill must move on a fixed axis, ignoring the rotation of his upper-body.
6. All damageable items in the scene must be able to take damage and die.
7. Infection Meter must be updated each time a cell is destroyed.
8. Enemies must follow their assigned AI type.
9. Player must die when health runs out.
10. Explosions must cause damage in a defined radius.

## Non-Functional

Non-Functional Requirements are not necessarily required for the game to completely function, however, they may add a drop of immersion and aesthetic that may not have been present before-hand. Such things include high quality sound effects or fine-tuned global lighting. The requirements for this game have been listed below.

### Audio & Visuals

1. The game should include a diverse soundtrack of at least 3 musical pieces.
2. Sound effects should be use to notify the player when events have triggered (i.e. Enemies spawning, enemies dying, ammo running out).
3. Subtle global lighting should reflect the current health of the subject’s body, regarding how many healthy/unhealthy cells are currently alive.
4. Particle effects should be used frequently to display the carnage being caused.
5. Cartoon-like effects and textures should be used to fit the theme of the game.

### Interface

1. User Interface should be kept to a minimum to avoid clutter.
2. Displays should be easy to navigate, and avoid showing unnecessary information.
3. A general theme should be followed to reduce player confusion.
4. Interfaces should scale with the player’s screen size to ensure consistent gameplay on different hardware.