**SCP REBREACHED**

Contents

[Phase 1: the Player 2](#_Toc56516760)

[Requirements 2](#_Toc56516761)

[Components and their links to each other 3](#_Toc56516762)

[Structure 4](#_Toc56516763)

[Current functions 4](#_Toc56516764)

[Phase 2: SCP-049 – The Plague Doctor 5](#_Toc56516765)

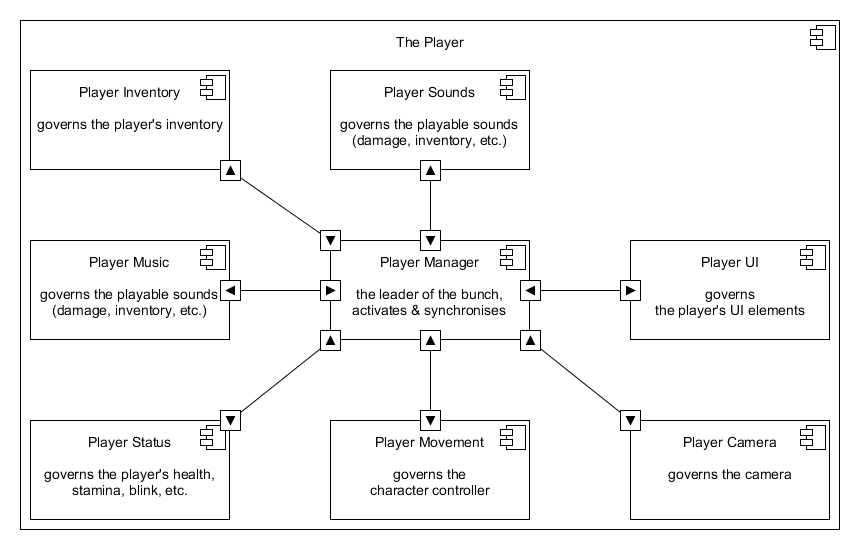
[Requirements 5](#_Toc56516766)

# Phase 1: the Player

## Requirements

|  |  |
| --- | --- |
| # | Requirement |
| 1 | The Player must be able to move around in a 3D environment, using the keyboard.  The System must be able to toggle this ability at will. |
| 2 | The Player must be able to look around a 3D environment, using the mouse.  The System must be able to toggle this ability at will. |
| 3 | The Player must be able to sprint, to increase their speed for a short time.  Sprinting will be limited by a stamina bar. Stamina will only be drained if moving and sprinting, and will regenerate over time. Draining the stamina bar fully will make the Player exhausted, being unable to sprint until 20% stamina has been recovered. |
| 4 | The Player should be able to jump, to overcome obstacles.  Jumping will be limited by a stamina bar. The same rules of exhaustion for sprinting apply here. |
| 5 | The Player should be able to sneak, to avoid enemy SCP’s. |
| 6 | The Player must be able to take and recover from damage.  Taking damage will be limited by a health bar. If the Player’s health is less than 30%, the Player will become crippled. If the Player is slashed during any instance of damage, the Player will bleed. If the Player’s health is depleted, the Player will die. |
| 7 | The Player must be able to blink.  Blinking will be forced by a blink bar. Will make SCP-173/other scenarios a challenge.  The Player will also be able to manually blink, for strategic purposes. |
| 8 | The Player should be able to bleed.  Bleeding will reduce the Player’s health over time. |
| 9 | The Player should be able to be crippled.  Crippling will reduce the Player’s overall speed and disable the ability to sprint. |
| 10 | The Player should be able to recover from bleeding and/or crippled.  The first-aid kit will be the tool to recover from mortal wounds like these. |
| 11 | The Player must be able to spectate other players or SCP’s after death. |
| 12 | The Player must have an inventory.  Essential tools or SCP items will be kept here, and stored on the Player. |
| 13 | The Player must emit footstep sounds.  These must be audible to anyone nearby. |
| 14 | The Player model should have visibly blinking eyes. |
| 15 | The Player must be able to manipulate the environment, by taking usable objects or by interacting with usable scene objects, such as keypads. |
| 16 | Locally played music should be smoothly faded into each other (crossfade). |

## Components and their links to each other



The Player is largely managed by the Player Manager. The Manager will process inputs and execute those via [Command] functions, and if necessary, [ClientRpc] functions.  
Using the free networking library, Mirror, mandates the use of [Command] and [ClientRpc] functions to allow for smooth, multiplayer gameplay.

A Player can use [Command] functions to interact with the server as a client, and [ClientRpc] functions are executed among all connected clients, allowing, for example, synchronized footsteps.

The Manager takes data from all the other components and synchronizes them where necessary.   
For example: the health bar. The Manager takes the health value from Status and places it in UI,  
so that the UI can update the bar’s value on screen.

Player Movement allows the Player to move around in a 3D space, being able to move in eight directions: north, west, south, east, northwest, northeast, southwest, southeast.

Player Camera allows the Player to freely look around the 3D space using a computer mouse.  
The X-axis of the camera is limited to 90 and -90 degrees, so the Player cannot flip the camera upside down. Furthermore, the camera has a Z-axis attached to it as well, allowing it to roll to the left and right. This will allow for more ‘immersive’ low health situations, making the camera roll a bit to one side when the Player’s health is low.

Player Status handles everything related to the Player’s numbers: health, stamina, blinking, etc.  
According to those values, it also sets flags using Booleans, allowing the Manager to handle more efficiently without continuously reading and parsing float values.

Player Music and Sounds obviously handle the audio in or out of the Player.

Player Inventory handles the entire inventory of the Player. What is the Player holding? How can it be used? This will be stored in the Manager as a simple int value.

## Structure

The Player exists out of multiple GameObjects, be it an empty object or a filled one.

The root GameObject contains a Character Controller, allowing for simple movement script interaction. It also holds all scripts together in one object for simplicity, and contains everything Mirror needs to function properly.

The first child of the GameObject is the camera, which obviously gives you the ability to look around in the 3D space. The camera contains an Audio Listener, allowing the Player to hear sounds relative to the camera. Furthermore, the camera contains a Post Process Layer and internal   
Post Process Volume, allowing for additional screen effects to be shown when deemed necessary.

Only one camera exists per scene (even in multiplayer) so the camera is parented to the local player on runtime.

The second child of the GameObject is the Player’s model, which is at the moment a collection of primitive shapes, namely a Capsule and a Sphere with two more Spheres in it. This crudely resembles a human being with a head and two eyes.

The third child of the GameObject is the Player’s collection of Audio Sources. Music, footsteps, hurt sounds, etc. is contained in this object.

The fourth and currently last GameObject is the Player’s personal UI Canvas, allowing for individual UI elements per player.

## Current functions

The Player can currently:

* Move;
* Look around;
* Sprint;
* Take damage;
* Recover from damage;
* Blink;
* Play music via triggers/flags;
* Play footstep sounds;
* Use keypads.

The Player still needs to be able to:

* Jump;
* Sneak;
* Access their inventory;
* Use items from their inventory/in their hand.

# Phase 2: SCP-049 – The Plague Doctor

## Requirements

|  |  |
| --- | --- |
| # | Requirement |
| 1 | 049 must be able to move around in a 3D environment, a little bit slower or equally as fast as a player. 049 should be a ‘trap’ character, not a chasing character like 173.  The System must be able to toggle this ability at will. |
| 2 | 049 must be able to chase a player. |
| 3 | 049 must be able to select a target, preferably the closest one. |
| 4 | 049 must be able to kill a player with a single touch. |
| 5 | 049 should be able to convert a player’s corpse to an instance of 049-1, a zombie. |
| 6 | 049 must be able to interact with doors and open them. |
| 7 | 049 must be able to ‘hear’ nearby players without directly spotting them.  If a player is sneaking, 049 should be unable to ‘hear’ them. |
| 8 | 049 should change the player’s currently playing music to a searching and/or chasing theme. |
| 9 | 049 must be able to play voice lines in certain situations, such as: a first encounter with a player, searching for a player, spotting a player, chasing a player and killing a player. |
| 10 | 049 must also emit footstep sounds while moving, these must be distinct from actual player footsteps. |
| 11 | 049 should raise its right hand while nearby a player, demonstrating that it is going for the killing blow. |
| 12 | If 049 currently has no target, it will patrol around the level, ‘searching’ but pretty much just bumping into players. |