# Design Document - 平行線 (heikousen)

## Background Story

Good morning Commander. After our last try to shut down the AI, it has gone rogue. You have been given our newest combat vessel, equipped with advanced planning and timeline travelling capabilities. Your mission, should you choose to accept it, is to disable this cursed program, once and for all.

## Gameplay

The gameplay is split into two stages: a planning phase and an execution phase.
First you plan your movement through a level without time going by, where you don't know how certain elements will behave later on.
Then see your move played out with all the dynamic elements active, with some limited interaction abilities (like activating a short-term shield, or erasing all bullets currently in existence).
However, if you fail, you will have to try again with all your previous actions known to the AI, having to anticipate a different AI response.

You are controlling the spaceship in first person view, and are able to move and rotate in all three dimensions.
The dynamic elements are things like doors opening and closing, lasers switching on and rotating, and turrets trying to hit you with bullets.

| | |
|---|---|
| W, S | Thrust forward/backward |
| A, D | Thrust left/right |
| R, F | Thrust up/down |
| Q, E | Yaw left/right |
| Mouse up/down | Pitch up/down |
| Mouse left/right | Roll left/right |
| Enter | Start planning / start the run |
| Space | Use your special |

# Technical Details

- Vulkan will be used as rendering backend.
- For collision detection and physical effects, the bullet engine will be used.
- Our script engine will be the V8 JavaScript engine, as used by the chromium project and Node.js.
- The configuration file will be a simple .ini file.
- cmake will ensure cross-platform builds on Linux (gcc) and Windows (msvc).
- glslang will compile the glsl shaders to the spir-v format used by Vulkan.
- GLFW is used for window management.
- GLM is used for vector and matrix calculation.

## Types of objects in the game

All objects are illuminated, only static level elements cast shadows (baked into the lightmap):

- Your spaceship - A textured mesh controlled by the player
- Level interior - A textured mesh representing the boundaries of our level
- Doors - An animated mesh with collision detection
- Lasers - A simple mesh with collision detection
- Turrets - A complex mesh controlled by simple AI, shoots bullets
- Bullets - A simple mesh with collision detection and response

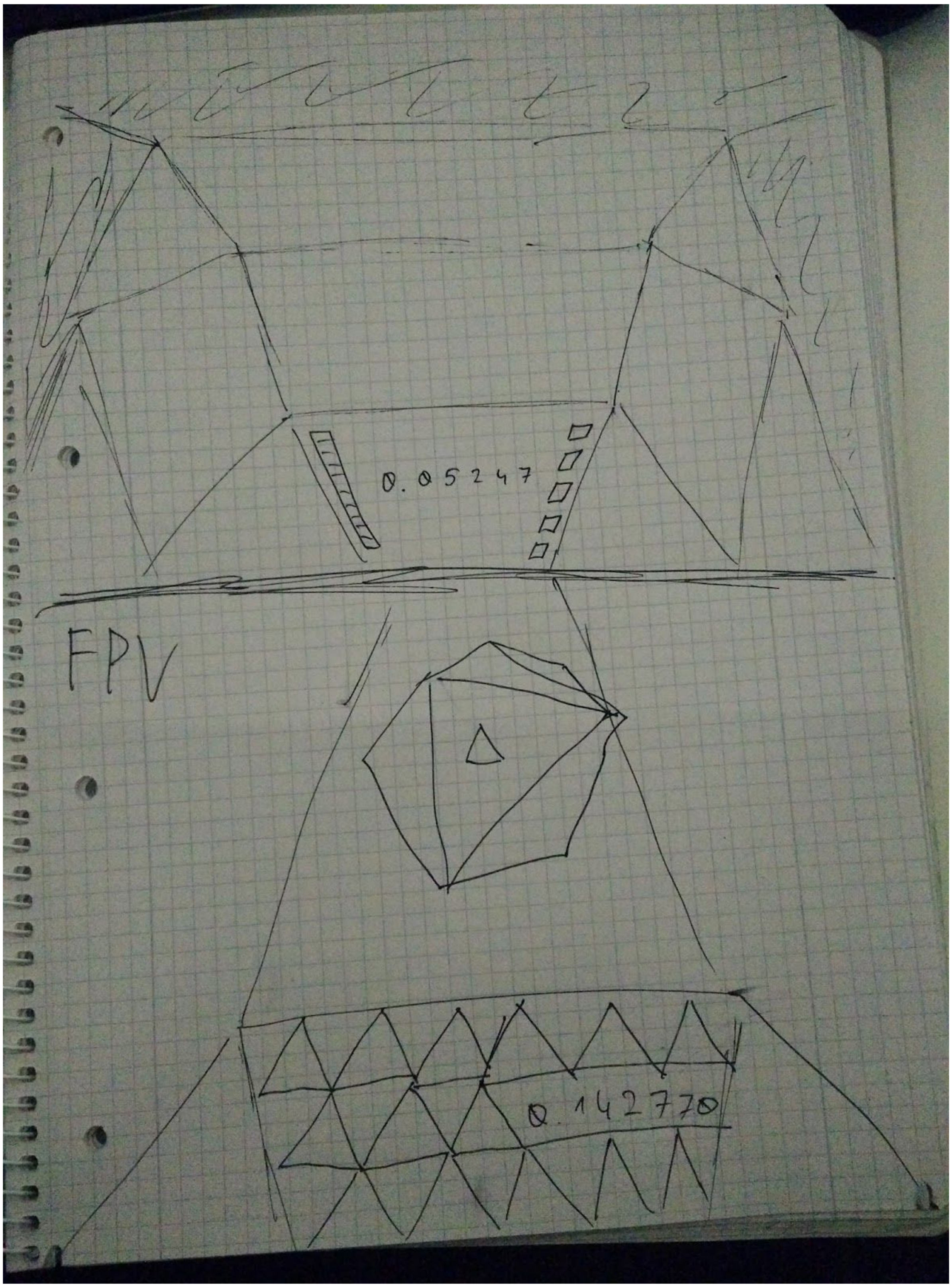
## Illumination

There is no global light source, the levels are interior levels and illuminated by pointlights. Shadows of static elements are baked into the lightmaps. The lightmaps are stored separately from the actual texture.

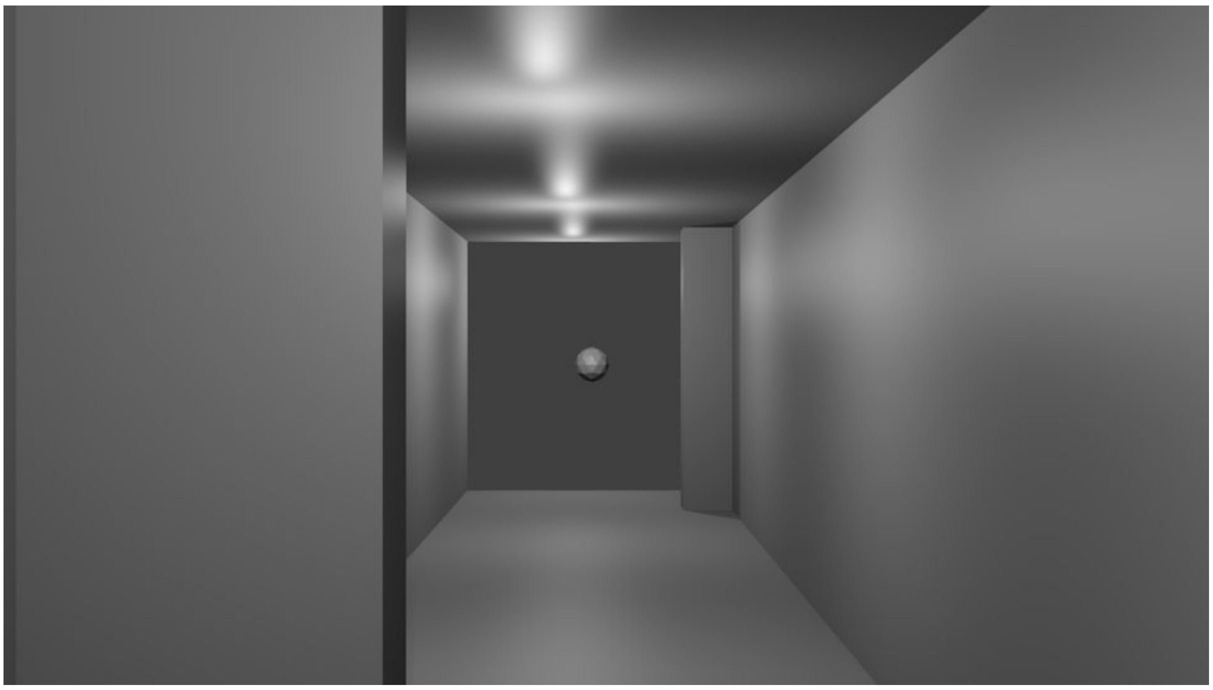
## Effects

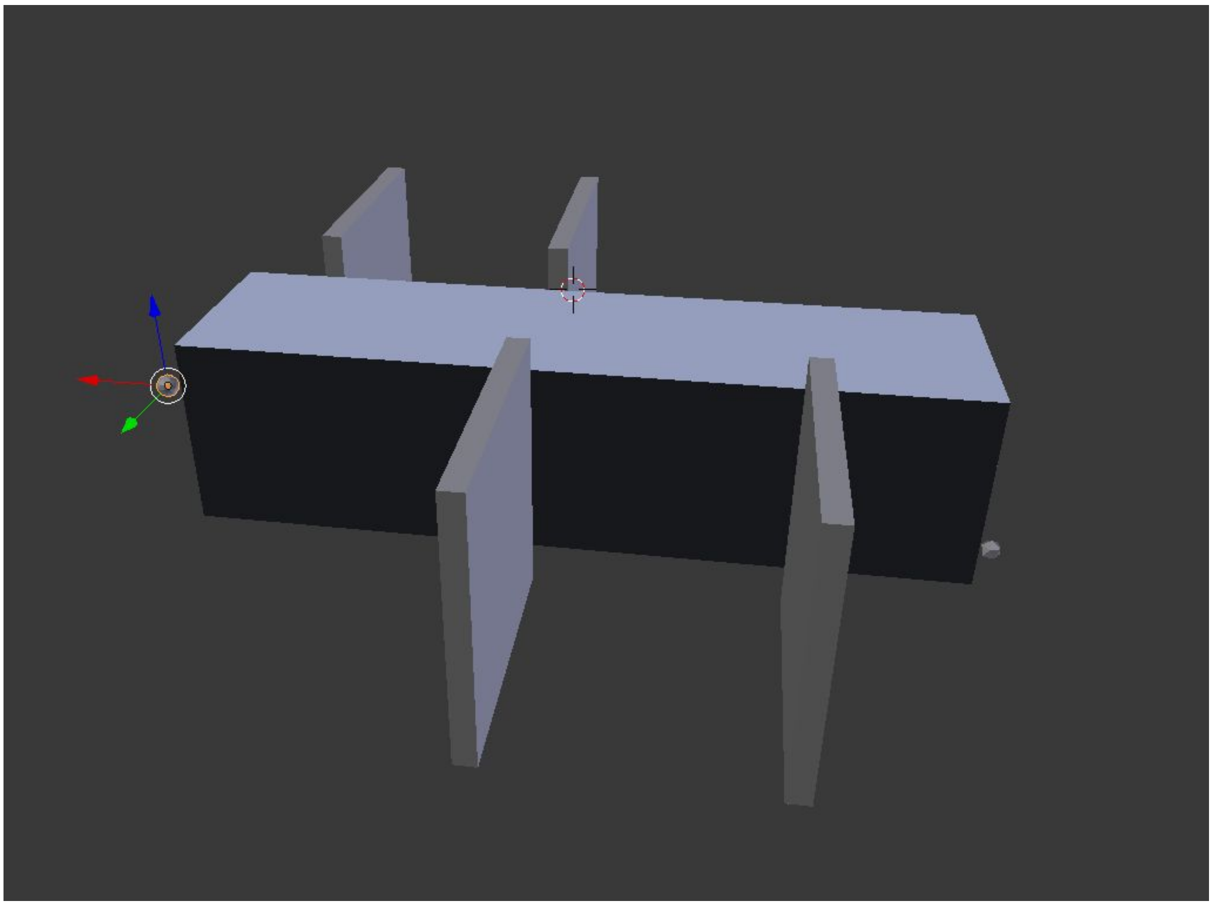
| Points | Effect |
|---|---|
| 0.5 | Separate Textures for Lightmapping |
| 1 | Scripting Language |
| 1 | HDR rendering and tone mapping |
| 0.5 | Simple Normal Mapping |
| 1 | BSP Trees |

First person view



## Very simple level mockup





You can find an animated video mockup on our github in the “docs” folder.