ROCKET MECHANIC

GAME DESIGN DOCUMENT

HIGH CONCEPT

Congratulations! You are the first human on Mars! But you have to leave … NOW!
The starting sequence has been initiated, but the escape pod is damaged and needs some last-minute repairs. Fix things up quickly or your last day on Mars might end badly!

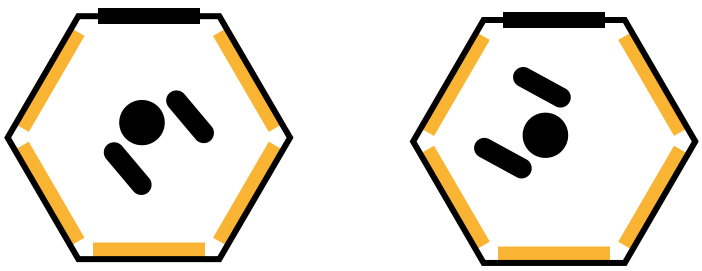
Genre:	VR Puzzle Game, Singleplayer
Platform:	Oculus Quest
Target Audience:	Puzzle Enthusiasts, Indie Game Lovers, VR Owners with a sense for exploration, Space Fanatics
Features:	- A quick-paced and tense gameplay - Unique puzzle mechanics - Intuitive Controls and Interactions

GAMEPLAY

The players find themselves inside an escape pod that is about to launch into space but looking around, they find that nothing is working. Broken panels, blinking lights and a merciless countdown urge them to quickly get an overview of the situation and start fixing things!
The goal of the game is to make the gameplay as intuitive as possible and allow players to jump into the experience without being taught how to play the game. All challenges the player faces should be self-explanatory and give them an instant idea on how to solve them.

LEVEL DESIGN

The space capsule is represented as a small hexagonal room surrounding the player. Except for the door, each side of the capsule is covered in modules that will be randomly generated for each mission. The player can turn around in their seat to view all sides of the capsule or alternatively use the controllers to turn their view in a fixed angle. The symmetry of the environment makes it hard for players to orient themselves. While this is part of the challenge, additional features like colours, symbols or environmental details can help the player to find visual anchors.



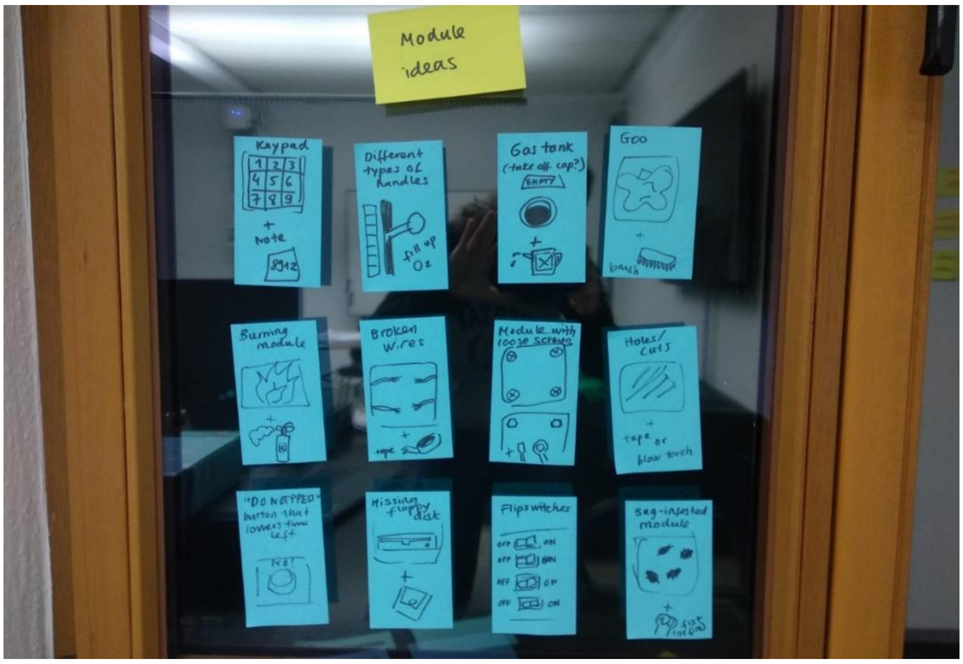
OBJECTIVE

The goal of the game is to repair all modules in the given time so that the space capsule can launch safely. The time left will be prominently visualized in the capsule, and additional audio cues will be given.
Modules that are not yet fixed will be indicated with a red light, turning into a green light when successfully repaired.
After the timer has run out, the player will either have a successful launch by repairing all modules in time or fail the mission.

MODULES

The modules are made up of different sized panels and will be randomly generated for each mission from a selection, adding to the replayability of the game. Modules can be either big or small and can be placed in a variety of ways on the square-shaped wall panels. This leads to a minimum number of 10 and a maximum number of 20 different modules.

Modules will be of different types that require different solution strategies. Some might only require button-pressing or turning switches while others need additional interactions with tools. The exact types of modules have yet to be designed, but several ideas have already been developed.



Sketches for first module ideas

TOOLS

Some modules will require specific tools for fixing. All required tools will be present within the capsule but may float around because of low gravity. There will be magnetic fix points where the tools can be attached for safekeeping.

Current ideas for tools involve:

- Wrench and screwdriver for fixing screws
- Blowtorch for fixing holes
- Canister to refill the gasoline
- Brush for scrubbing slimy surfaces
- Tape for connecting cables
- Fire extinguisher for stopping fires

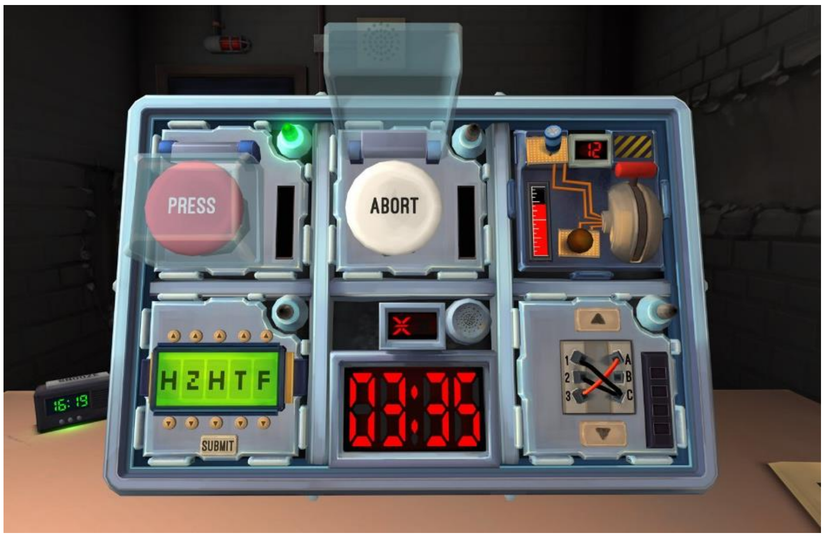
BALANCING

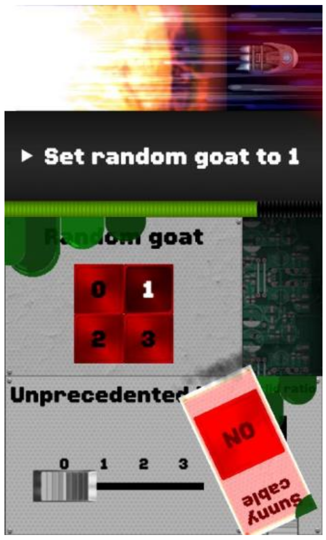
Balancing the modules to make the mission achievable in the given time will be key to a satisfying game experience. Potentially, each module type will be given an approximate *solving time,* and modules will be generated in accordance with the overall mission time.

If balancing should pose itself too difficult, players can still play in *Zen Mode*, simply trying to repair the modules and launch the capsule.

INSPIRATION

The gameplay is partly inspired by games like *Spaceteam* by Sleeping Beast Games and *Keep Talking and Nobody Explodes* by Steel Crate Games. The general concept of modules that have to be repaired or disarmed under time pressure is transferred into a single-player VR experience.

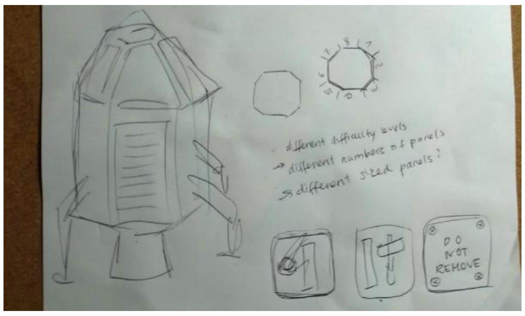




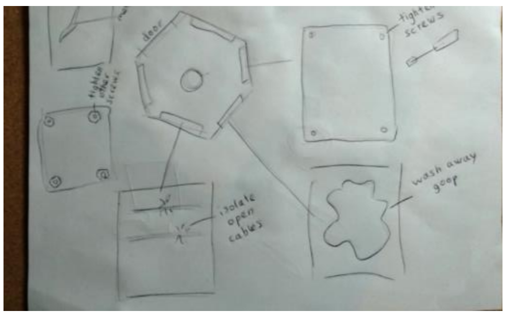
Screenshots from *Keep Talking and Nobody Explodes* and *Spaceteam*

VISUALS

FIRST SKETCHES

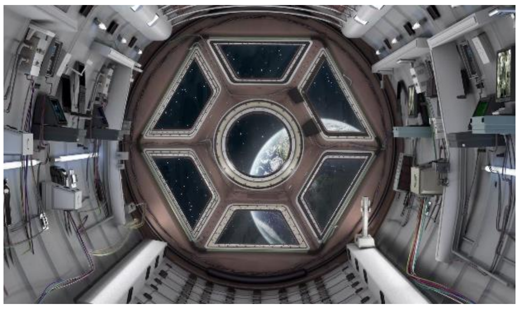


different difficulty levels
→ different numbers of panels
→ different sized panels?
DO NOT REMOVE



MOODBOARD

[1] Space Capsule Exterior

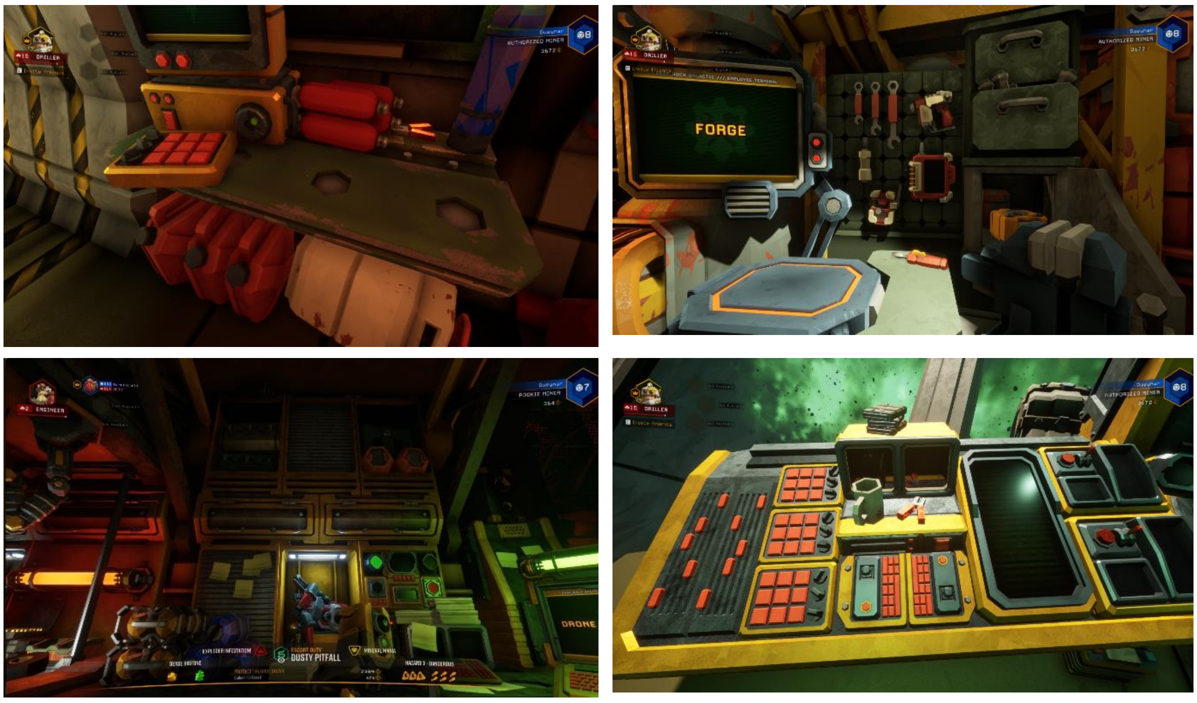

[2] Space Capsule Interior

[3] Stylized Control Panels

[4] Real-World Control Room

1 https://pc-tablet.com/spacexs-red-dragon-mars-mission/
2 https://www.turbosquid.com/3d-models/spaceship-interior-3d-model-1390244
3 https://thehungryjpeg.com/product/3700215-control-panels-spaceship
4 https://imgur.com/gallery/bwamoHB

ART STYLE



Screenshots from *Deep Rock Galactic* by Coffee Stain Studios

AUDIO

All interactions should be accompanied by according sounds to create a satisfying experience and feedback. The exact types of sounds will depend on the types of modules but will involve

- Module feedback sounds
- Tool feedback sounds
- Timer feedback at certain checkpoints

The game will have a fast beat background music that will increase in intensity when the time is running out.

The general mood transmitted by the sound effects and music should not be too serious but give the game a lighthearted note.

ROCKET MECHANIC

PROJECT DOCUMENTATION

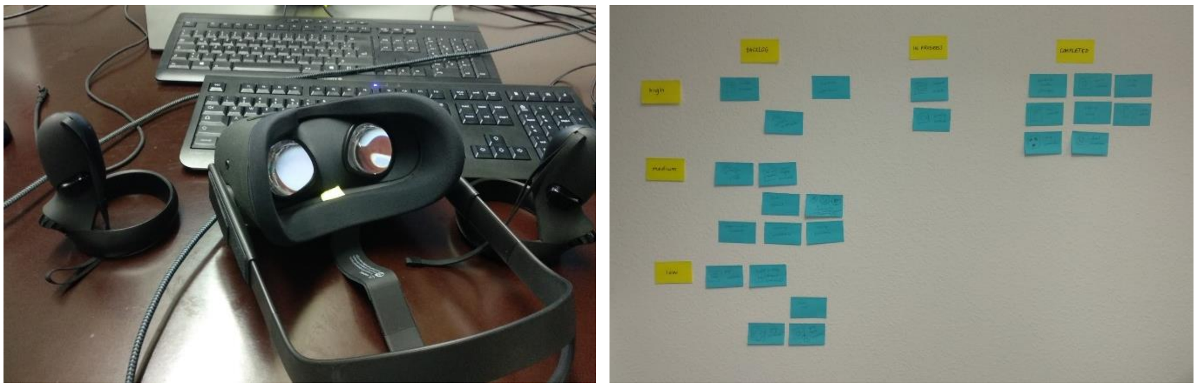
CONCEPT

The concept of the game was created to fit the topic of the thesis with the goal to create an intuitive VR experience that could be played without extensive tutorials or instructions. In *Rocket Mechanic*, players should figure out how all modules are supposed to be solved by themselves. The goal was to create a working prototype of the game that could be used as a proof of concept and for playtesting.

DEVELOPMENT PROCESS

I chose to work in the Unity game engine since I had the most experience with it and am familiar with the basics of C# scripting. However, I did not want to start at the very beginning, which is why I purchased the *Auto Hand* toolkit from the Unity Asset Store, which provides a set of basic VR interactions and objects. For testing, I had an Oculus Quest headset at my disposal, so I set up the game accordingly to be able to test and preview my created scenes.

To keep track of the tasks ahead and my personal progress, I created a Kanban Board consisting of a backlog, an *in progress* column and a section for completed tasks. This helped me to visualize my progress and also with task prioritization. In the end, all tasks on the board were either completed or cut from the scope, resulting in the final prototype.


Oculus Quest for development (left) and personal Kanban board (right)

C# Scripting

After everything had been set up, I started working on the prototype. I had already created a list of potential ideas for modules, so I first started working on the ones that were easiest to implement to test the basic functionality. I wanted the project to be build up from the beginning to adaptively adjust to different numbers of modules.

To keep track of the game's progress, the game manager creates a list of each module in the scene and compares it with the number of solved modules. If both values are the same, the winning condition is reached. I also implemented a timer script that could be set to any value and called the losing condition whenever the timer ran out without all modules being solved.

After setting up this basic functionality of winning and losing conditions, I continued working on the modules – prioritizing either ideas that were relatively easy to implement or those that offered the most value to the playing experience. In a few specific cases, I asked for help from Manuel Ott and Jonas Zimmer, who helped me with problem-solving. The particular sections where they helped me improve my code are accordingly commented.

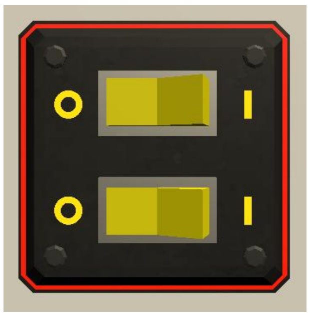
LIST OF MODULES

In the following, I will shortly list all modules that I was able to implement and their functionality. All in all, I created ten modules plus variations.

Button Module
This one is simple: You only need to press the button. This was one of the first modules I implemented to test the overall functionality of the modules and win conditions. It uses a standard VR button from the *Auto Hand* asset pack.

Lever Modules
Both levers were included in the *Auto Hand* asset pack and offered a good starting point for creating the modules. I checked either their local position or their hinge value to find out whether they were in the wanted position.
In the end, the mass of the levers was increased since they moved too easily, even when only being slightly touched. These modules are solved by simply pulling down the levers.

Valve Module
The basic model was included in the *Auto Hand* asset pack as well and uses a Hinge Joint component that can be manipulated to adjust maximum and minimum positions. I did not fully understand the extensive and complicated joint settings, but I managed to make the module activate once it has been turned to a certain degree to the right side. The yellow arrow was included to prevent any confusion about into which direction to turn the valve.



Flip Switch Module

The flip switches were created by me. After trying to alter their rotation via code, I eventually decided to use an animation state machine instead. Both sides of the flip switch have a small trigger zone that starts the animation. However, they flip on very easily. The sprites were included to make it obvious which direction to push them into. The module is solved when both switches are set to their active status.

Battery Module

The battery module consists of an empty battery holder, indicated by a bolt symbol. A battery from the level must be picked up and snaps into it once released close enough to the trigger zone. Snapping works by setting a new parent object for the battery and adjusting its position and rotation accordingly. Once the battery is snapped, its rigidbody is turned kinematic, and it cannot be interacted with anymore. Placing the battery solves the module.

Screw Module

This module is not solvable on its own but requires the electric screwdriver that floats freely in the level. Finding the additional tool adds to the difficulty of the module. Each Screw has to be touched with the Screwdrivers tip to be screwed in. Once all screws have been driven in, the module is solved.

Keypad Module

In this module, the four keys are always spawned in a random position. The player has to quickly register changes in each module and press the numbers or letters in ascending order in order to activate the module. The number of each key is sent as a value on each button press and compared with the previously registered number.

Cable Module

On the cable module, open sparking cables have to be fixed. In order to insulate them, players need an additional tool: the tape. It can be found in the capsule. Once the player touches a cable with the tape, it will be covered up. If however, the player touches the open cables, a sound effect will be played, and the according controller gives off a short vibration feedback.
Once all cables are taped, the module is solved.

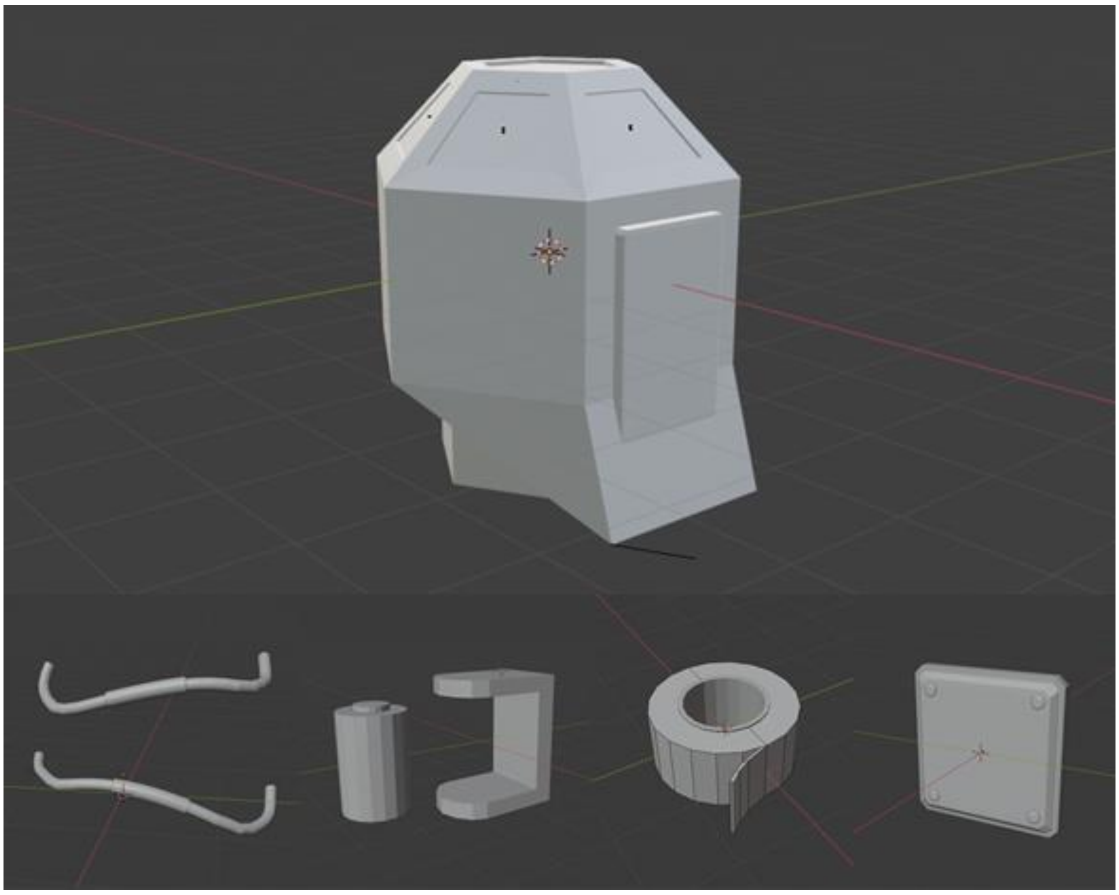
Bug Module

This was by far the module that took the longest time and, ironically, produced the most bugs. The random movement and spawn mechanism of each bug led to various problems regarding their rotation in the world space and their communication with each other, so only one bug would spawn at a time. The bugs appear and move to a number of random targets before returning to their spawn and disappearing again. The module is solved if all bugs are swatted (indicated by darker-textured holes in the module).

In general, almost all the modules have been programmed in a way that allows easy customization and variation by adding, removing or positioning elements differently without breaking the module and overall game functionality.

ART

While some assets could be found online or were included in the bought asset pack, the prototype had some requirements that were too specific to be met by them. This is why I created some of my own assets in Blender, most prominently the space capsule which forms the basis for the level and the back panels for the modules.


Assets created for the game in Blender

Additionally, when I started to apply Materials to the scene, I decided that I wanted all interactive sections of the game to have a distinct colour. This required custom colouring options for all objects and therefore created the additional need to create my own adjustable assets. To give the scene an overall consistent style, I added a custom texture pattern to the surfaces and generally went for a simple low poly optic. In the end, I also created a logo for the game, which is featured in the splash screen on start-up.

AUDIO

Giving the fitting sound feedback is essential for achieving satisfying interactions. Picking fitting audio clips was a task in itself. When possible, I tried to use arrays of sound or randomly pitched the value, so that there was some natural variation in the soundscape and tried to align it to player interactions as well as possible.

In the end, I added a random music generator to add mood and give a slightly different experience on each playthrough. I also added wind ambience to give the scene a more natural, less sterile feel.

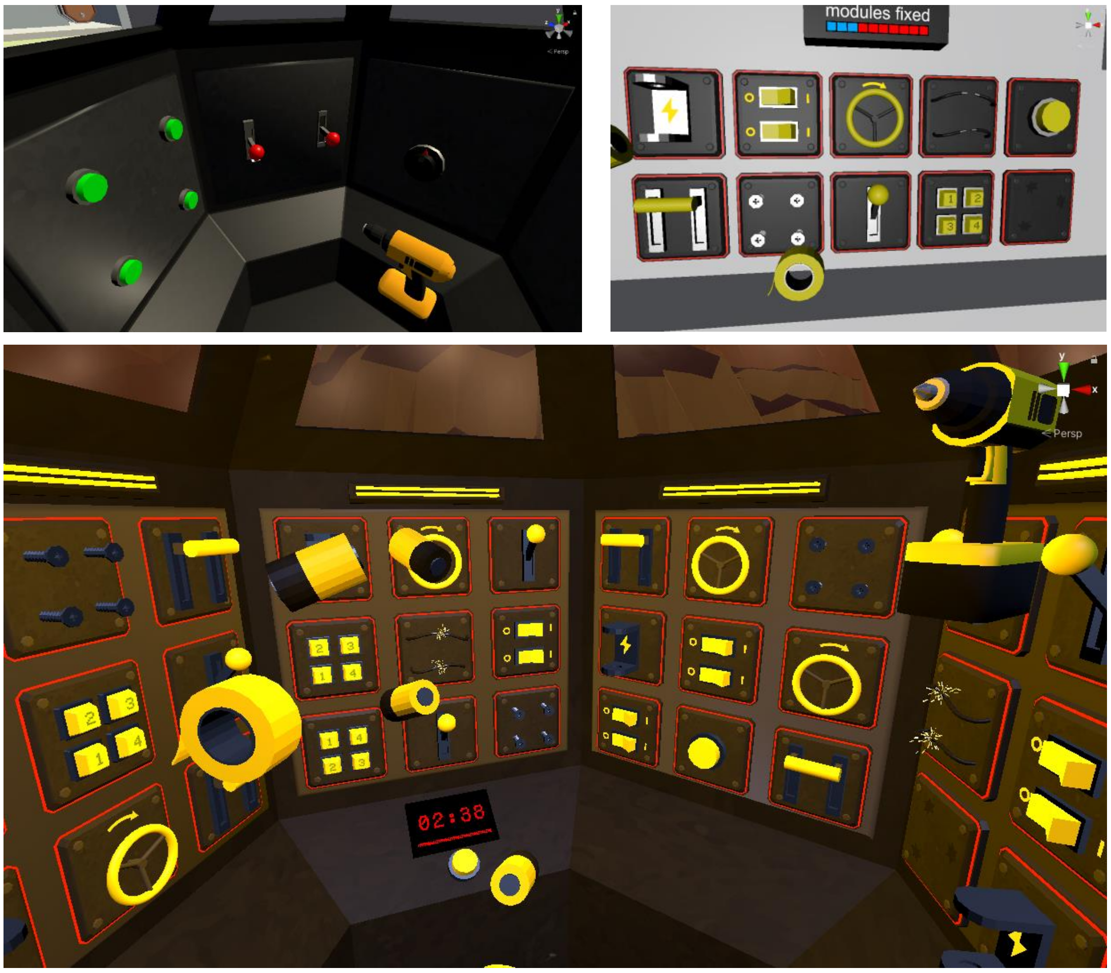
RESULTS

Bringing it all together in one scene was fun, but also caused some problems with the module prefabs that had to be fixed (the bug module specifically). I had planned from the beginning that objects inside the capsule would float without gravity since this offers a nice way to reach objects in a confined space in VR. It removed the need for the player to always place the tools carefully or risk them dropping to the floor. I managed to spawn random modules in predefined spots to create a different play experience each time and added some module variations. This makes the environment feel slightly less repetitive and at the same time, requires the player to be attentive and watch out for small changes.

In the end, after making some adjustments in the build settings, I was able to create a working build of my game for the Oculus Quest.

Because of the circumstances of this thesis and the time pressure, proper playtesting was not included in the schedule, but I look forward to showing the prototype to different people and observe if my design choices have reached the intended effects.

During playtesting it myself, I noticed that the remaining modules were not clearly identifiable as solved, no matter if I changed or removed the glowing colour ring around them. As a result, the difficulty of the game came through identifying which modules were remaining rather than solving them. Therefore, I decided to completely remove the modules’ content shortly after they were solved.



Different stages of development: First layout test (upper left), module prototyping (upper right) and final level (bottom)

REFLECTION

All in all, I am happy with my result and think that it offers a good starting point for investigating intuitiveness in VR.

However, this is only a first prototype, and there are a lot of opportunities for improvement. Right now, a fixed number of modules get generated completely randomly, which results in unpredictable difficultly levels. It would be interesting to create a dynamic difficulty curve by introducing new modules over time while increasing or decreasing the timer accordingly. Because of the progressing complexity of the prototype, some of the modules (like the button module) now seem too simple in comparison to others. When introducing a difficulty curve, a selection could be made, so that simple modules would not appear in higher levels or increase the complexity of each module.

It already becomes apparent that different modules require different problem-solving techniques from the player. The variety of modules featuring diverse types of challenges could be increased to enhance this effect.

Working with an external asset pack that uses physics and hinges created some unwanted effects that could not easily be fixed. Some of the hinges cause troubles with the hand collisions, resulting in frustrating interactions that I do not know how to prevent. While the overall game loop works, visual feedback (for example for the modules appearing and disappearing) and narrative could still be added to contextualize what is happening and prevent confusion. Additionally, the time pressure and as well as progress visualization should be more prominently featured.

Nevertheless, I achieved my goal of a working prototype and proof of concept. Working on this project was a lot of fun and required various skills from my studies. I hope that I can soon test the prototype with more people to see if my design achieves the intended effects, and if people are able to play and understand the game without instruction.

ROCKET MECHANIC

ASSET LIST

SCRIPTS & TOOLKITS

Name	Author	Source
Auto Hand - VR	Earnest Robot	Unity Asset Store
Tick Health Bar	Maxim Tiourin	Unity Asset Store
Timer Script	John French	gamedevbeginner.com

3D ASSETS

Name	Author	Source
VR Robot Hands	Earnest Robot	Unity Asset Store
Levers, Buttons, Valves	Earnest Robot	Unity Asset Store
Mars Environment	k0rveen	Unity Asset Store
Marspods and Dust	Overwound Entertainment	Unity Asset Store
Star Skybox	Pulsar Bytes	Unity Asset Store
Electric Screwdriver, Screws	Ilias Kap	Unity Asset Store
Lights	karboosx	Unity Asset Store
Bugs (altered)	shuldyakov	Turbosquid
Space Capsule	Rebecca Nöll	created in Blender
Module Base	Rebecca Nöll	created in Blender
Cables and Tape	Rebecca Nöll	created in Blender
Keypad Keys	Rebecca Nöll	created in Blender
Flipswitch	Rebecca Nöll	created in Blender
Battery Holder and Battery	Rebecca Nöll	created in Blender

2D ASSETS

Name	Author	Source
Palamecia Titling	Typodermic Fonts	dafont.com
Fake Receipt	Typodermic Fonts	dafont.com
Electronic Highway Sign	Ash Pikachu Font	dafont.com
Game Logo	Rebecca Nöll	created in Illustrator
Various other sprites	Rebecca Nöll	created in Illustrator

AUDIO ASSETS

Name	Author	Source
Brass Orchid (Music)	Bobby Richards	YouTube Music Library
Crazy (Music)	Patrick Patrikios	YouTube Music Library
Fast and Run (Music)	Nico Staf	YouTube Music Library
June (Music)	Bobby Richards	YouTube Music Library
SciFiAlert (Music)	Adam Bielecki	Unity Asset Store
Explosion	Iwiploppenisse	freesound.org
Rocket Thrusters 1	MATRIXXX_	freesound.org
Rocket Thrusters 2	MATRIXXX_	freesound.org
Wind Ambience 1	ERR0	freesound.org
Wind Ambience 2	jackyyang09	freesound.org
Countdown beeps	PITCHEDsenses	freesound.org
Countdown voice	tim.kahn	freesound.org
Module Solve	Chris Markert	Unity Asset Store
Bug Smash	mrickey13	freesound.org
Bug Movement	Hawkeye_Sprout	freesound.org
Keypad	JZProductions	freesound.org
Keypad Error	Jacco18	freesound.org
Electrocute	tonycarlisle	freesound.org
Electricity	deleted_user_7146007	freesound.org

Battery Click	jorickhoofd	freesound.org
Battery Powerup	Diamond00744	freesound.org
Screwdriver	GowlerMusic	freesound.org
Screwing	gfxwiz	freesound.org
Tape	djlarson3	freesound.org
	natenaterson	freesound.org
	AlaskaRobotics	freesound.org
Flip Switch	deleted_user_7146007	freesound.org
Button	PhilStrahl	freesound.org
Lever	Motion_S	freesound.org
Latch	deleted_user_7146007	freesound.org
Valve	FreqMan	freesound.org