Team Rastermotte: Project “Elevator Pitch”

Milestone 1: Formal Game Proposal

1 Game Description

General Description and Core Mechanics:

Project “Elevator Pitch” is supposed to be a 2D rogue-like (if the time allows it maybe even a rogue-lite) game in which the player will control a space elevator which is equipped with two mechanical multi-purpose arms that are its means of defense as well as accomplishing most tasks during the dangerous climb up into space.
The theme of “UP and DOWN” is at the very core of the whole gameplay loop as the player will start at ground level (or even in a sort of hangar underground) and try to reach several goals further up, returning back down with loot / new knowledge/ etc. and in a final run eventually reach space. During these very vertical missions/climbs the player will face waves of enemies trying to attack the elevator and will have to control the mechanical arms to grab and smash enemies into each other or smack them back down to earth. The arms are the main way to combat enemies, but also a tool used to collect resources from enemy wrecks and the environment and loading in mission objectives down in the hangar. With the arms as the main feature we try to make them feel the most polished. They are supposed to work as a physics-based combat system. Elements like the arms, especially the claws, enemies and environmental objects will each have their own mass and will thus influence the way the arms are able to grab, move and fling them. Punching something with the claws while holding a heavy object will have more impact than with empty claws. The arms are sword and shield at the same time. On the one hand (hahah) grabbing, flinging and punching enemies will serve as attacks while on the other side forming a flat surface will allow the claws to be used as shields for the purpose of pushing back enemies or reflecting projectiles.
Smaller and faster projectiles can be deflected back at enemies that way and larger, slower projectiles might need to be caught for example.
The arms will be controlled via mouse input. Only one arm at a time will be under manual control (later upgrades to the elevator might allow for the inactive arm to be in some sort of automatic defense mode). The claw of the currently controlled arm follows the movement of the mouse as closely as its physical limitations will allow and the movement of the arm will be handled by inverse kinematics. The speed and range of the claw will be limited by its current weight and the length of the arm, so extremely speedy and erratic mouse movements won’t have much of an effect as the claw will need to de- and accelerate accordingly first. Left-clicking will close the claw, right-clicking will make a flat claw to deflect

projectiles. Alternatively, we could make the closing and opening of the claw be controlled by the mouse wheel, which would leave the left mouse button free for another function e.g. forming a fist to punch with, playtesting will have to show what feels more satisfying.
Moving the mouse from one side over to the other side of the elevator will lead the currently active claw back to the elevator-cable and the other claw to become actively controlled.
Since the arm has a limited range, the player will have to use the ‘W’ and ‘S’ keys to move the elevator up and down a bit to allow for a small amount of additional vertical movement during combat. To make throwing enemies both easier and more satisfying, the arms each have a ‘fling mode’. For example by pressing the ‘space’ key, the mouse control scheme will switch from directly controlling the claw to directly controlling the movement of the upper arm. In this mode the rest of the arm is simulated by forward kinematics in a ragdoll-like manner, allowing the player to fling grabbed objects (like enemies) in a horizontal direction provided properly timed release of the grabbed object.
The other core mechanic will be the managing of the elevator during the non-combat climb parts. While out of combat the player will control a single character, the pilot, inside the elevator. The player is then free to move inside it for the purpose of refueling the engine, making repairs, controlling the climb rate, adjusting settings of the arms (or if development time allows it even explore the elevators immediate surroundings with e.g. a jetpack). This aspect is inspired mostly by the game “FAR: lone sails” although adapted to our genre and setting and it seems like a nice balance to the dexterity-based combat parts of the game.

General Gameplay Loop:

A single gameplay loop will look something like this: The player starts in the hangar and can browse through several missions (maybe predefined, maybe procedural) and select one or upgrade the elevator with earned funds. Depending on the mission type the player will then have to load cargo into the elevator (using the mechanical arms of course) or let passengers aboard.
Optionally, the hangar might be capable of horizontal underground movement to dock at different space-cables, so the player might need to select a specific one for a mission or choose freely for an exploratory climb. In that case, the final goal might be to upgrade the elevator enough to be able to make a climb at a specific final and particularly dangerous cable to reach an end-goal. Otherwise the end goal might be to just raise enough funds to retire or pay off a debt.
In any case, after starting a climb the player will start out inside the elevator managing it until an enemy-alarm sounds and mechanical arms will have to be manned for defense, thus starting combat mode.
Now, several waves of enemies attack the elevator and using the very versatile arms, the player defends themselves.
After all enemies are vanquished, the opportunity to scavenge the battlefield for resources comes up. The player can collect whatever is in reach, until the cargo is full or the battlefield is empty. Resources allow for repairs and fuel, but a heavier cargo load will also increase fuel consumption (or slow down maximum climb speed. thus giving enemies more opportunities to attack).
These climb, combat and collect phases then repeat until the mission goal is reached or the player is forced to abort and drop back down.
After falling back to the hangar the whole loop reiterates.

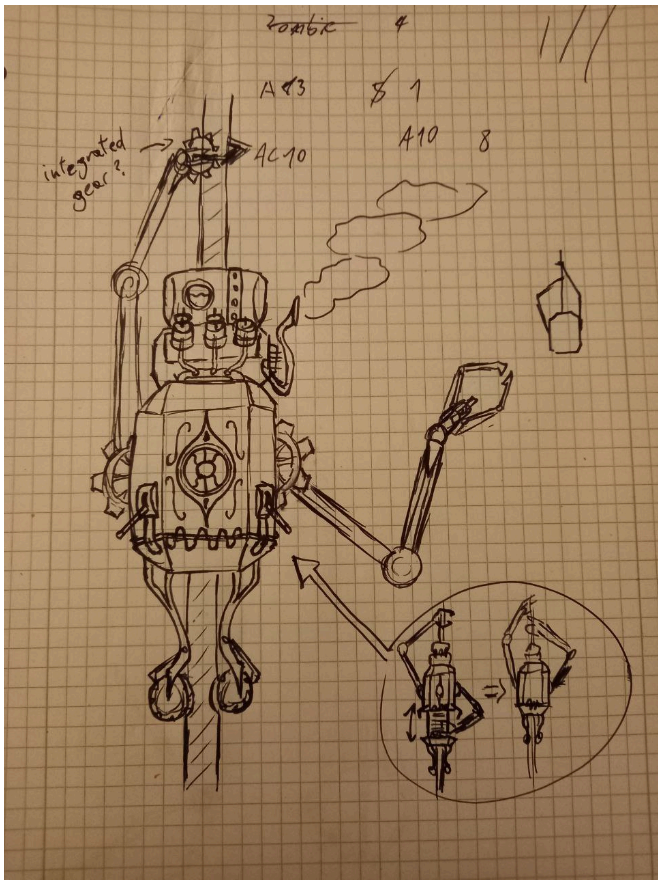
Story and Setting:

We aim for a steam- or dieselpunk aesthetic. Inspiration for the style and world are games like the “Deponia”-series and maybe in some aspects classic post-apocalyptic settings like the “Fallout”-series, although with the aforementioned steampunk twist.
The ground of the planet is mostly a wasteland of trash, scrap metal and derelict machines of a long past and more developed world, but anything but dead. While plant-life is rare, humanity survives between all the trash in smaller groups and villages, a far cry from the continent spanning states of the past. Following environmental collapse, the rich and mighty of the people started building a new society in orbit, creating thousands of (almost) self-sustained space stations and spanning equally many cables between the earth and orbit for use with space elevators to efficiently reach those stations. The poor were left on Earth. While the original upper class fled the furthest out into space, with time some of the more powerful lords that arose from the trash-folk of the surface started to claim certain space-cables and building stations around them inside still breathable parts of the atmosphere, forming a new kind of ‘middle’-class. The new rule of human society thus became that a person's status was literally determined by how ‘far up’ they are, although quite a big buffer still exists between the true space stations of the now-called ‘Orbitals’ and the ‘Scrappers’ who are still bound to breathable parts of the atmosphere.
We have several ideas for the story of the player character themselves all depending on what the actual long term goal will be.

1. The player could be aspiring to live among the Orbitals in space and in need to raise enough funds to afford the enormous entrance fee.
2. The player could be a secret revolutionist trying to abolish the class system, upgrading the elevator until it is capable of rising up to actual space, delivering a payload of a small revolutionist army to one of the stations.
3. The player could be an explorer with the goal to discover the mysteries of the upper atmosphere, dreaming of proving the legends of another society in space right (or disprove them)

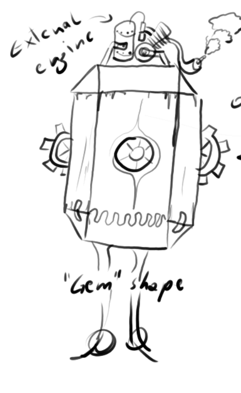
Concept Art and Illustrations

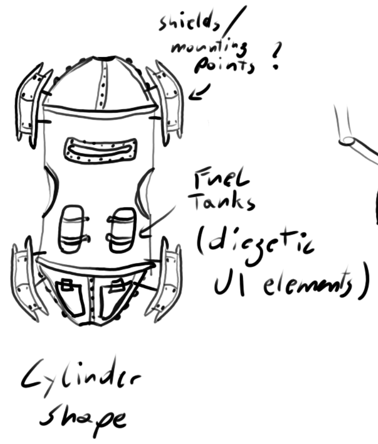
Ideas for the design of the Elevator:

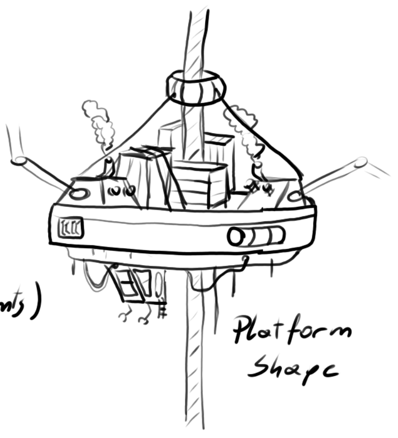


Elevator Designs: (to be realized in 2D Pixel Art!)

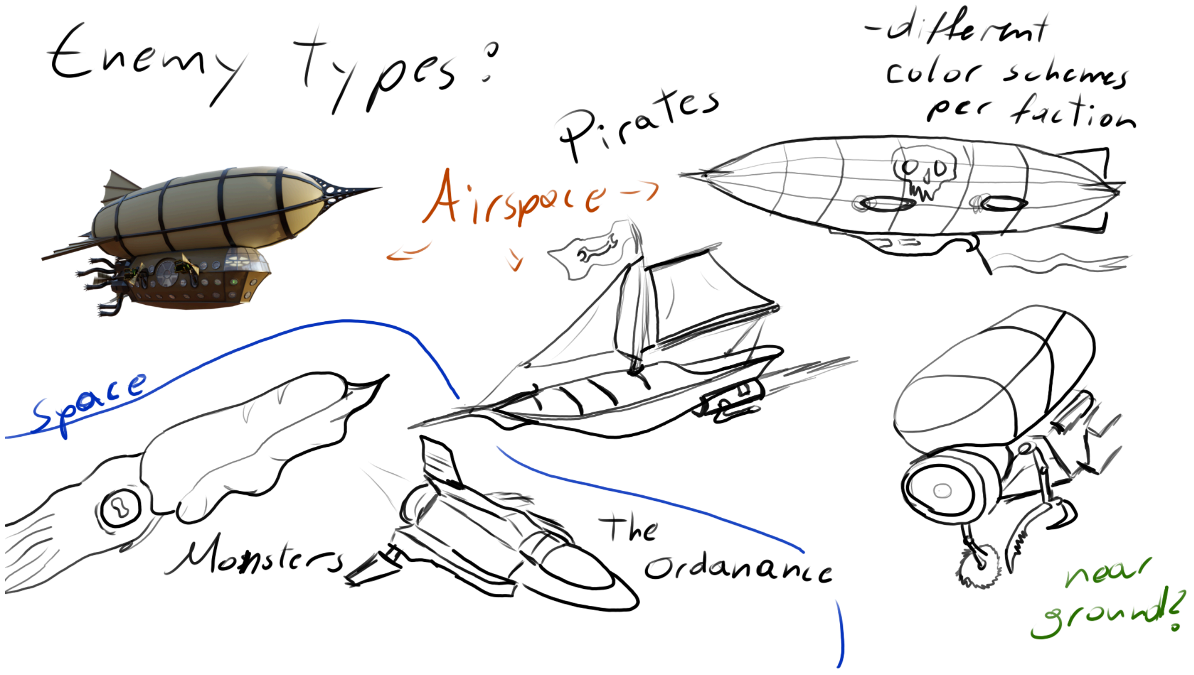
Body:

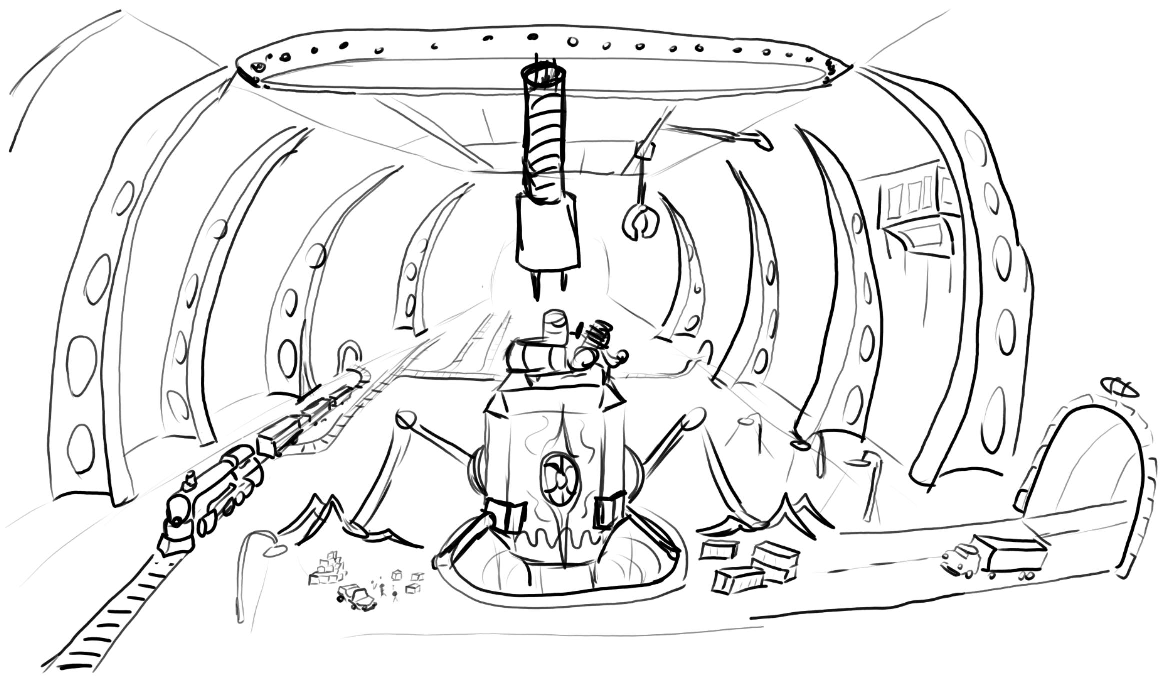


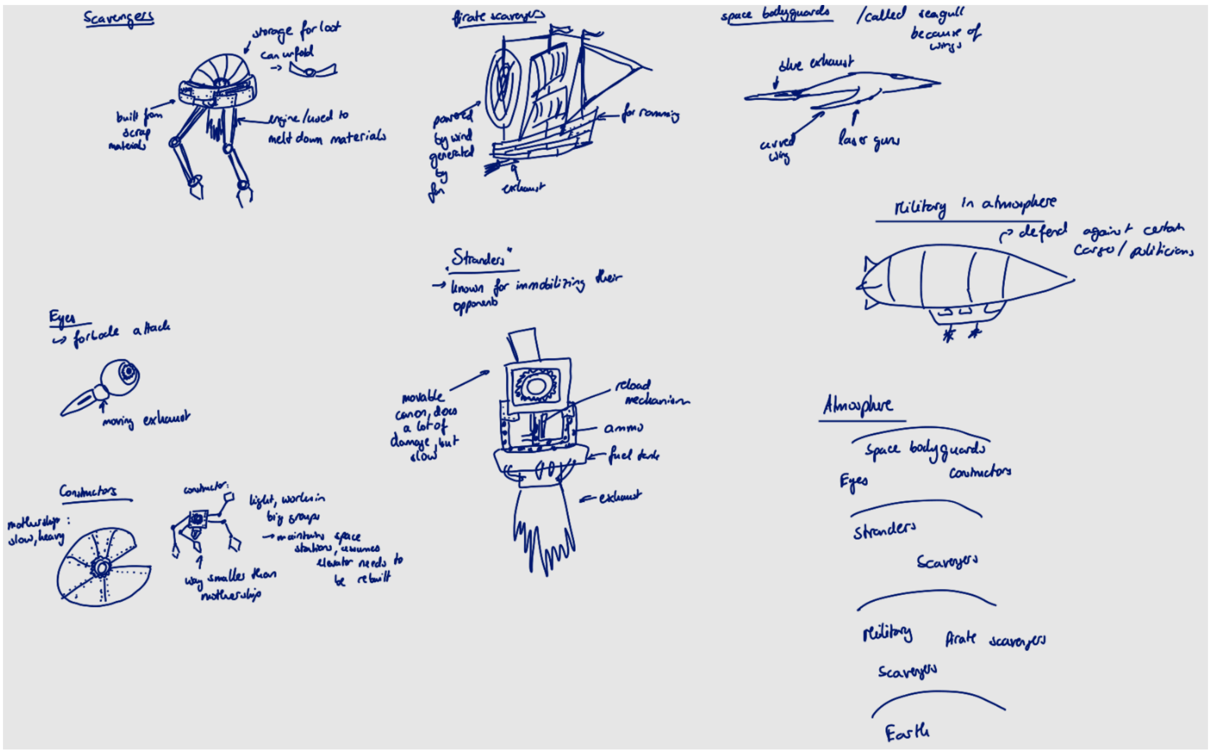

shields / mounting points ?
Fuel Tanks (diegetic UI elements)
Cylinder shape



Enemy types?


Pirates
-different color schemes per faction
Airspace ->
Space
Monsters
The Ordanance
near ground?




Scavengers
storage for loot
can unfold
built from scrap materials
engine / used to melt down materials
Pirate scavengers
powered by wind generated by fan
for ramming
exhaust
space bodyguards
/called seagull because of wings
blue exhaust
curved wing
laser guns
Military in atmosphere
defend against certain cargo / politicians
"Stranders"
→ known for immobilizing their opponents
Eyes
forlorn attack
moving exhaust
movable canon, does a lot of damage, but slow
reload mechanism
ammo
fuel tank
exhaust
Constructors
mothership: slow, heavy
constructor:
light, works in big groups
→ maintains space stations, ensures elevator needs to be rebuilt
way smaller than mothership
Atmosphere
Space bodyguards
Eyes
Constructors
Stranders
Scavengers
Military
Pirate scavengers
Scavengers
Earth

2 Technical Achievement

Physics-based Combat

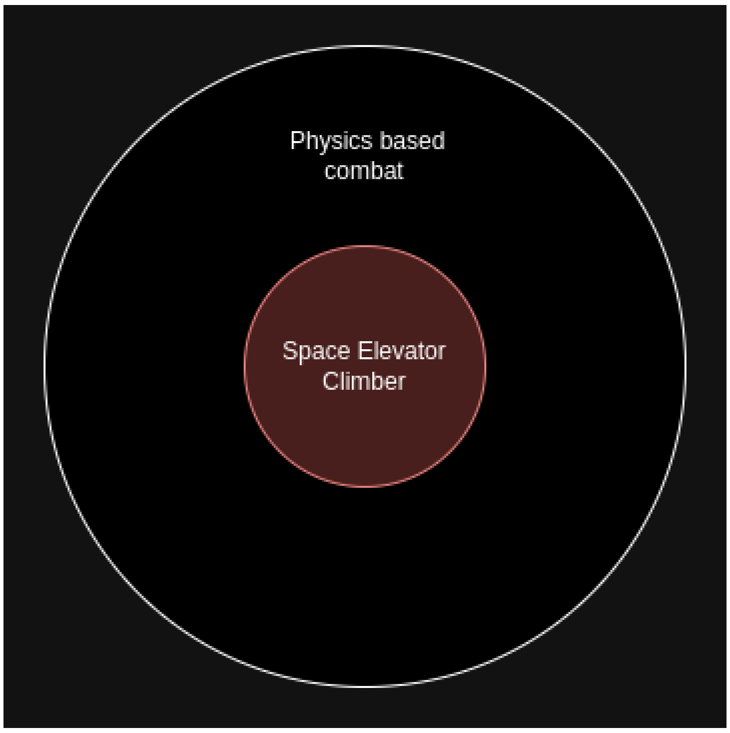
The combat is going to be the focus of the game. We want to make it very physics-based, not only to make the control of the mechanical arms feel satisfying but also to let the player make use of it in creative ways.
We want to associate mass with most objects in the game and have it take effect in various ways like how hard punches hit or how hard/far objects can be thrown. We also have the idea of a simplified air-drag simulation based on object shape and material also mostly affecting throwability. This last point goes well together with giving objects (mostly enemies) different parts with different structural integrity, so that the player is able to rip up (or rip out) parts that affect air-drag negatively or to rid enemies of their buoyancy. This should also make combat more interesting because it should allow the player to strategically target specific parts of enemies.

Inverse/Forward Kinematics

A secondary feature we want to take a look at is the use of Inverse and Forward Kinematics to make the arms and claws look good. This is something we want to experiment with out of interest and because we think it would greatly improve the look of the game. How much or even if it will be a technical challenge at all will reveal itself once we get to that part.

3 Big Idea Bullseye

The core idea of the game is the space elevator with its mechanical arms that are used both for movement as well as defense.
As a twist, we will implement physics based combat, where the movement and the effect of the claws is based on physical properties as well as the throwing and interacting behavior of the enemies.



4 Development Schedule

Goals

1. Functional Minimum
    a. Basic sprite for elevator with arms
    b. Basic movement of the elevator and arms
    c. Basic grabbing and throwing of enemies with arms
    d. Enemies that can be grabbed and attack elevator in a simple way
    e. Simple movement inside of elevator and press button to keep elevator going
    f. Cycle between movement in elevator and fighting
    g. Simple health bar
2. Low Target
    a. Basic, static environment
    b. Loading phase: grabbing of grabbables and counter how much is loaded
    c. Basic mission: load either cargo or people, no real impact on gameplay (always similar enemies)
    d. UI to indicate health of elevator and cargo
    e. Delivery of cargo

    f. Basic sound effects when grabbing enemies
    g. Different movement of arm depending on mass
    h. Flinging mechanic
    i. Final goal: maximum points
    j. Music
    k. Enemy dismemberment
3. Desirable Target
    a. Enhanced sprite for elevator and arms
    b. Changing environment (depending on height)
    c. Type of mission (cargo or people delivery) determines types of enemies that attack
    d. Cargo types imply story (radioactive material, food, politicians)
    e. Satisfying movement of arms
    f. Player inside of elevator tasked with fixing and fueling elevator
    g. Mounting points for extra stuff (artillery, shields, gadgets…)
    h. Shield that has to be timed to reflect incoming enemies
    i. Enhanced enemy sprites (space pirates, trash junkers, bodyguards, depending on height of elevator)
    j. Sound effects for space elevator (on hit, fueling, fixing)
    k. Collect money for final ticket
    l. Switch between different cables (locked by money), to reach final climb
    m. Different physical properties on different enemy parts
    n. Voluntary Mission Abort
    o. Complex enemy behavior (different behavior depending on enemy type, pirates attack alone, bodyguards more often in formation)
    p. Different themed music depending on height and location.
    q. Procedural generation of missions
    r. Player Character selection. Unlock new characters by winning. Different skin and special ability.
    s. Player can move outside of space elevator to scavenge for resources
4. Extras
    a. Online Cooperative Multiplayer
    b. People/Cargo smuggling
    c. Interactive Story (Mission dependent NPC dialogue etc.)

Tasks

Based on our goals, we have identified the following tasks for each milestone to achieve our goals:

Interims demo	**Responsible**	**Time estimate in h**	**Actual time spent in h**
1. Elevator sprites	Philipp	6	3
2. Elevator and Claw	Philipp	5	0.5

code			
3. Interior code	Natia	6	10
4. Interior interactables	Natia	6	8
5. Interior sprites	Natia	4	8
6. Environment sprites	Lukas	5	2
7. Enemy sprites	Lukas	5	4
8. Enemy code	Phil	10	25
9. UI graphics	Phil	2	
10. UI code	Phil	3	4
11. Code/Asset tuning	All	8	9
12. Presentation	Natia	1	1

Alpha release	**Responsible**	**Time estimate in h**	**Actual time spent in h**
1. Enemy AI	Phil	15	
2. Enemy dismemberment		4	
3. Claw physics	Lukas	10	7
4. Base station sprites	Lukas	6	6
5. Base station logic	Lukas	5	4
6. Loading phase code	Lukas	3	3
7. Mission generation		10	
8. Adaptive environment		7	
9. Mission abort		2	
10. Physics fine tuning		8	
11. Extra sprites		4	
12. Cable switching		4	

13. Presentation		1	

Playtest release	**Responsible**	**Time estimate in h**	**Actual time spent in h**
1. Equipment modules code	Lukas	12	4 (scope scaled back)
2. Eq. Mod. sprites	Lukas	4	1
3. Eq. Mod. UI		3	
4. Enemy part physics		9	
5. Sound effects		4	
6. Extra sprites		5	
7. Questionnaire preparation		2	
8. Presentation		1	

Final release	**Responsible**	**Time estimate in h**	**Actual time spent in h**
1. Better mission generation		10	
2. Complex enemy AI		10	
3. Movement outside of elevator		20	
4. Presentation		1	

Timeline

Our timeline looks as follows:

Working				
Interims demo	8.11.-14.11.	15.11.-21.11	22.11.-28.11	29.11.-5.12.
1. Elevator sprites	🟥			
2. Elevator and claw code	🟥	🟥	🟥	
3. Interior code			🟥	🟥
4. Interior interactables			🟥	🟥
5. Interior sprites			🟥	
6. Environment sprites				🟥
7. Enemy sprites	🟥			
8. Enemy code		🟥	🟥	
9. UI graphics	🟥	🟥		
10. UI code	🟥	🟥		
11. Code/Asset tuning	🟥	🟥	🟥	🟥
12. Presentation				🟥

Working					
Alpha release	6.12.-12.12.	13.12.-19.12.	20.12.-26.12.	27.12.-2.1.	3.1.-9.1.
1. Enemy AI	🟥	🟥			
2. Enemy dismemberment			🟥		
3. Claw physics improvement	🟥	🟥			
4. Base station sprites	🟥	🟥			
5. Loading phase code				🟥	🟥
6. Mission generation					🟥
7. Adaptive environment					🟥
8. Mission abort			🟥	🟥	
9. Physics fine tuning	🟥	🟥	🟥	🟥	🟥
10. Extra sprites	🟥				🟥
11. Cable switching				🟥	🟥
12. Presentation					🟥

Working		
Playtest release	10.1.-16.1.	17.1.-23.1.
1. Equipment modules code	🟥	
2. Eq. Mod. sprites	🟥	
3. Eq. Mod. UI	🟥	
4. Enemy part physics	🟥	
5. Sound effects	🟥	
6. Extra sprites	🟥	
7. Questionnaire preparation		🟥
8. Presentation		🟥

Working		
Final release	24.1.-30.1.	31.1.-6.2.
1. Better mission generation	🟥	🟥
2. Complex enemy AI	🟥	🟥
3. Movement outside elevator	🟥	🟥
4. Presentation		🟥

5 Assessment

In summary, the strengths of the game should be a fun way of fighting off enemies, while the micromanaging of the elevator should be a way to increase the challenge and make the player feel in control of their vehicle.
The game should be suitable for all ages and the intuitive control schemes should make it approachable for all levels of skill. That being said, players who are especially fond of the rogue-like genre and a good mix of dexterity and management based types of games will have the most fun.
The player will do multiple runs of the main gameloop until they have amassed enough skill and/or ressources and upgrade to tackle the final goal during which they will either win or fail and start over again.
The world that the player will get to explore should be in equal amounts fun to look at and atmospheric, which is why we are going for a light-hearted and novel approach of the otherwise very overdone post-apocalyptic setting.
The criteria on which we will judge our success are therefore:

1. Is the combat fun?
2. Does the combat benefit from the degree of physically-based systems in the background?
3. Is the game challenge balanced appropriately?
4. Did we successfully hit the genre target we were aiming for?
5. Does the game world feel unique enough and is fun to look at?

Milestone 2: Physical Prototype

1. Approach

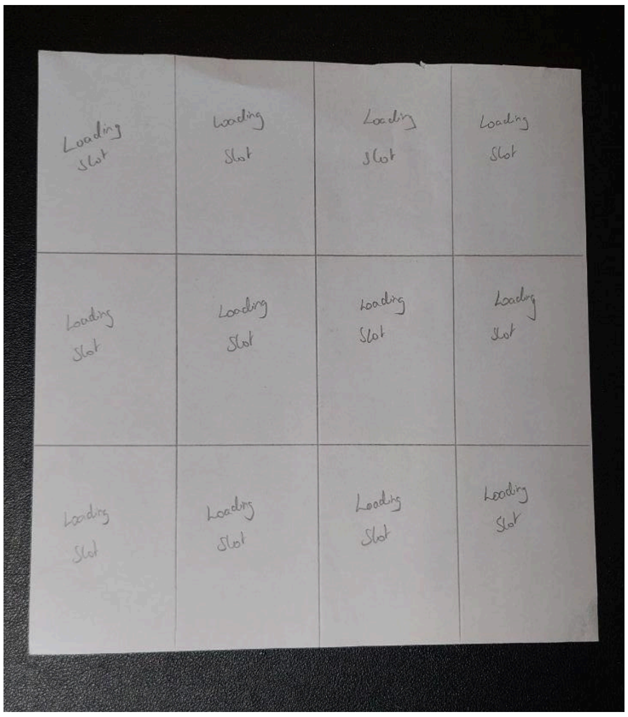
To create the prototype, we first gathered the most important aspects of our proposed game. These are:

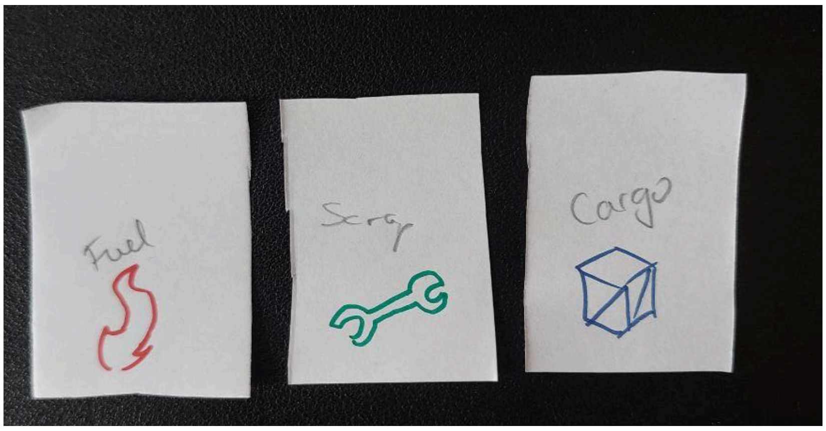
- upward movement of the elevator to deliver cargo
- downward movement of the elevator upon completed mission
- enemy types based on the height of the elevator
- physics based combat
- satisfying movement of the arm to defend against enemies
- flinging mechanic of arm
- loading phase

As the physics based combat as well as the satisfying movement of the arm are strongly bound to the actual digital implementation of our game, we did not find a good way to translate and incorporate this aspect into a physical prototype. One considered possibility was the use of an elastic rubber band based arm, which could translate the flinging

mechanic of the arm. This did not produce a satisfying result, so instead we opted to focus on our remaining aspects.
We built the prototype to include the three important phases of the later actual game:

1. Loading Phase: Here the player has to load up their cargo space with either cargo crates that they can deliver at their destination and exchange for money or they spend their precious cargo space on fuel which is needed for the climb itself.

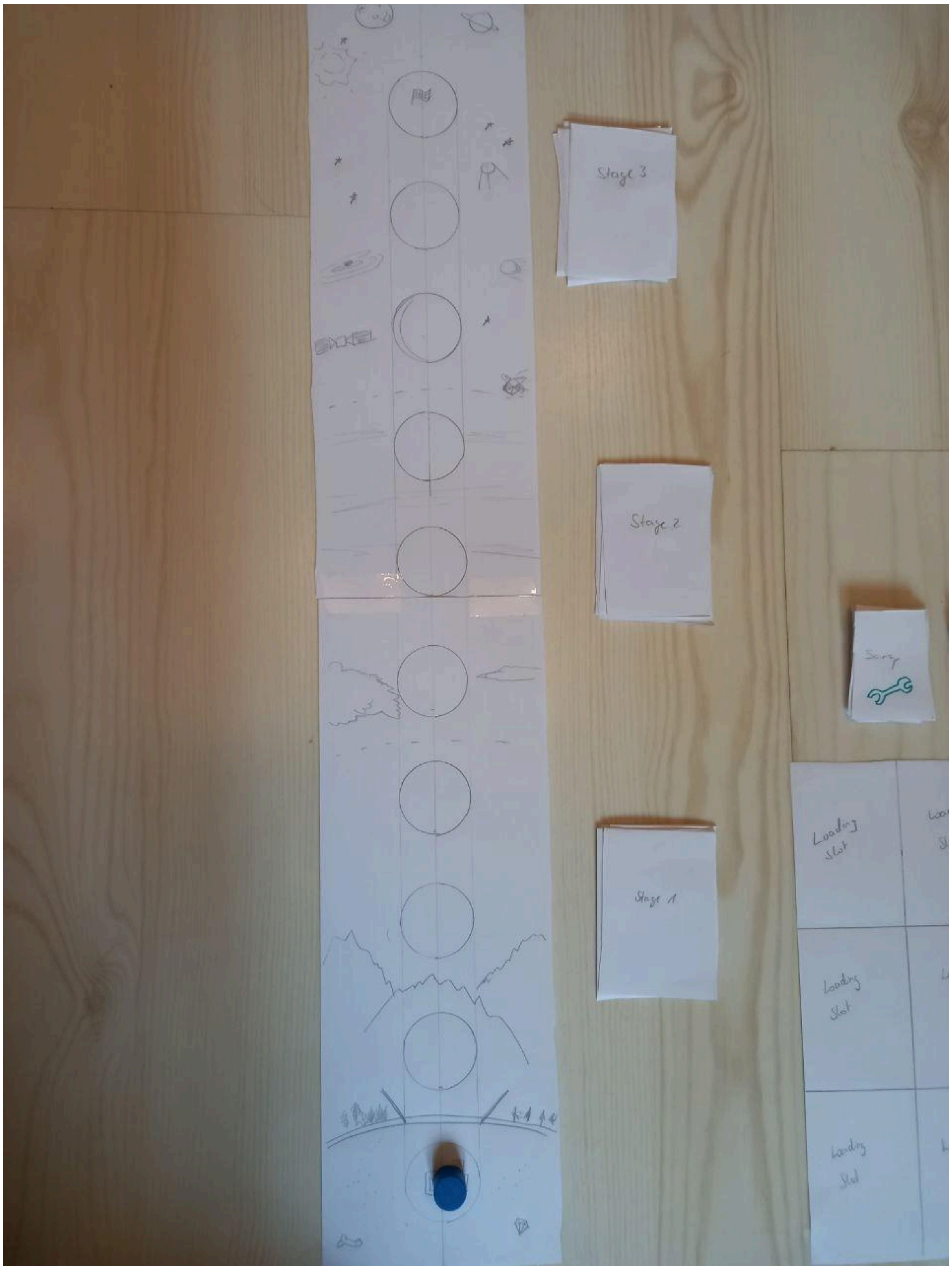




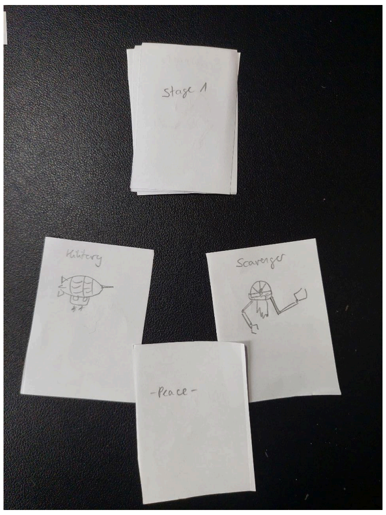
The first image shows the loading space with a total of 12 slots, while the second picture shows the items that can be put into these slots. In this phase, only fuel and cargo can be placed inside the slots, as scrap can only be obtained through enemy encounters.

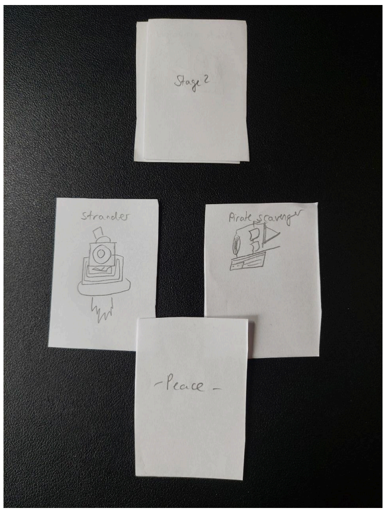
2. Climbing Phase: Here the Player can move one field up on the square which requires one fuel card. When reaching a new field, an event card is drawn which may result in

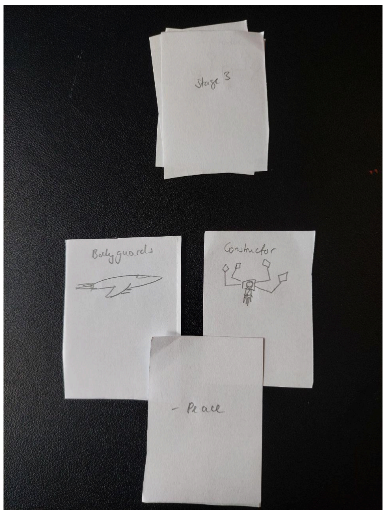
an enemy attack. Each phase consists of 6 different cards, 4 being enemies and 2 being “Peace” card, in which case no enemy attack occurs.



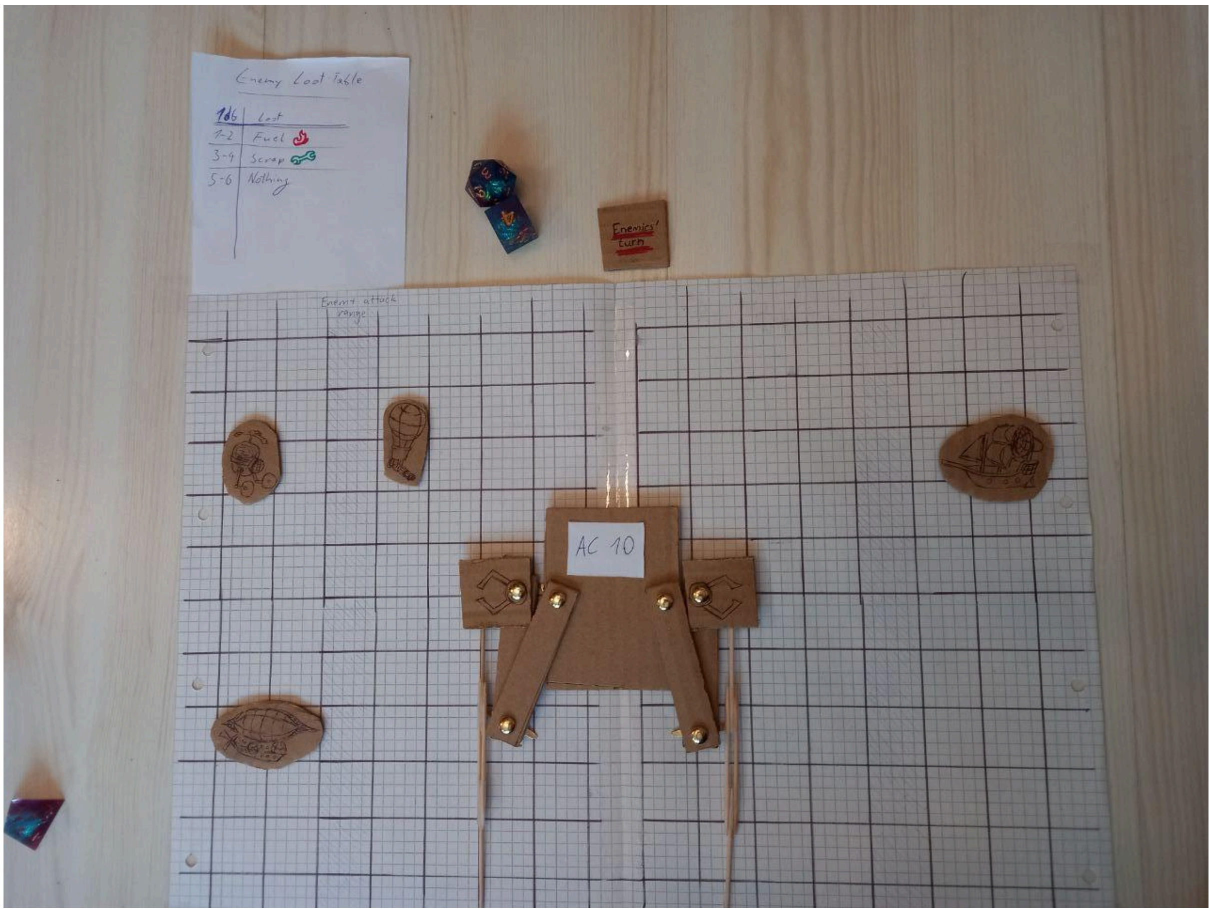
The following pictures show the cards for each stage in detail:







3. Combat Phase: When enemies attack the player will switch to the combat playboard. Here another player who acts as the enemies can attack while the elevator player defends themselves.


Enemy Loot Table

1d6	Loot
1-2	Fuel
3-4	Scrap
5-6	Nothing

2. Prototype Rules of Play

The rules are as follows:

- Since cargo space is limited, during phase 1, the player will have to make a choice whether or not they fill their cargo slots with enough fuel for the whole climb or if they take the chances of looting fuel during the climb but can therefore load more cargo which means more profit. Once the inventory is all filled up they continue with phase 2.
- Now, the game will alternate between the second and third phase until the player either loses all cargo or reaches the last field on the map.
- Moving up one field requires spending one fuel card and forces the player to draw an event card of the associated climbing height they are currently at. The card can either result in "peace" (nothing happens) or an enemy encounter.
  In the case of an enemy encounter, 1d4 is rolled to determine the number of attacking enemies. Then, the player switches to the combat scene for the encounter.
- During combat, one supporting player, from now on referred to as "evil player" may control the enemies while the other is controlling the arms of the elevator. Combat is turn-based and alternates between the evil player's and the elevator player's turn. On their turn, the evil player can either move one enemy up to 2 fields and attack (if within range, marked on the board) or spend their turn to introduce one new enemy from their pool of enemy tokens. On the elevator player's turn, they can move one arm up to 4 fields and if they reach an enemy token they automatically destroy it by crushing the enemy. Only one of the two arms can be active at a time. If the player switches control from one arm to the other, the former has its position reset to hug the elevator again. When an enemy is crushed, the player must roll 1d6 on the loot table to determine whether they receive fuel, scrap or nothing. When an enemy attacks, the evil player rolls 1d20. If they manage to roll equal or higher than the Elevators Armor Class (AC), then they cause damage, which manifests itself in the player losing one cargo card. When all enemies are defeated or the player loses all cargo, combat is concluded and the player either returns to the Climb Map or has lost the game.
- After making it to the last field at the top of the Climb Map, the player can exchange all remaining Cargo Cards for one Money card each and then drop back down to base.
- Once back down the player can spend 5 Scrap Cards to upgrade their AC, buy Scrap with their earned money (exchange rate 1:1), and start another climb. When accumulating 20 money cards, the player wins the game.
- The amount of enemies is increased for each encounter by 1 per 5 money cards the player has. E.g when in possession of 5 Money Cards, on each encounter 1d4+1 is rolled.

A small overview of the available resources and their function:

- **Cargo Cards**: These represent the players health points as well as being the goal of the climb. Bringing these cards to the highest field on the map lets the player exchange them for money cards and win the run.
- **Fuel Cards**: One of these each is required to move one field on the Climb Map.
- **Scrap Cards**: These can be looted from enemies (or possibly bought during the loading phase for money). 5 of these can be used to increase the Elevators AC by 1.

- **Money Cards**: Collecting 20 of these represents gathering the entry fee for the final climb, making the player win the game. They can also be exchanged for 1 scrap during the loading phase.

3. Verdict

Utilizing dice rolls, similar to the way they are used in Dungeons&Dragons, gave the game a nice element of randomness and helped a little against it feeling repetitive. Also rolling dice is fun.
Overall however, it does definitely feel more like an analog simulation of what is obviously supposed to be a videogame rather than a functional board game with replay value, especially due to the very asymmetric nature of the two players' roles. The second player who we aptly named the "support player" is needed only to simulate a basic enemy AI, which would have been very hard to do otherwise without introducing a lot more complicated dice rules.
While we would not play it for fun in our free time, it was still a nice way of looking at the way our planned ressources and the three different phases of gameplay are interacting.

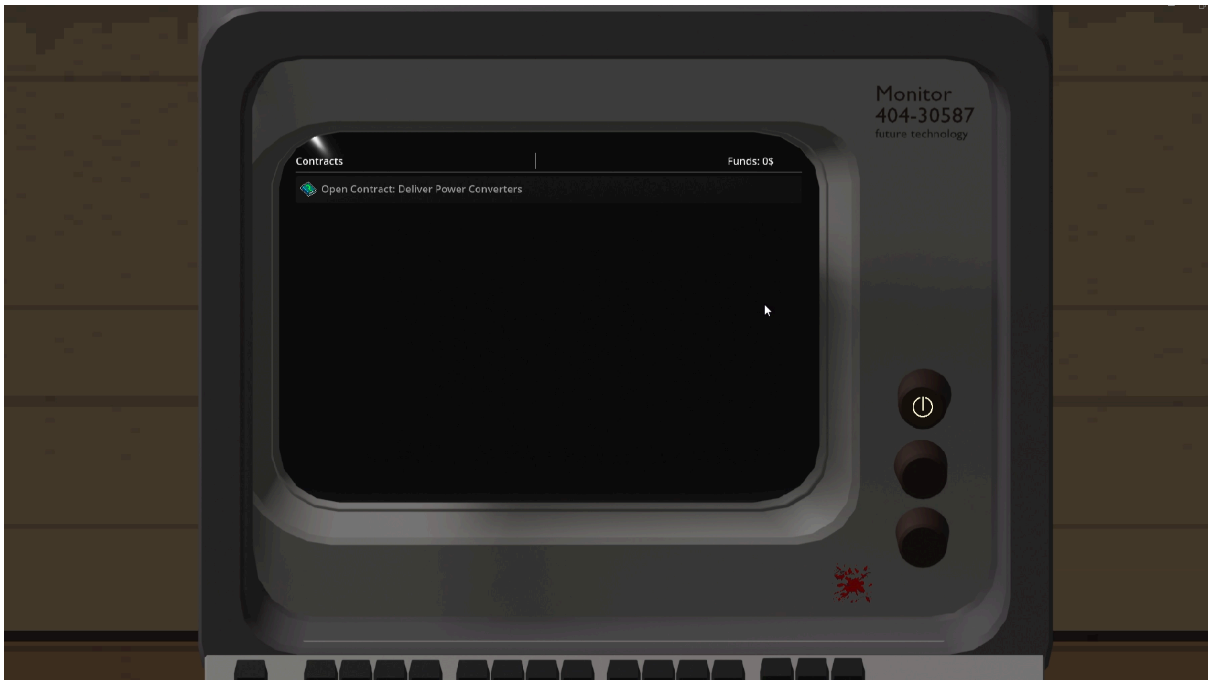
4. Lessons learned

- Having either nothing or an enemy encounter happen, makes the climb phase feel a little bare bones. While the digital game version will include the whole "repair and manage the elevator" mechanic during these climbing phases as well, the idea of also introducing small random encounters other than just attacking enemies came up a lot.
- The choice to make combat turn-based was only due to the limitations of the analogue pen-and-paper medium, but it still confirmed our suspicion that turn-based combat definitely is not the way to go for the actual game.

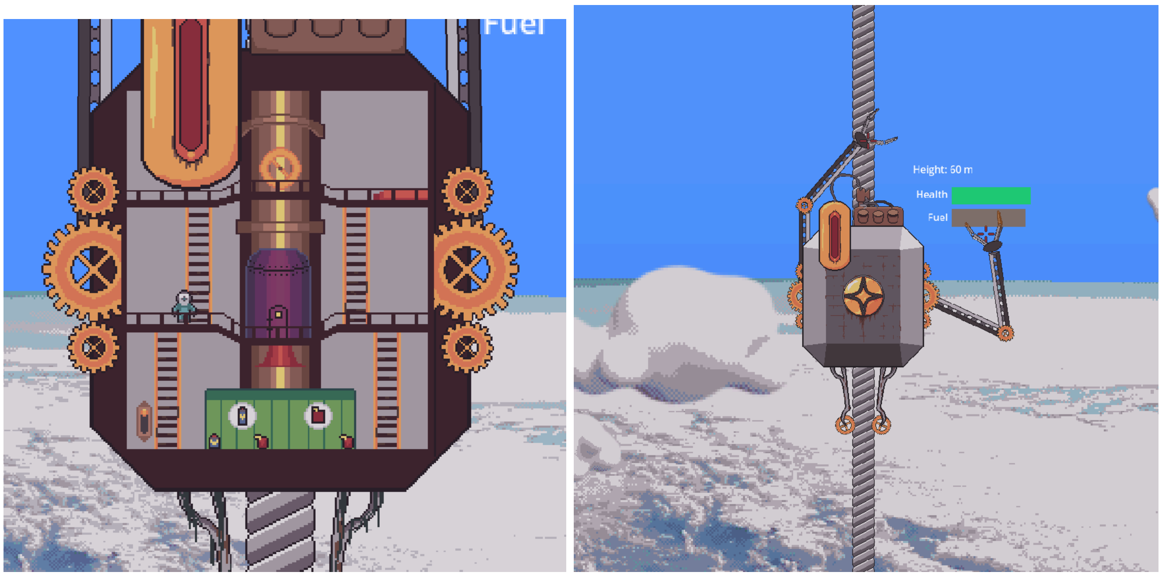
Milestone 3: Interim Report

1. Progress report

Most of the functional minimum and low target has been implemented, alongside select things of the desirable target, though some parts have been altered for a timely implementation.

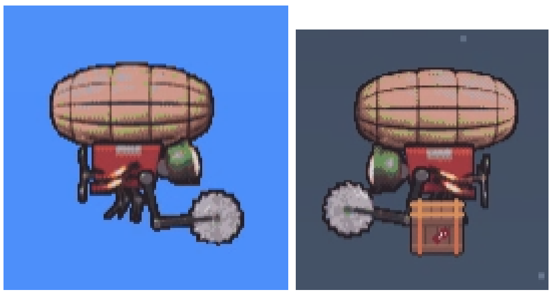


The game starts off with a loading phase, which is now a computer terminal where the player first selects a mission, and the amount of fuel and cargo they wish to take (limited by a maximum inventory size). After clicking on a button to start the mission, the main gameplay loop begins. In this new scene the player can toggle between an interior and exterior view of the space elevator as it climbs the cable, which also changes the gameplay.

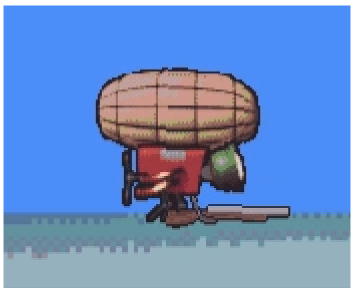


In the interior view, the user can move a small player character with standard side scroller controls, picking up fuel and scrap from a deposit interface to refuel and repair the elevator respectively. The player is alerted of incoming enemies with a light and sound effect. In the middle a control station is situated, interacting with which switches over to the exterior view. While in the exterior view, the player can control the two mechanical arms of the elevator as described previously, i.e. moving the mouse moves a target point which the arm closest to it will move towards. As such only one arm is controlled at a time, with the other returning to a rest position. Using left-click, enemies can be grabbed and swung around as a blunt weapon. Alternatively, holding space will lock the arm in its current position and draw a line from the claw to the cursor. Letting go of space will then unlock the arm and accelerate it towards the target faster than possible through mouse movement alone, automatically letting go of the enemy once the maximum speed in the desired direction has been achieved. The elevator climbs automatically, but stops and sounds an alarm once a wave of enemies approaches. These waves occur at random intervals, and their size is determined by the current height of the elevator. Currently, three types of enemies have been implemented:

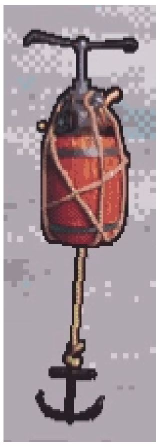
- *Melee*: These enemies will move in close to the elevator and attack it with sawblades. If left to attack, they will extract cargo from the player's inventory and make off with it.



- *Ranged*: These enemies pick a point (within grabbing range) around the elevator and move towards it. Once reached, they will begin to periodically shoot the elevator with projectiles.

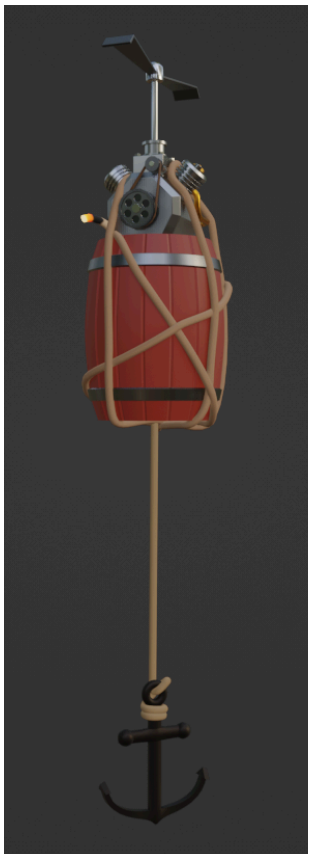
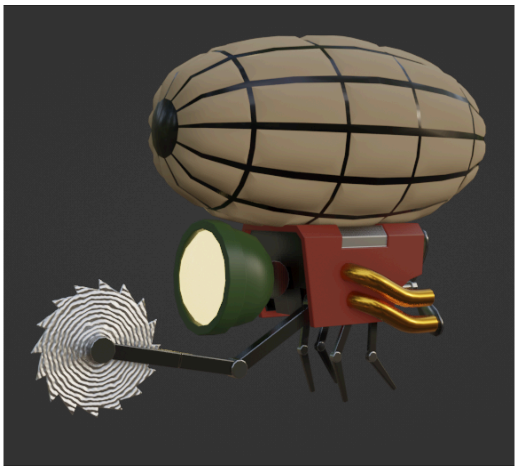


- *Bomb*: These enemies move similarly to the melee variant, however they explode on impact, and grabbing them with an arm will disable that arm, forcing the player to repair it before it can be operated again. These enemies can be disarmed by either launching another enemy at them, or by grabbing the anchor hanging from them, making their engine fly off and the barrel fall down.

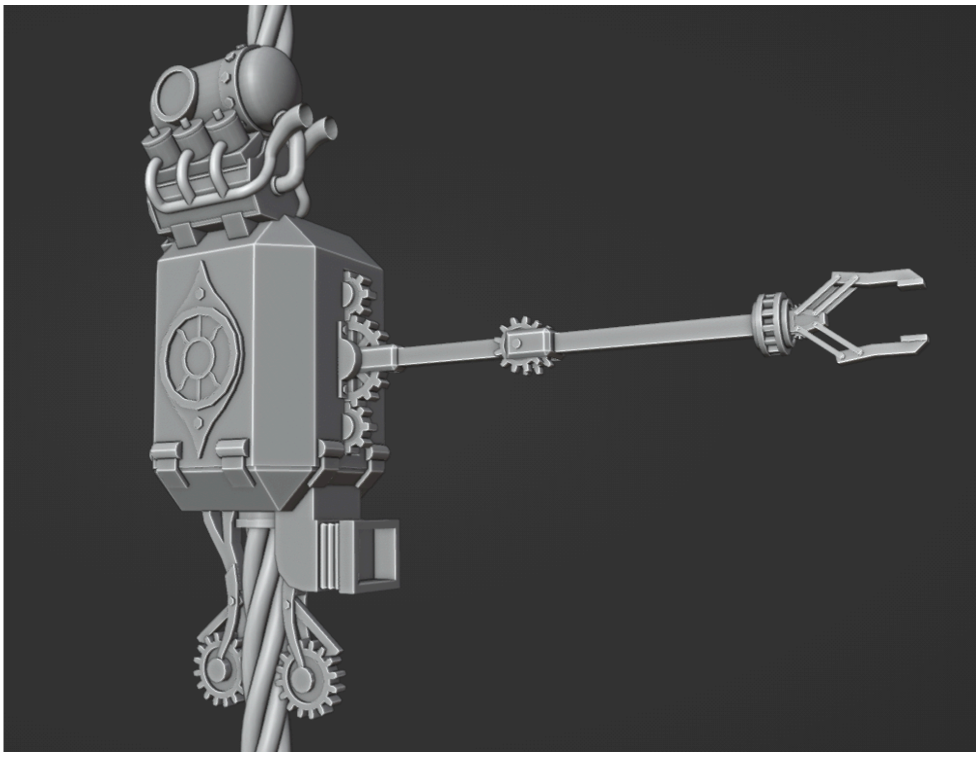


Once all the enemies of a wave have been cleared, the player needs to switch to the interior view and pull a lever to continue the climb. At the very end, the elevator enters a space station and all remaining cargo can be sold off, increasing the player's funds, after which the elevator drops back down and the player can choose another mission to repeat the process.

In terms of assets, the enemies were modeled in 3D and can be rendered in a variety of perspectives and animations. To maintain a somewhat cohesive aesthetic, enemies are rendered with a black outline and a custom dithering tool is used to reduce all sprites to the same palette.

Work has also begun on a model of the elevator, though gameplay implementation takes priority so whether this will be finished in time for the final product is unclear.



Preliminary sound design exists for all enemy interactions (attacking, getting hit by another enemy, balloon popping, etc.) and the elevator’s movement, though music is still lacking. Here is an overview, over what features were implemented, green indicating that the feature has been implemented, yellow features that are half implemented and red that the feature is yet to be implemented:

1. Functional Minimum
    a. Basic sprite for elevator with arms
    b. Basic movement of the elevator and arms
    c. Basic grabbing and throwing of enemies with arms
    d. Enemies that can be grabbed and attack elevator in a simple way
    e. Simple movement inside of elevator and press button to keep elevator going
    f. Cycle between movement in elevator and fighting
    g. Simple health bar
2. Low Target
    a. Basic, static environment
    b. Loading phase: grabbing of grabbables and counter how much is loaded
    c. Basic mission: load either cargo or people, no real impact on gameplay (always similar enemies)
    d. UI to indicate health of elevator and cargo
    e. Delivery of cargo
    f. Basic sound effects when grabbing enemies
    g. Different movement of arm depending on mass
    h. Flinging mechanic
    i. Final goal: maximum points
    j. Music
    k. Enemy dismemberment
3. Desirable Target
    a. Enhanced sprite for elevator and arms
    b. Changing environment (depending on height)
    c. Type of mission (cargo or people delivery) determines types of enemies that attack
    d. Cargo types imply story (radioactive material, food, politicians)
    e. Satisfying movement of arms
    f. Player inside of elevator tasked with fixing and fueling elevator
    g. Mounting points for extra stuff (artillery, shields, gadgets…)
    h. Shield that has to be timed to reflect incoming enemies
    i. Enhanced enemy sprites (space pirates, trash junkers, bodyguards, depending on height of elevator)
    j. Sound effects for space elevator (on hit, fueling, fixing)
    k. Collect money for final ticket
    l. Switch between different cables (locked by money), to reach final climb
    m. Different physical properties on different enemy parts
    n. Voluntary Mission Abort
    o. Complex enemy behavior (different behavior depending on enemy type, pirates attack alone, bodyguards more often in formation)
    p. Different themed music depending on height and location.

q. Procedural generation of missions
r. Player Character selection. Unlock new characters by winning. Different skin and special ability.
s. Player can move outside of space elevator to scavenge for resources

2. Unexpected Hardships

The implementation of the main feature, the robotic clawed arms of the elevator, were more complex than originally expected. A few design iterations went into them until the current implementation was settled on (for now).
Currently, the arms work by having hidden skeleton based arms driven by inverse kinematics to reach a target node that is mostly attached to the mouse pointer but does collide with the body of the elevator itself to make reaching into it impossible. These hidden arms do not have any physical properties and are only used by the visible physics-based arms as a target position. The physical arms do have mass and are part of the physics engine, therefore able to physically interact with certain game objects like the enemies. They are also mostly bound to some physical limitations like their acceleration and inertia, although the latest version is quite snappy already. At first, the arms were able to collide with most other physics bodies, but that did not work out to benefit the game as we were mostly bug fixing all the edge cases in which we did not want to have collision interfere with the arms movement. For example, due to the arm (not the claw) being able to collide with enemies it was a very common issue that the player would wedge enemies under the elevator's arm and thus making it impossible to reach them anymore. In the end we settled on very minimal collision of the arm or claws, which did make for the best 'feel' while also forcing the player to actually use enemies(and their mass) as weapons against each other, as originally envisioned, rather than just thrashing about with the claws themselves.
The claws and arms are now relatively stable and soft-locking by jamming the arms into other static bodies is also not possible anymore.

3. Design revisions

Due to the scale of current enemies, the initial plan to make all of them consist of separate pieces with different physical attributes, which can all be independently dismembered, was scaled back a lot. Currently the melee and ranged types have balloons on them which pop when grabbed, at which point they no longer hover when dropped, and the bombs can be somewhat dismembered by pulling their anchor, however both of these are a far cry from the previously planned mechanic. We intend to implement larger boss-like enemies where such a system would make a lot more sense.
Something else that evolved a lot was the flinging mechanic. The initial idea was to make the fling-input switch the control scheme such that the upper arm points towards the cursor and the lower arm acts like a spring of sorts, letting you punt enemies in a satisfying way. Further contemplation quickly revealed that this would make it extremely hard to actually aim the flung enemy this way. Once the physically simulated arms were implemented, a quick test where the input merely locked the target in place until release showed that this was a much more viable approach.

The loading phase where the player has to grab and insert all the cargo manually was abstracted as a terminal to meet deadlines. This interface will certainly stay for mission selection, but the manual loading phase is still planned to be implemented later.

Milestone 4: Alpha Release

1. Progress report

Overview

As we had already completed the functional minimum in the last sprint, it is omitted here, and only the layers 2 and 3 are shown here:

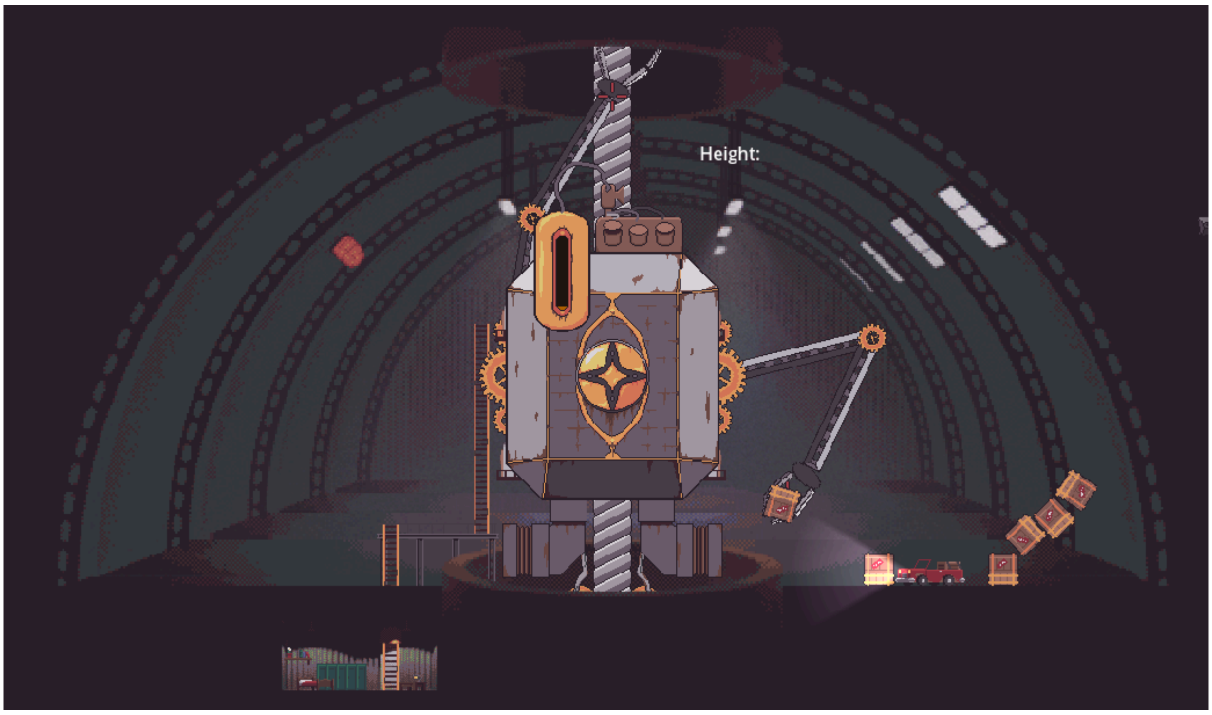
2. Low Target
    a. Basic, static environment
    b. Loading phase: grabbing of grabbables and counter how much is loaded
    c. Basic mission: load either cargo or people, no real impact on gameplay (always similar enemies)
    d. UI to indicate health of elevator and cargo
    e. Delivery of cargo
    f. Basic sound effects when grabbing enemies
    g. Different movement of arm depending on mass
    h. Flinging mechanic
    i. Final goal: maximum points
    j. Music
    k. Enemy dismemberment

3. Desirable Target
    a. Enhanced sprite for elevator and arms
    b. Changing environment (depending on height)
    c. Type of mission (cargo or people delivery) determines types of enemies that attack
    d. Cargo types imply story (radioactive material, food, politicians)
    e. Satisfying movement of arms
    f. Player inside of elevator tasked with fixing and fueling elevator
    g. Mounting points for extra stuff (artillery, shields, gadgets…)
    h. Shield that has to be timed to reflect incoming enemies
    i. Enhanced enemy sprites (space pirates, trash junkers, bodyguards, depending on height of elevator)
    j. Sound effects for space elevator (on hit, fueling, fixing)
    k. Collect money for final ticket
    l. Switch between different cables (locked by money), to reach final climb
    m. Different physical properties on different enemy parts
    n. Voluntary Mission Abort
    o. Complex enemy behavior (different behavior depending on enemy type, pirates attack alone, bodyguards more often in formation)
    p. Different themed music depending on height and location.
    q. Procedural generation of missions

r. Player Character selection. Unlock new characters by winning. Different skin and special ability.
s. Player can move outside of space elevator to scavenge for resources

Hangar

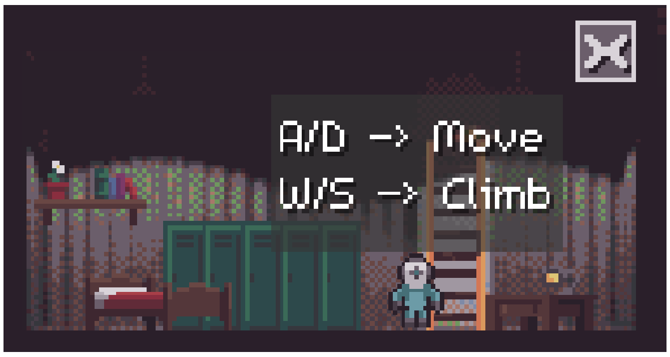
After a mission has been selected and the inventory contents chosen, the user is sent to a hangar scene, where the player character starts out in a little living quarters area underground.



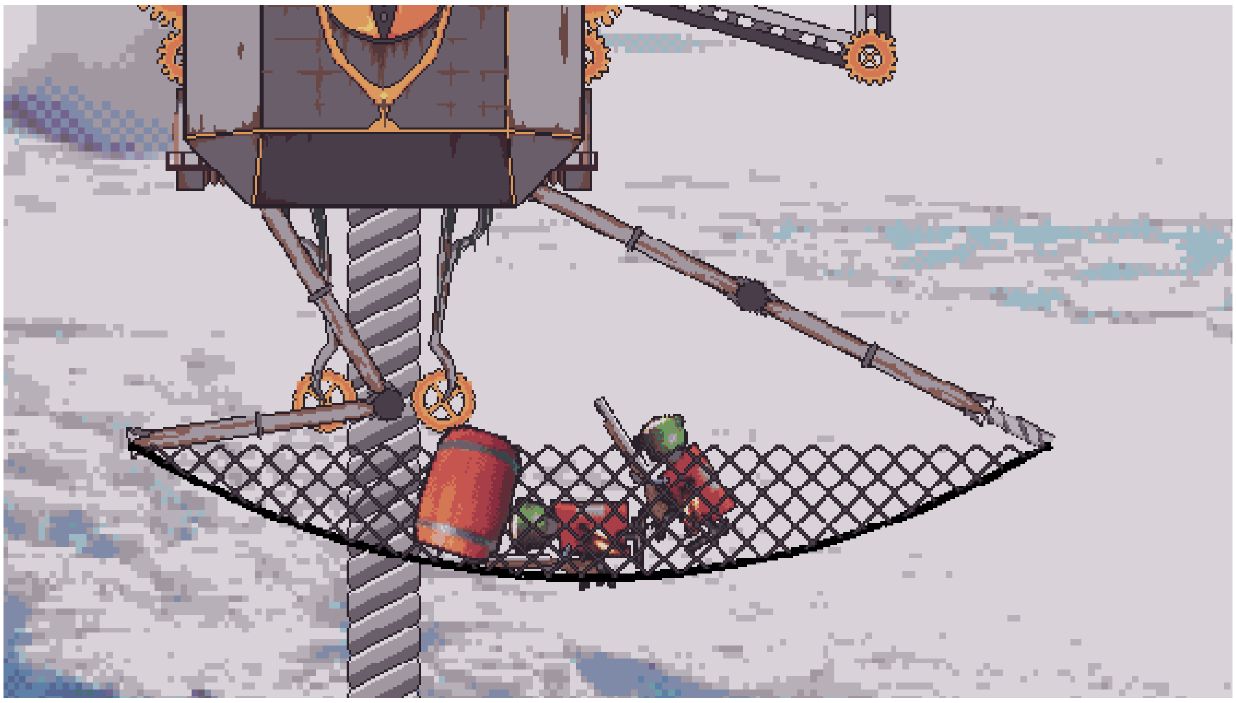
Once they climb up, enter the elevator, and take control of the arms, they must deploy the item chutes and use the arms to load in the contracted cargo. After all crates have been loaded, a button to open the hangar's hatch unfolds on the side, and the provided car has to be thrown at it to progress. This was intended as an implicit way to teach users how to throw things. After the hatch has opened, the player can unlock the brake and take off, leading to the main level.
In terms of visuals, the hangar has been modeled in 3D and rendered in a series of slices which are then moved around using a custom parallax script, giving the background more depth.

Tutorial

In order to keep the implementation simple, tutorials currently consist of simple pop-ups which appear on a trigger, like the player entering an area or picking up a specific item.


A/D -> Move
W/S -> Climb

They can be closed by clicking the X and the savefile stores whether a specific tutorial has been completed yet, such that it doesn’t pop up again on the next run.

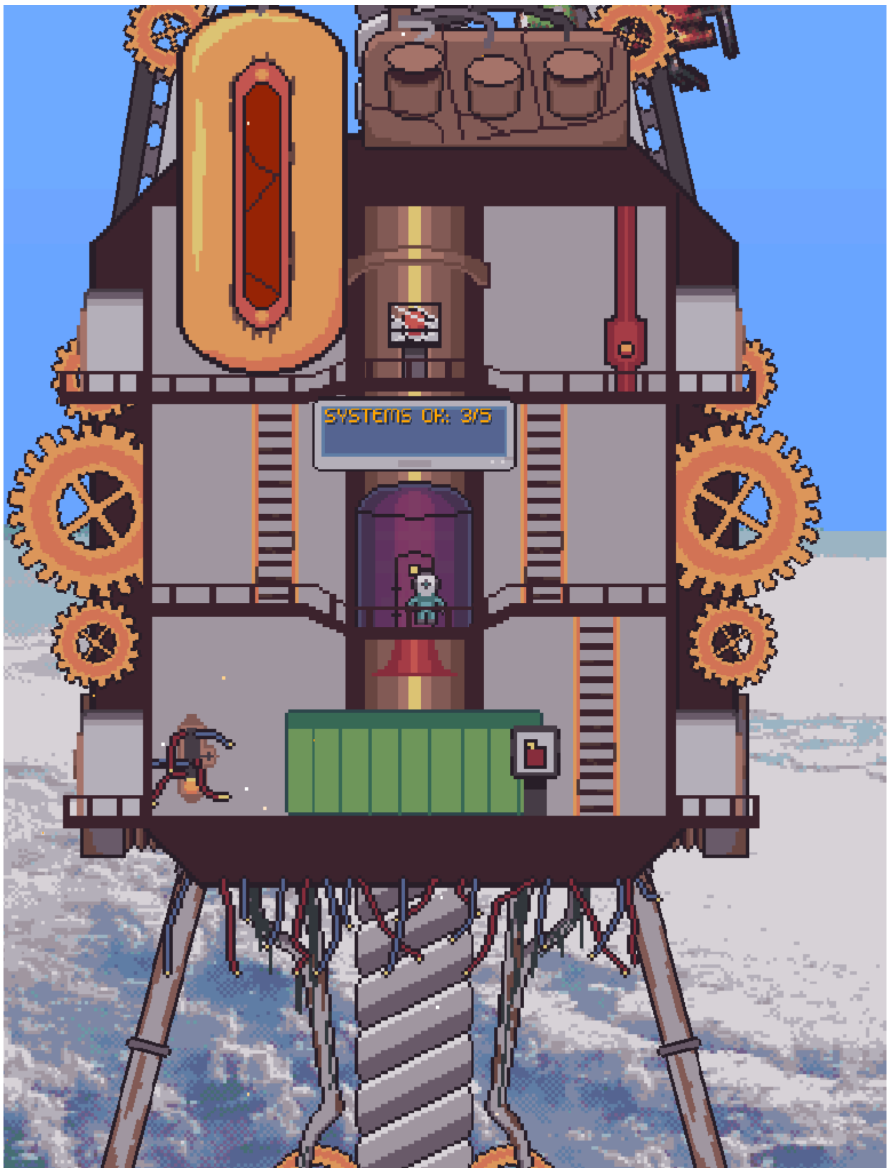
<u>Trawl Net</u>
One feature implemented due to feedback received at the last presentation is the trawl net. It can be used to catch enemy remains in order to salvage them for fuel and scrap or be used to just store parts of dead enemies to have objects in reserve for throwing at new enemies later on. It can be unfolded while in control of the arms by pressing Q and maneuvered left and right with A and D. The net spans between two arms which are controlled completely by inverse kinematics.
To implement the net, we first needed to implement a stable physically simulated rope. A few ways of doing that were explored but in the end the simplest one turned out to be the most practical one: a simple system of chained rigidbodies.
Using a chain of rigidbodies instead of for example a Verlet Integration has the advantage of all parts already being present in the physics engine of Godot and thus interaction with all other physics objects in the game works out of the box. The disadvantage of this approach however was that getting the rope to be stable enough was a little tricky and required a process of fine-tuning a lot of values like the rope segments mass distribution, joint bias, angular and linear damping and so on. Finally a general purpose rope was implemented which could be used in various ways.
For the trawl net we then span a rope between two static bodies and a textured polygon is drawn between the vertices of the line that draws the rope which results in our trawl net.


<u>Diegetic Health</u>
We wanted to give the player more interactions inside the elevator as well as make the health of the elevator diegetic and more directly controllable by the player, rather than only being a health bar. This is why we made 5 of our components destructible: the two arms have already been destructible in our last milestone by grabbing a bomb enemy. Additionally, every time the elevator takes damage from enemies, either the engine, the net or the brake

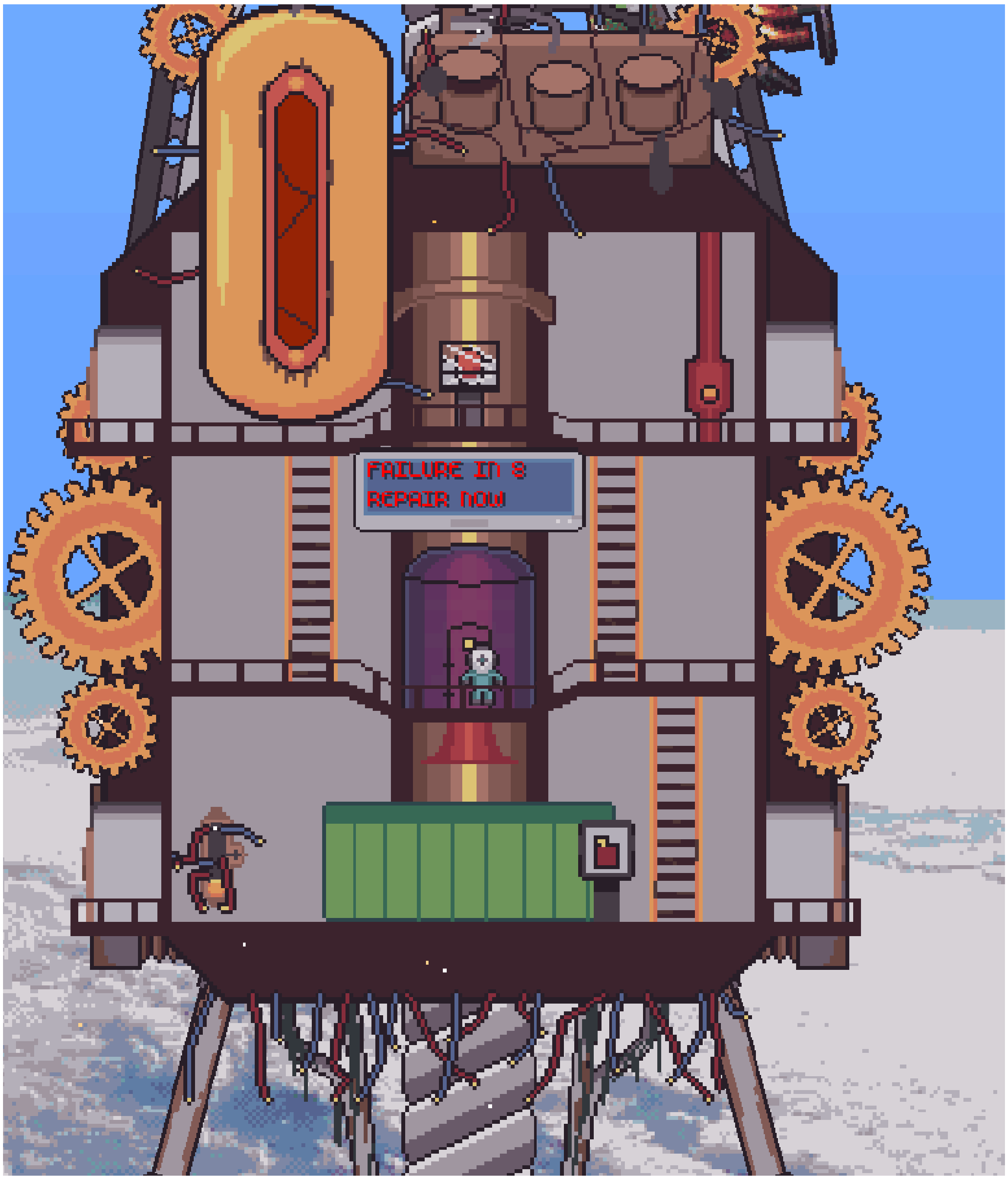
will take the respective damage. The module, to which the damage is delegated is chosen randomly. We considered also making the arms a part of these randomly destructible components, however decided against it, as the arms are such an integral part to the gameplay and should only break in case of misuse by grabbing a bomb.
Each of the modules has a health of 7. A saw enemy deals 1 damage to the elevator per attack, while a bomb deals damage depending on its distance to the elevator, which considering damage to the elevator always has to occur in its vicinity and is therefore always going to be around its maximum value of 150. Gun enemies deal 7 damage to the elevator on hit.
The damage is propagated to the components. Each of them has three states: functional, damaged and broken. They can be differentiated with visual cues, as well as functionality. A component is considered to be damaged when it has less than its maximum health, but more than 0. Once its health reaches 0, the component is considered to be broken.
Upon becoming damaged, the components can be repaired by interacting with them for 2 seconds while holding “R” and holding scrap. In this state, the functionality is not impaired. Once the component is broken, the following functionality impairments can be observed:

	Net	Engine	Brake	Arm
Broken	Drops down, not movable/ retractable	Continuously leaks fuel	Cannot switch in the fastest mode	Not movable

To repair a broken component, the player has to interact with it for 4 seconds while holding scrap and pressing “R”.
In the game, a monitor displays the number of operational components. This monitor, as well as the different visual cues for the damaged/broken states can be seen in the following image, where both the net and brake a broken while the engine is only damaged:

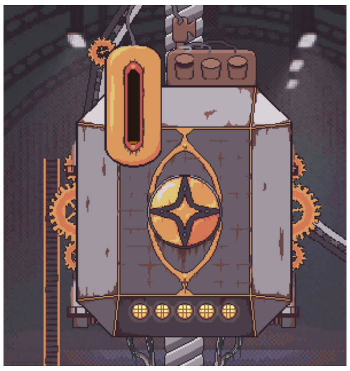


The color of the font color on the monitor changes depending on how many systems are operational. In case all 5 components are operational it is green, once 1 component breaks it changes to orange. Once 3 out of 5 components break, the font color on the monitor changes to red, a timer of 20 seconds is started, alongside an alarm sound. In this time, the player has to repair at least one component:



Otherwise the game is over and the process is wiped.
Originally, the game over condition was only triggered when all 5 components were broken, however this proved to be too easy to beat, as it is relatively simple to not break the arms, so we decided to lower the number of components that need to be broken.

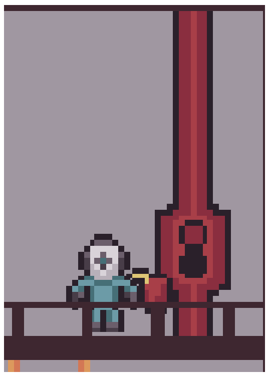
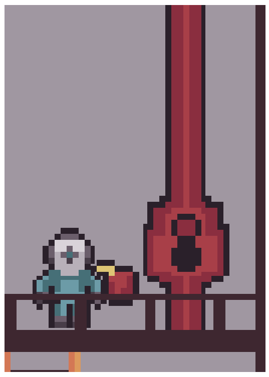
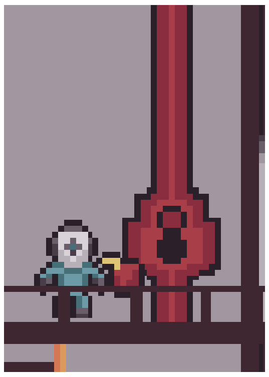
During our own playtests, we noticed that it could be quite difficult to notice if something breaks in the middle of an enemy attack, as the display of the number of operational components is only visible inside of the elevator. This meant that it could sometimes be relatively surprising when the game over condition was triggered. To counteract this, we added 5 lamps on the outside of the elevator, which show the amount of functional components while inside the arm station. Once a lamp is turned off due to a new broken module, a short sound effect is played to direct the players attention even more to the changed state of the elevator:



<u>Interior rework</u>

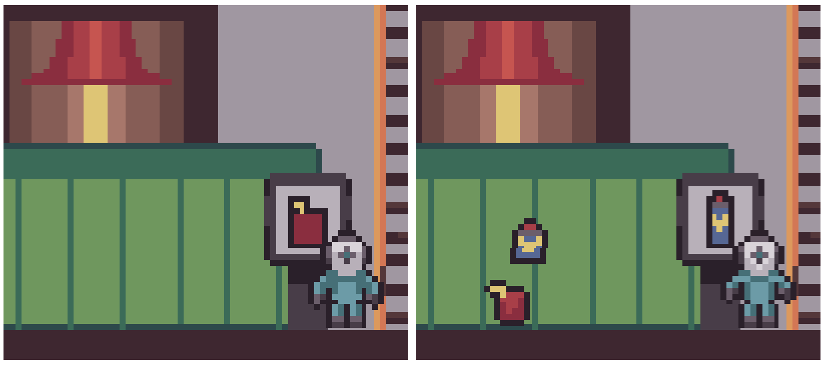
We reworked and extended the interior of the elevator to give the player more interactivity with the elevator.

The fuelling mechanic was reworked: previously, the player had to drop a fuel canister in a designated area and then press a button to fuel the engine. This was not very intuitive, so we changed it to resemble a smaller tank that has to be filled with fuel and is then handed over to the engine tank automatically. Fueling is done by holding a fuel canister, standing next to the fuelling station and pressing “E”. This starts a 3 seconds timer. If the player interrupts the fuelling process due to leaving the vicinity of the fueling station or not pressing “E”, the fueling process is concluded. The sprite of the fueling station is used to visualize how long the player has already been interacting with it:

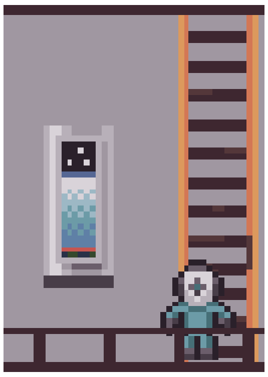
Interaction time	< 1s	> 1s, < 2s	> 2s, <= 3s
Sprite			
Refuel amount	10	12.5	15

We also wanted to ensure that players are aware that they are missing some fuel when they do not interact for the maximum duration. The amount of fuel differs depending on how long the player interacts with the fueling station, and a splashing animation as well as a splashing sound play when less than the maximum amount of fuel was refueled.

Another reworked component is the dispenser. Previously, we had two buttons which would either spawn fuel or scrap. We added a terminal, with which the player can either interact by pressing “S” which will cycle through the available resources (fuel and scrap) and “E” to dispense those items. Dispensing only works when the respective item is actually in the inventory and otherwise an error sound effect is played.

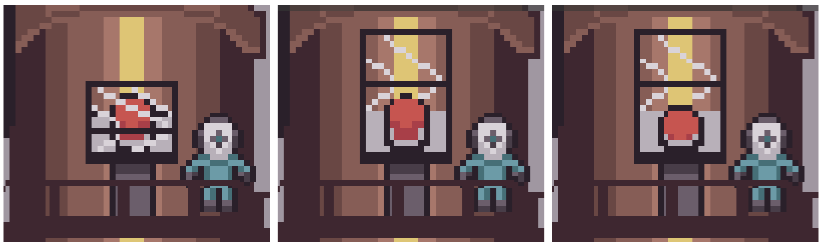


As the health is now diegetic, we also integrated the height meter in a diegetic manner:



The diegetic health system introduced some new breakable modules outside of the elevator. To repair them, we added a jetpack (see next section), as well as doors to the elevator that automatically open when the elevator is standing and the player approaches them, so that the player can leave the inside of the elevator and repair modules on the outside.

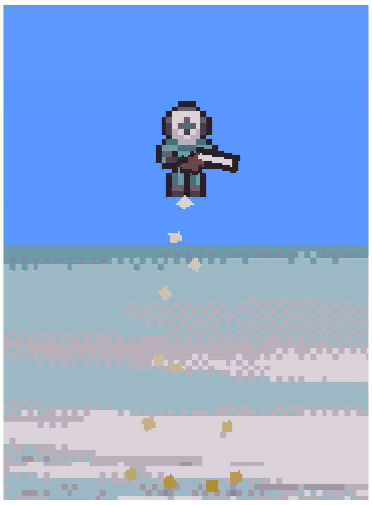
The breakable modules introduced with the diegetic health can, in theory, lead to a softlock, for example when the engine is broken and the player has no access to any new fuel. In this case, we wanted to give the player the opportunity to abort the run, so we introduced the “Drop Button”.



To make sure the player does not press the button by accident, it has to be interacted twice by pressing “E”, once to open the glass box surrounding it and then once more while the box is open to press the button. Leaving the vicinity of the drop button while it is open will automatically close the box.

<u>Jetpack</u>

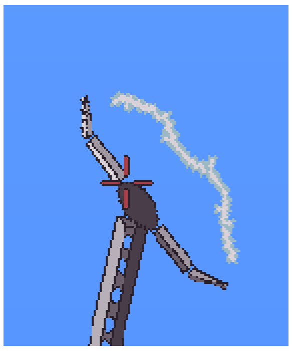
To facilitate repairing things on the outside of the elevator like the net or engine, we’ve given the player a jetpack which deploys automatically once they step outside of the elevator. The functionality of this is rather simple, letting go of any direction keys will make the player simply hover in place. Moving while in jetpack-mode has a lot more drag than otherwise, giving the whole movement a “spacey” feel, which to be honest makes most sense once the elevator reaches space. There is currently no limit to the time they can spend flying, though this will be changed in the future. As long as the player has the jetpack equipped, they also wield a small firearm which can be used to take down enemies if the arm on that side has failed, as otherwise the player would have to sit and watch as those enemies slowly drain the elevator’s health. The gun is also fun to shoot with and to make aiming possible the camera uses a ‘peek mode’ while the gun is equipped which means that it follows the cursor a bit.



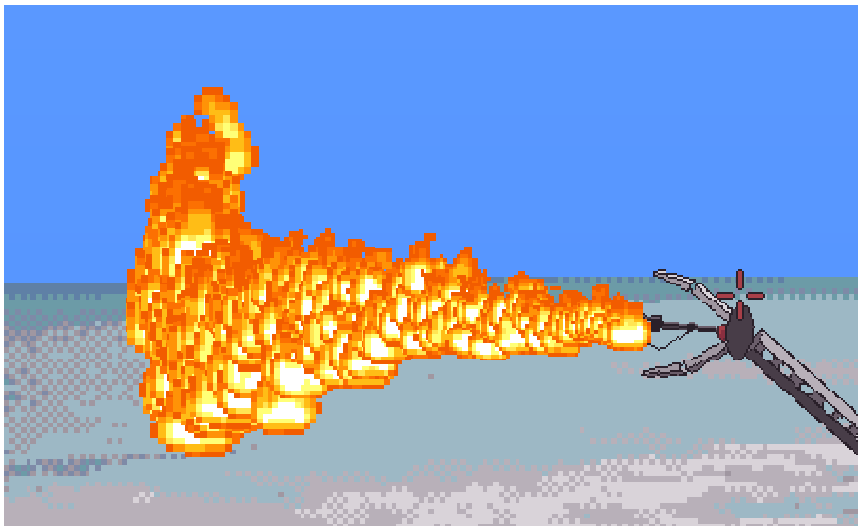
Arm modules

While we initially planned to have modules for the arms and the elevator itself, we opted to focus on only arm modules for now. Currently, there is a shield, flamethrower, and arclight projector module, and they can be activated while controlling the arms with right-click. The shield is permanently equipped and the player can choose to also equip either of the other two weapons when selecting a mission, provided they’ve bought that module. In-game you can switch between the shield and weapon by turning the mouse wheel. To visualize which one is selected, the weapon extends out of the claw while active.

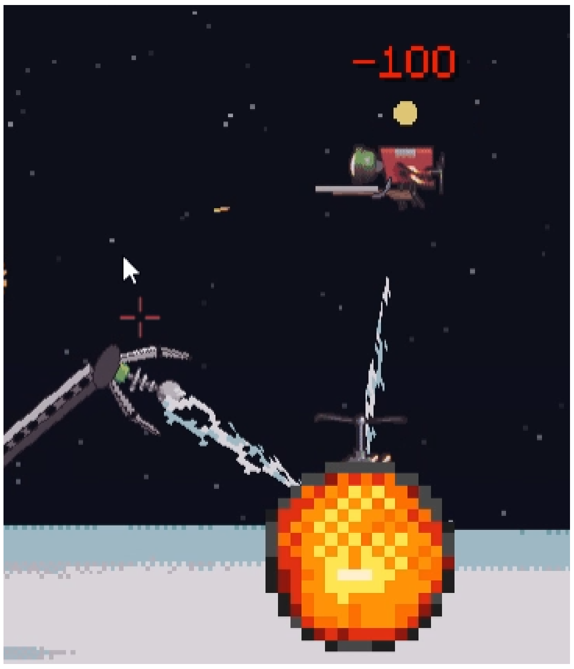
- Shield: Absorbs small projectiles while active. When activated, briefly emits a charge which will reflect larger projectiles once we implement those.

- Flamethrower: Flammenwerfer, it werfs Flammen. Flames deal immediate damage to intersecting enemies, and enemies which have been engulfed in particles for at least 0.5 seconds will continue burning for 3 seconds and take 20 damage per second. Firing this costs engine fuel.

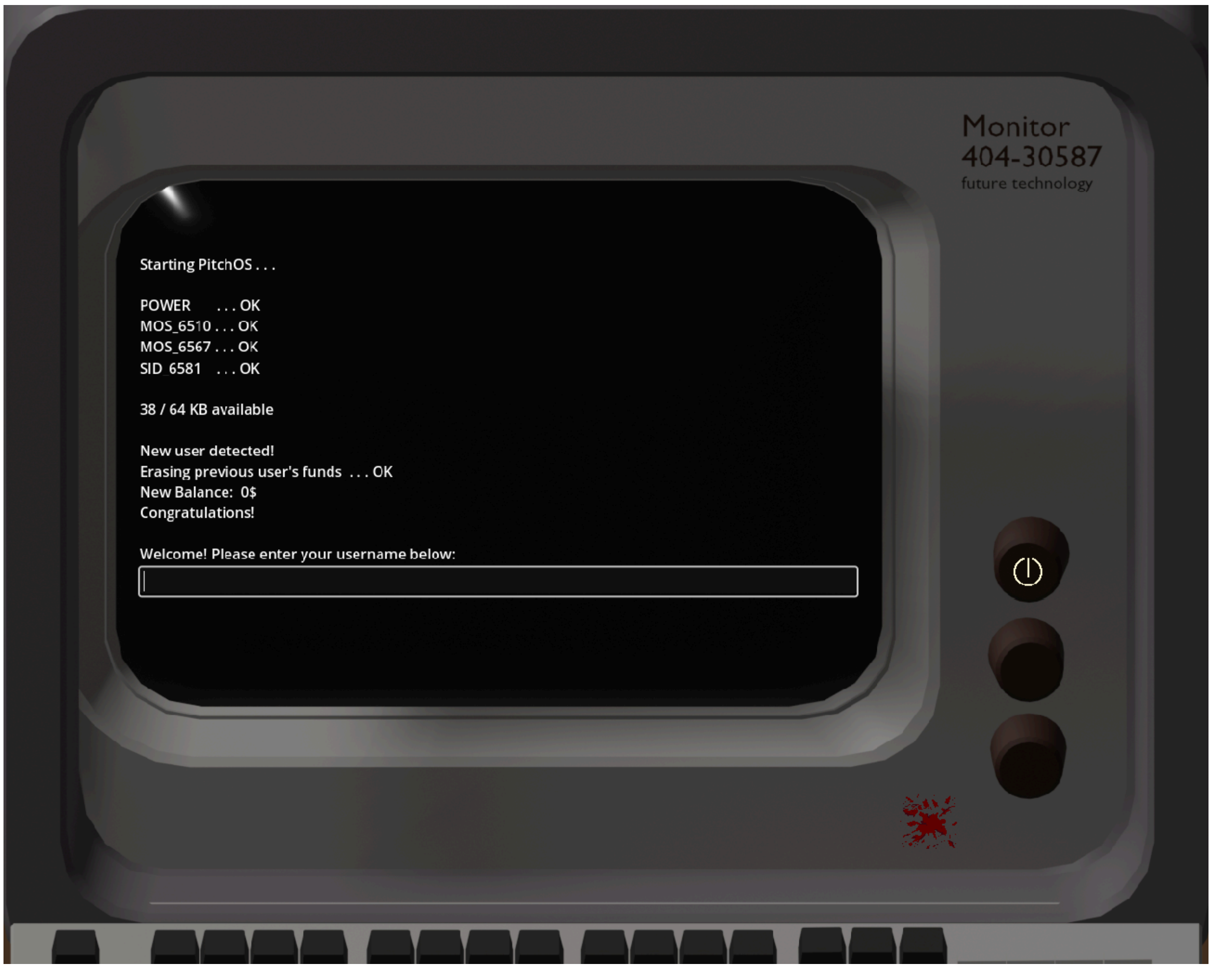


- Arclight projector: A tesla coil which can be charged up for up to 2 seconds by holding right-click. On release, a bolt will jump to the nearest enemy within range and recursively jump to others as long as it has enough charge left, dealing damage to each enemy along the way. Using this weapon consumes ammo.



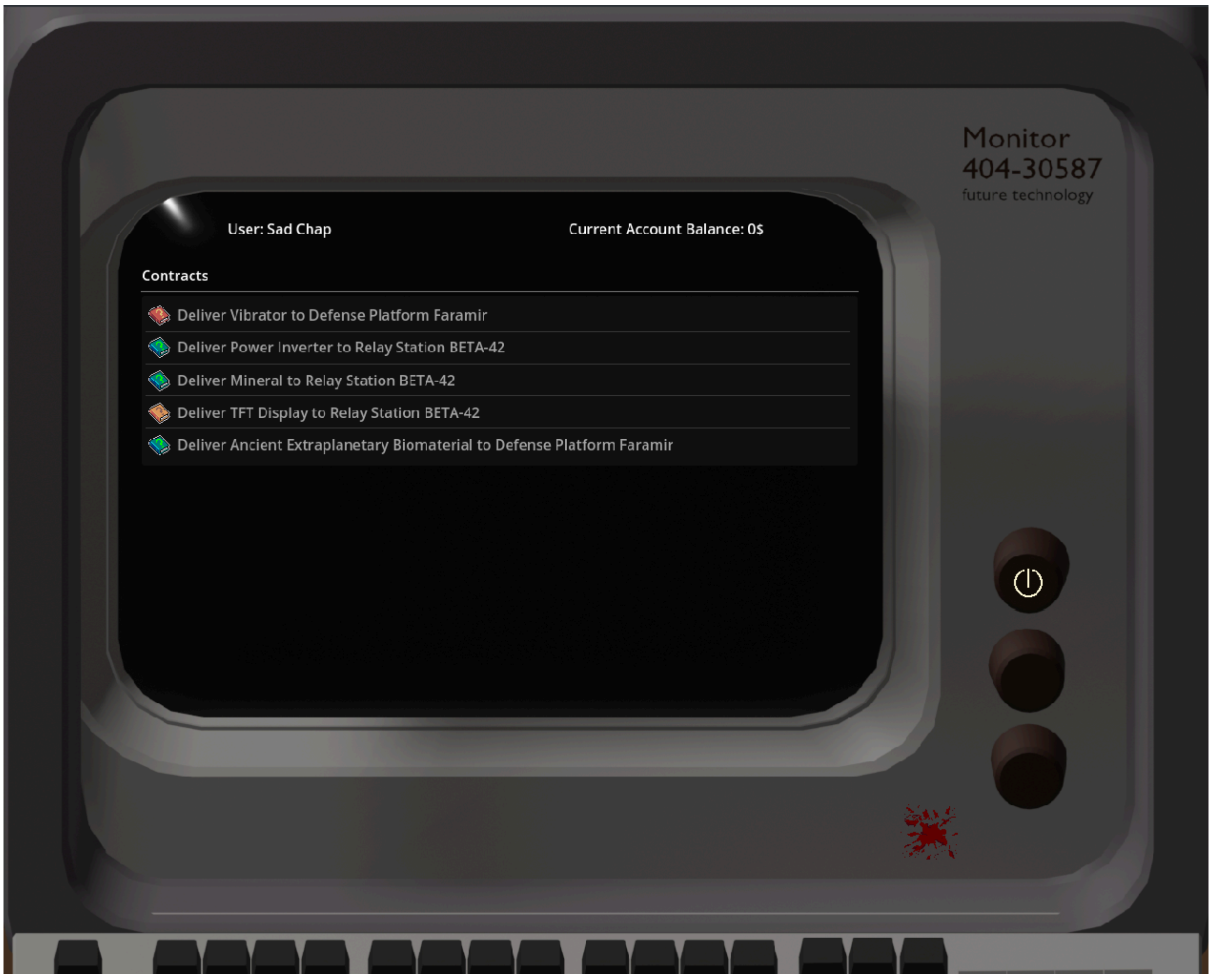
<u>Contracts</u>
The game starts with a view of a terminal on which the player can select their contract. The contracts are randomly generated by combining a scenario, cargo type and risk level, each coming with their own bits of text and variant which are assembled to form all the parts of a contract, including the description, the destination, risk of enemy encounter and their size and of course how well the contract pays. Right now the destination does not have an effect because all three different destinations will spawn the same test-level we have been using all this time, but once the other level scenes are ready each destination will start the elevator ascension on a different cable with different enemy variants, weather conditions and, well, destinations.

When the terminal is first booted up or after a gameover, a user name has to be entered:



```
Starting PitchOS . . .

POWER     . . . OK
MOS_6510 . . . OK
MOS_6567 . . . OK
SID 6581   . . . OK

38 / 64 KB available

New user detected!
Erasing previous user's funds  . . . OK
New Balance:  0$
Congratulations!

Welcome! Please enter your username below:
```

The terminal first shows a list of 5 these contracts and selecting them leads to a detailed view with the options to go back or accept the contract. Accepting it will then lead to the equipment menu where the player chooses their elevator loadout. Right now this also functions as a small shop for the arm modules.



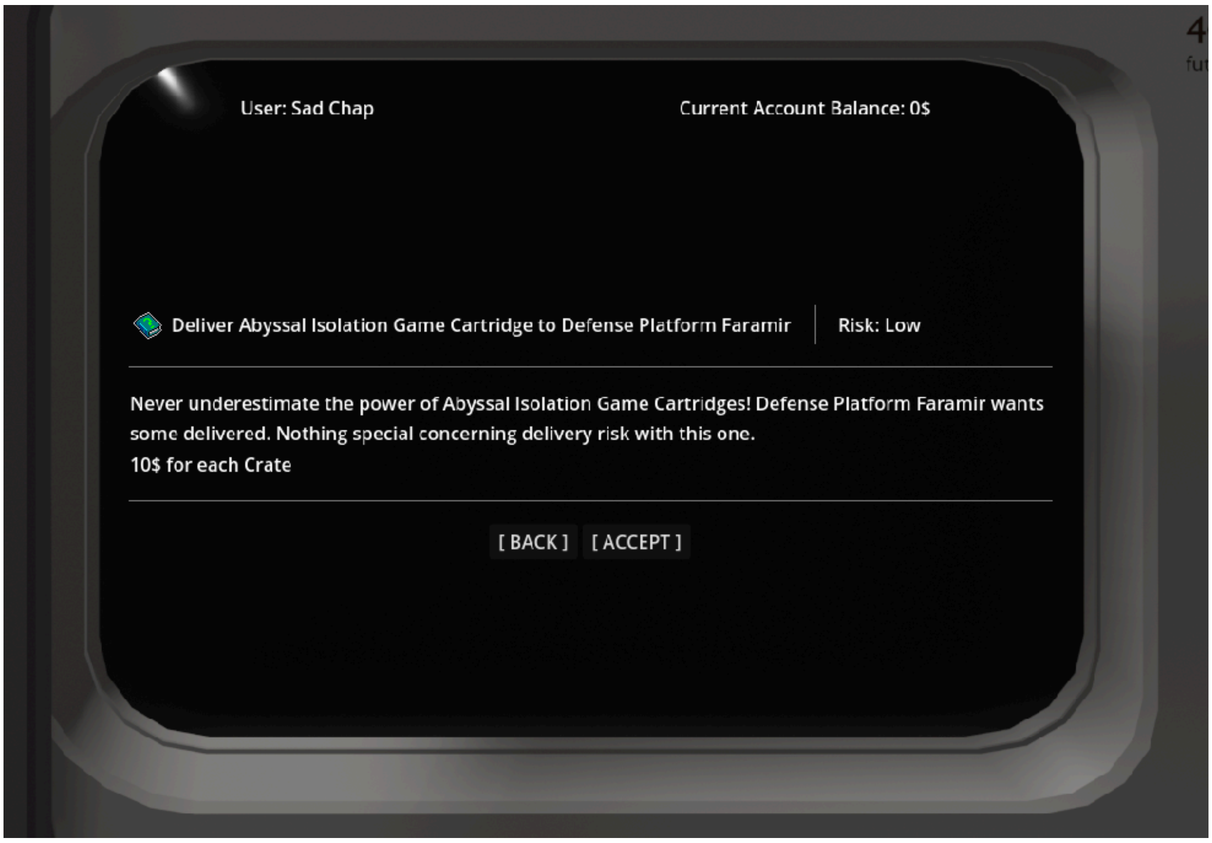
Monitor
404-30587
future technology

User: Sad Chap

Current Account Balance: 0$

Contracts

- Deliver Vibrator to Defense Platform Faramir
- Deliver Power Inverter to Relay Station BETA-42
- Deliver Mineral to Relay Station BETA-42
- Deliver TFT Display to Relay Station BETA-42
- Deliver Ancient Extraplanetary Biomaterial to Defense Platform Faramir



User: Sad Chap

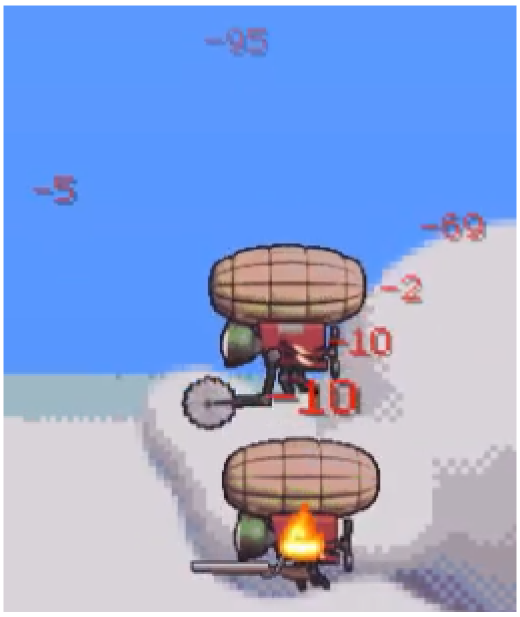
Current Account Balance: 0$

Deliver Abyssal Isolation Game Cartridge to Defense Platform Faramir | Risk: Low

Never underestimate the power of Abyssal Isolation Game Cartridges! Defense Platform Faramir wants some delivered. Nothing special concerning delivery risk with this one.
10$ for each Crate

Enemies

While we didn't get around to adding new enemies, we have done a complete overhaul of the enemy script, making it more generic and modular, which will allow us to quickly add more variants with different weapons and behaviors. The enemies are now based on a truly generic enemy base class from which all specific enemy types inherit. The code grew a bit convoluted before so this was necessary, although a bit time consuming, but still very much worth it. We have also added damage indicators to show the player how effective their attacks are.

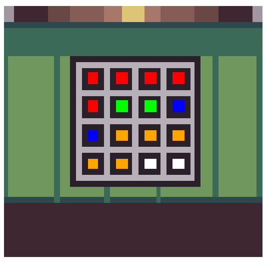


Music and options menu

Besides various sound effects, we also added music in the hangar scene as well as an options menu where the volume of the music, the effects and the master volume can be changed separately.

Inventory

We wanted to show the player in a diegetic manner what is currently available in the inventory inside of the elevator. To achieve this, a small matrix representation was implemented:



Red represents fuel, green is ammo, blue is scrap and orange is loaded cargo. These colors are also used in the beginning of the game when the player selects what they want to load into the elevator.