Final Release

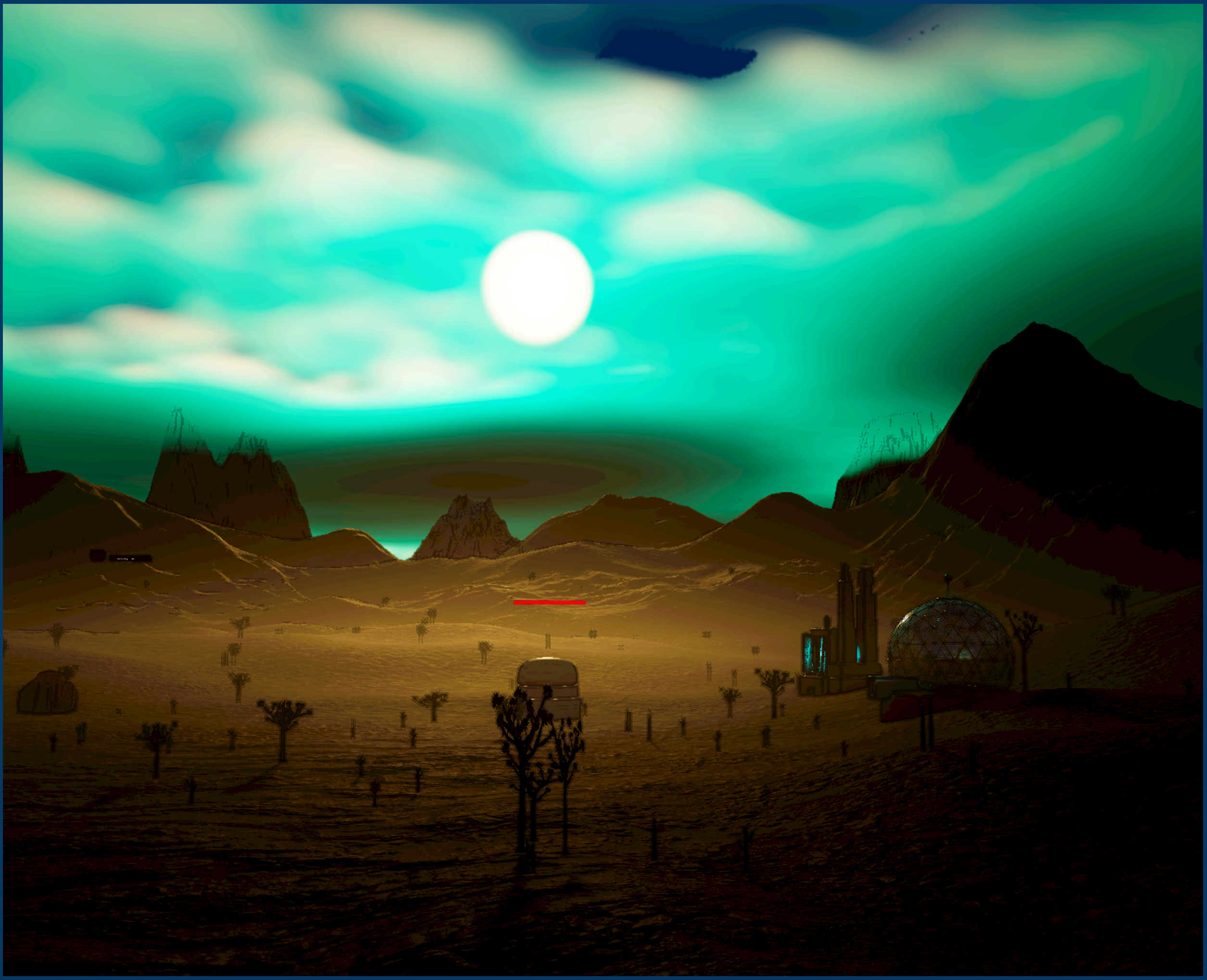
Doomsday: Underground Uprise

Technische Universität München

Winter Semester 2023/2024

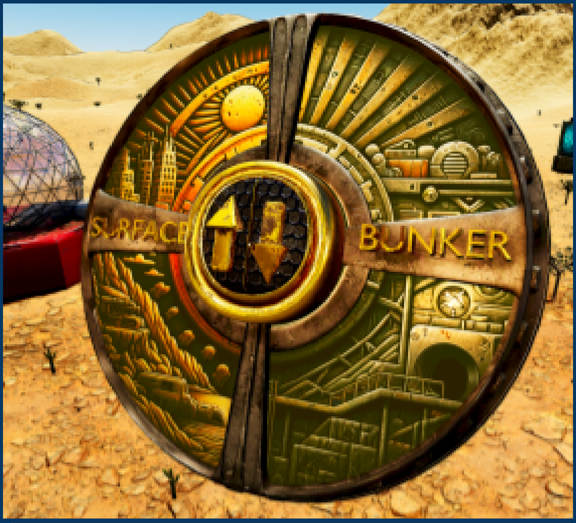
Games Laboratory

Matija Jajcinovic Hongbo Chen Haorui Tan



Up-Down Theme

Our idea of having a separated Underground/Bunker Basement (Down part) Building and a surface combat and RTS part (Up) is quite unique and we cannot think of a game that did something similar.
In our opinion the limiting theme helped us to come up with a game idea; otherwise the sheer amount of possibilities that all seemed fun would have been overwhelming.



Switch Button to switch between the up and down world

Final Result vs Initial Vision

Graphics and Animation

Our vision was to make a game that sticks out in graphics and has unique assets to avoid looking like an Asset Flip.
There are many subtle details like 4k texture with easter eggs, rotating fans in the bunker, blinking tower placement beacons etc.

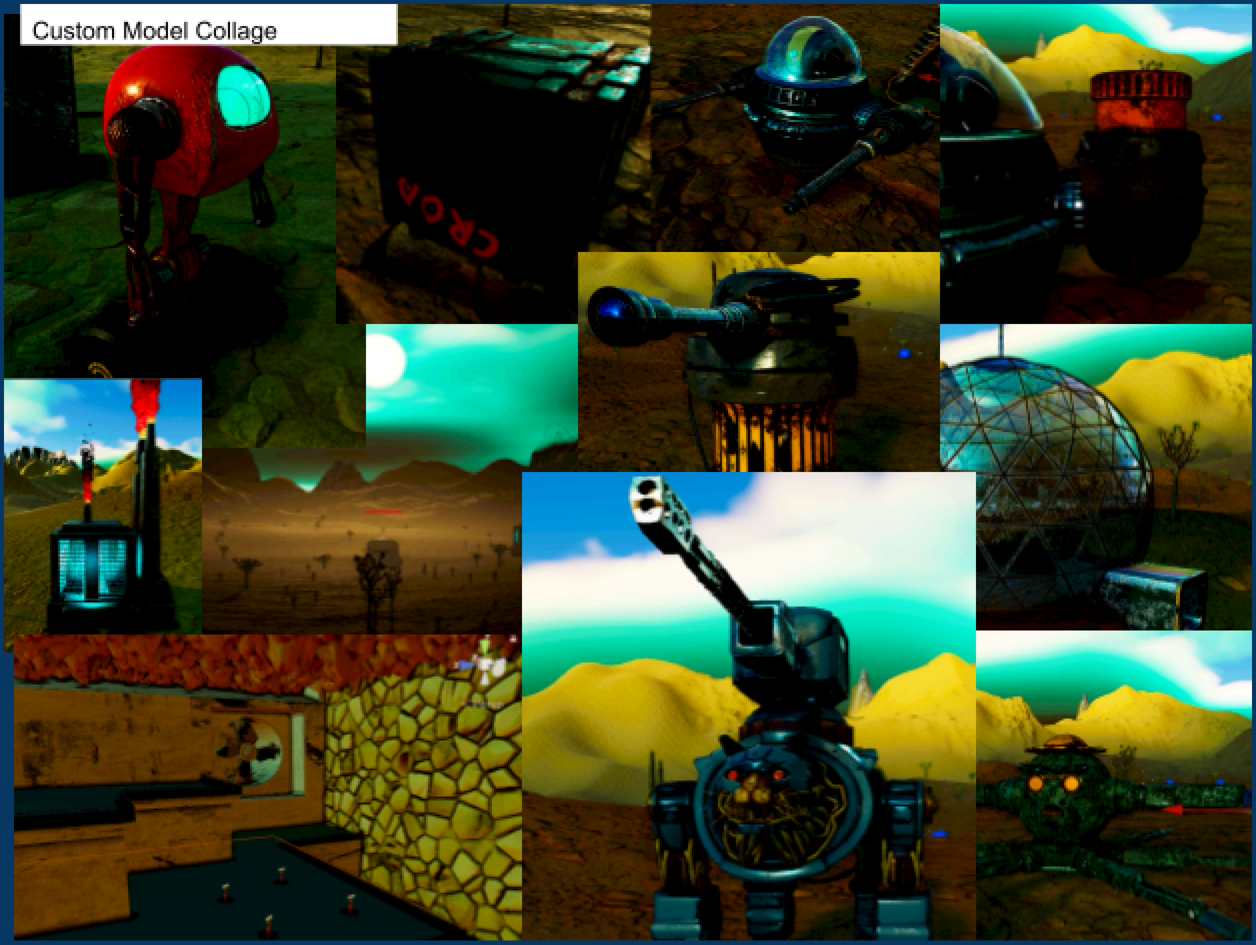
Our Robots are not just animated meshes; they are virtual robots using classical Robotics and Machine Learning, fully interacting with the Physics Engine which all in all achieves our desired effect.
This is quite unique because RTS games usually opt for more performance to utilitate more units; we chose the opposite way: less

performance/less Units for detailed ones.

Most of our meshes are self-made and highly detailed; self designed, modeled and textured; it requires a high amount of modeling experience and took us a lot of time.
Compared to the initial game art direction, we drifted from an isometric, cartoony looking style more into a realistic 3D one with still some playful accents (custom edge shader, super saturated colors, unfunctional robot design). The reason for this was that we wanted to fully benefit from the HDRP's capabilities for realistic rendering.

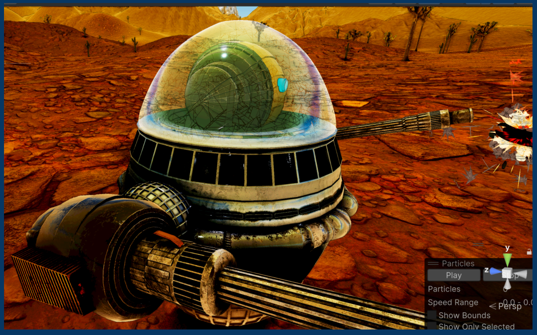
Utilizing the new render capabilities of Game Engines is not an easy task bc PBR materials are needed (and the know-how + programs to generate them) and needs more finetuning. Also the documentation is still lacking compared to the standard render pipeline.

Left our model in-game vs initial directional Game Art



RTS-Part

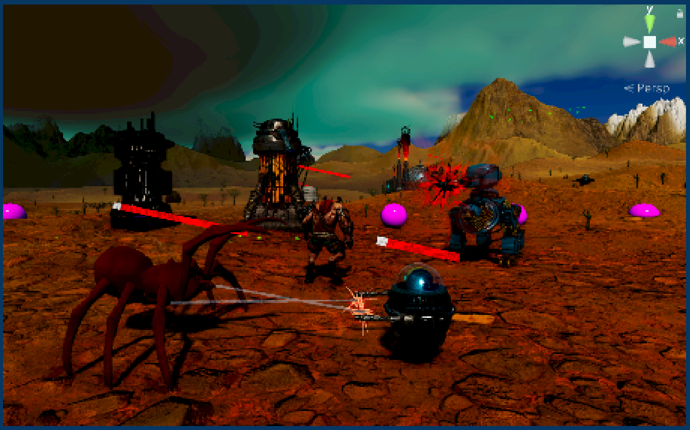
Regarding the RTS part, players can move the robot closer to or away from the enemy while the robot will automatically lock on targets; form a defense around their home; and go to the healing tower to restore health. Greatly increases the playability of the game. After all the objects are generated, the player will not just watch, but has no tactically move the robots to increase their effectiveness.



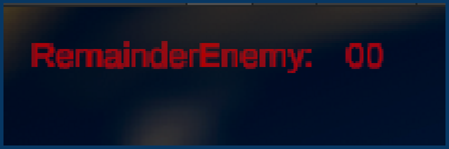
Tower Defense Part

We have added four defense towers (two attack, two auxiliary) and two farm towers. Each tower has its own function, and players can choose to build it themselves.

A wave of enemies will be generated every 120s. There are four types of enemies, with different attributes. Monsters will attack everything around them. When they attack the base and base’s health reaches zero, the game ends.



Wave System:

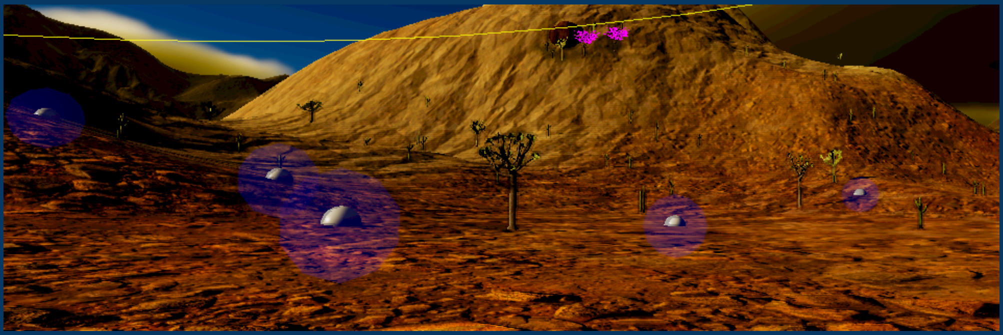


Game starts -> 2min countdown to the first wave:
Quantity on each side: (same quantity on left and right)
Spider: 1 Zombie: 2 Barbarian: 1 Beholder: 2

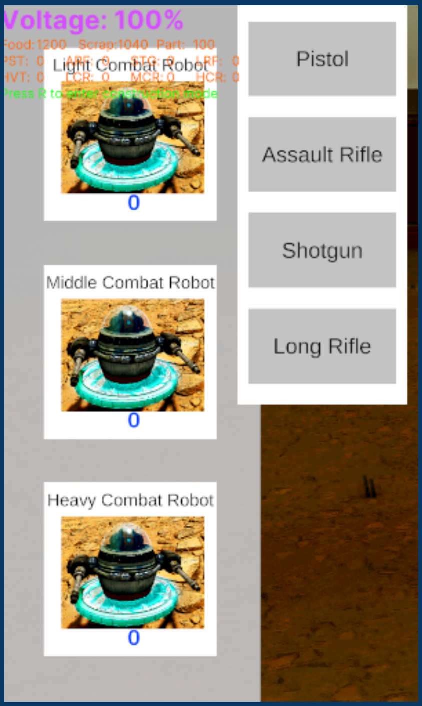
->After all monsters are killed->Countdown 2 minutes for the second wave:
Quantity on each side: (same quantity on left and right)
Spider: 2 Zombie: 3 Barbarian: 2 Beholder: 3

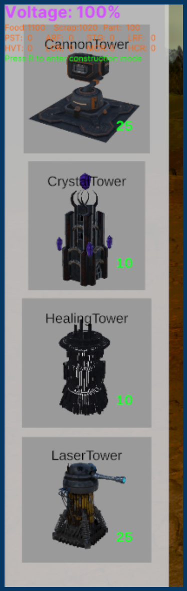
->After all monsters are killed->Countdown 2 minutes for the third wave:
Quantity on each side: (same quantity on left and right)
Spider: 3 Zombie: 4 Barbarian: 3 Beholder: 4

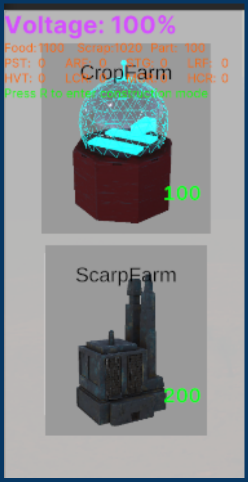
(Infinite loop, the number of each monster is +1 each time)



-Surface construction table







Possible Extensions (not part of timeline)

There are many things missing that one would expect from a finished game.

- a save system to save the state of the game for later play
- more “feel”, like camera shake when flying near units that are shooting, having hit effect on units
- sound effects are mainly missing
- more polished UI design
- more in depth tutorial (non skip-able learn by play one)

Feature added Based on Demo Day and Play Testing Critics

- more scrap per crate
- added glow to units to see them more distinctively in the dark (clouds cause darker spots)
- more visual Bunker UI
- more distinct colors

Timeline

We had a very packed timeline. The start was slow because due to past experiences we decided to make the framework (Mlagents, base building) stable and then start adding content.
But after that we accelerated and implemented most of the features.

In retrospect, we should have packed it somewhat less with content (maybe less units, models) to have a bigger buffer for technical problems (Git rep was full, bugs etc). But we still stress how much work all this was that we managed to implement.

What we could not finish

- sounds (like explosion sounds, enemies dying)
- several guns (due to illness, but a part was already done)
- several robot units (due to illness, but a bigger chunk was already done)
- more environment details to make it look more like a post apocalypse game (time)
- custom models for the people (time)

What did not made it into the game

Due to the illness of one of our team mates, several features did not get into the final release.

Four-legged Heavy Robot Tiger

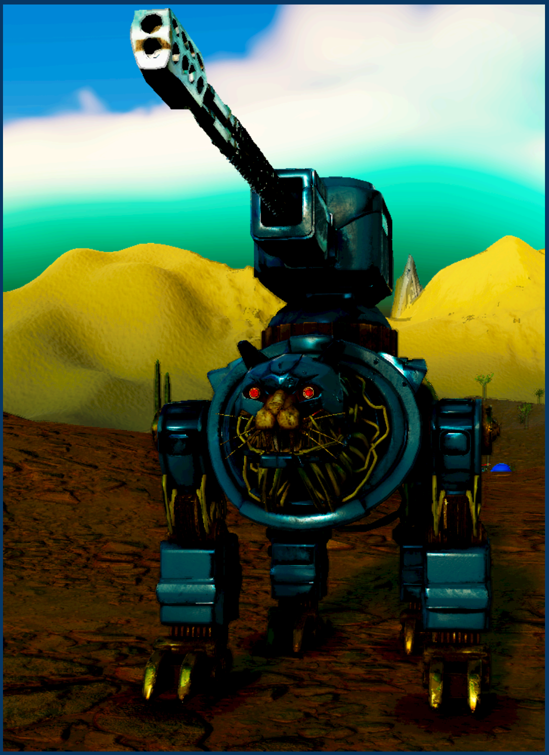
The idea was to have a quadruped robot with a tail for counter balance; to showcase the MLagents capabilities.
It went fine, after several unsuccessful tries (bugs with physics) it learned to walk (Train Time 50 hours).
The idea was then to make the mounted turret work with the gun script (not implemented due to git errors and illness).

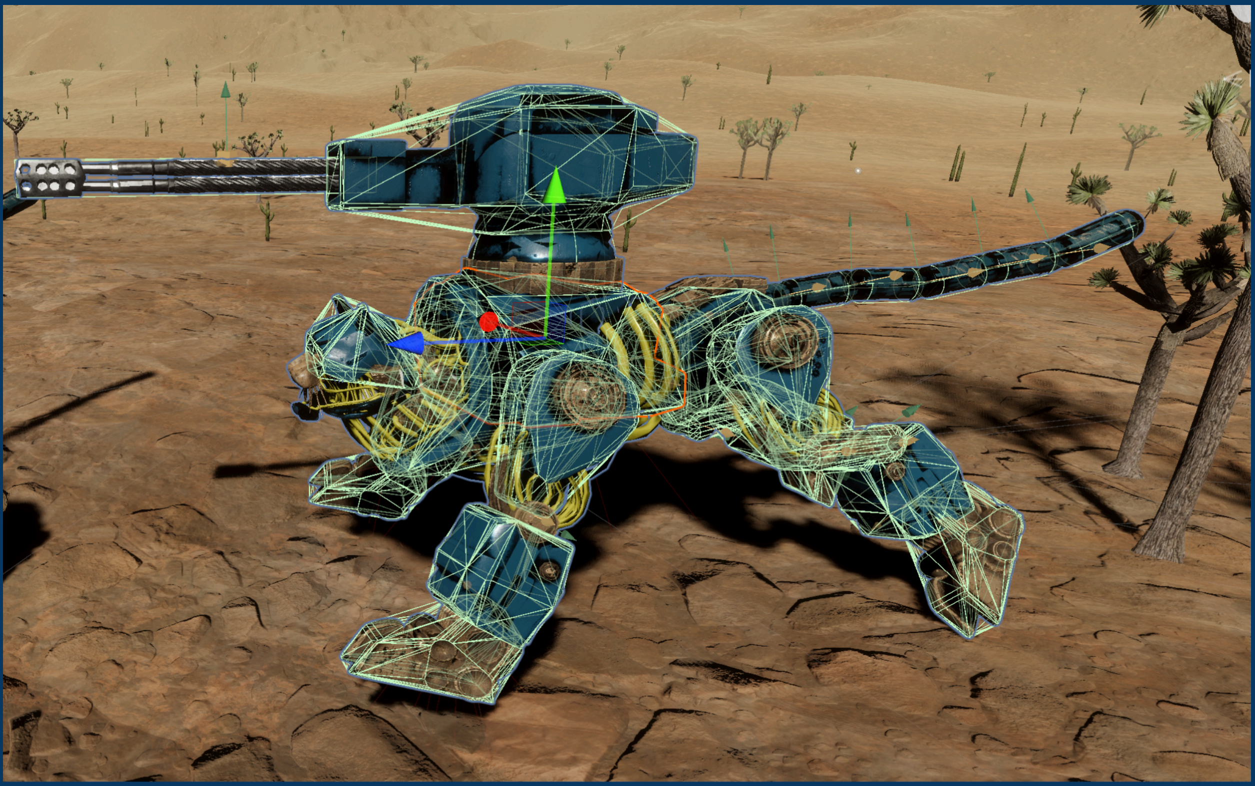
The walking pattern is very careful and (too) slow to maximize the long run walking reward (falling thus stopping early brings of course less points). It resembles a tortoise; For further reference I think a shorter episode time would motivate it to go faster.

A cool feature is that it learned to use the tale to counter balance as intended.

Still it looks kinda cool and relatable for a 10 meters tall monster robot.

It did not get into the final game but makes for some cool game art. It lacked the whole game logic (health, targeting, controlling etc.).





Sniper Gun

The idea was to shoot a self-guided rocket (again PID controller) to have an interesting spin on a sniper gun with high range and accuracy, but low damage per second.
Could not finish it due to illness (but very simple logic, just shoot every x seconds a rocket in the air if an enemy is in range).

Melee Unit Model

The idea was to use this to train POCA melee 3 vs 3. To be honest I think this was too ambitious anyway. But I have modeled the according 3D model.

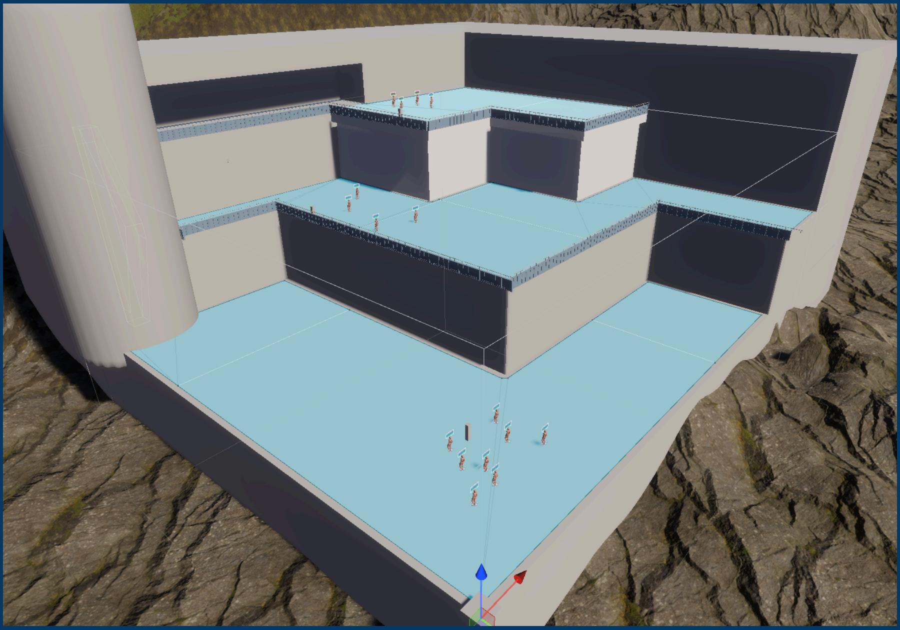
Underground Game Playa

What I have implemented:

1. **Underground citizens' navigation system with elevator logic:**

Completeness: **layer3:** completed all the needed logic and the people in the underground basement can navigate to different layers correctly.

Collision mesh(blue layers) used for nav mesh:



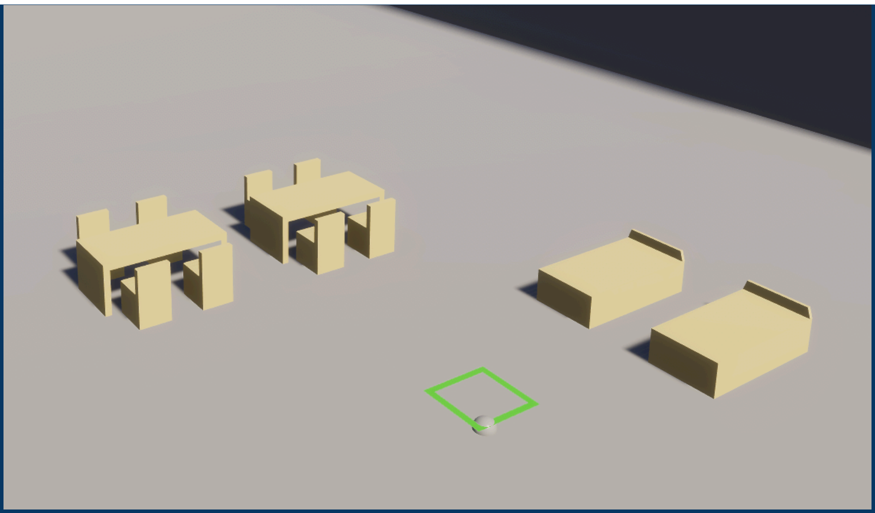
elevator effects:

2. underground basement building system:

Completeness: **Layer 3:** The underground basement building system has been completed, providing the fundamental building functions.

Further works: However, it still requires the implementation of border constraint and collision detection logic to ensure proper placement and prevent overlapping of buildings. Furthermore, future iterations of the system will require an expanded list of building options, which will involve collaboration with the team responsible for building modeling.

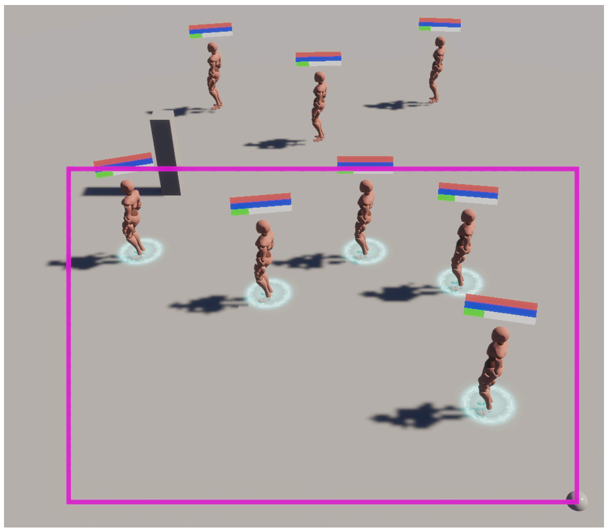
Construction System:



3. People's selection box and deployment logic:

Completeness: Layer 3: The current method of detecting the selection box, which utilizes a mesh trigger established by the vertices from the camera position to the vertices hit by the raycast function in Unity. Improved the robustness when the ground terrain is not a simple plane. The logic for establishing the trigger mesh has been enhanced to accommodate complex terrains and ensure accurate selection box detection.

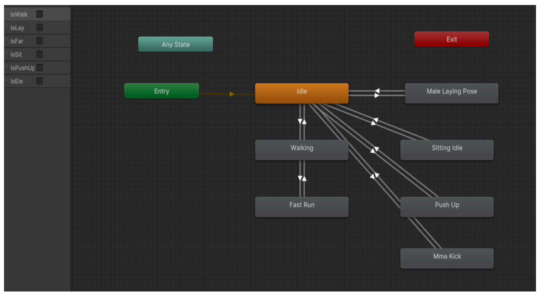
people’s selection box:

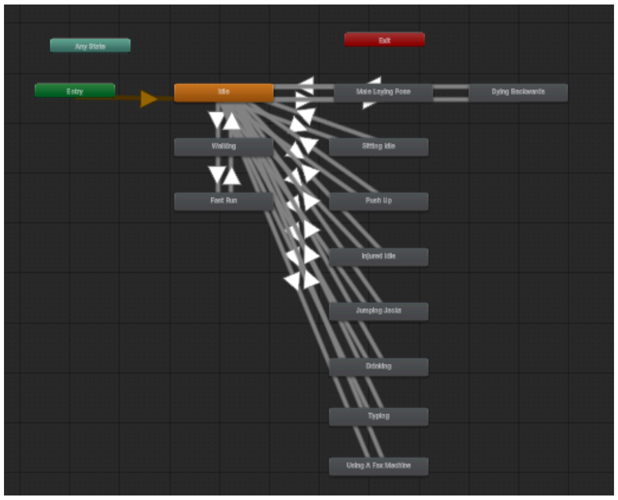


4. Gathering animations of people and achieving animation control logic:

Completeness: **Layer 3:** The fundamental logic for animations controls of people has been completed. Further variances have been made which includes expanding the available animations and refining the animation control logic to provide a wider range of movements and behaviors for the underground citizens.

animation control logic:

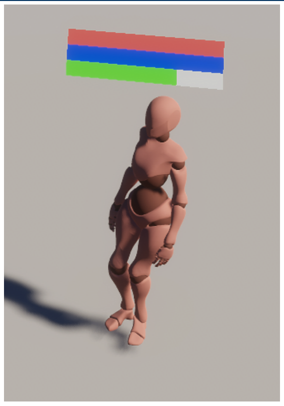




5. People's status bar:

Completeness: **Layer 3:** The current implementation of the people's status bar can display all the necessary information regarding an underground citizen's status.

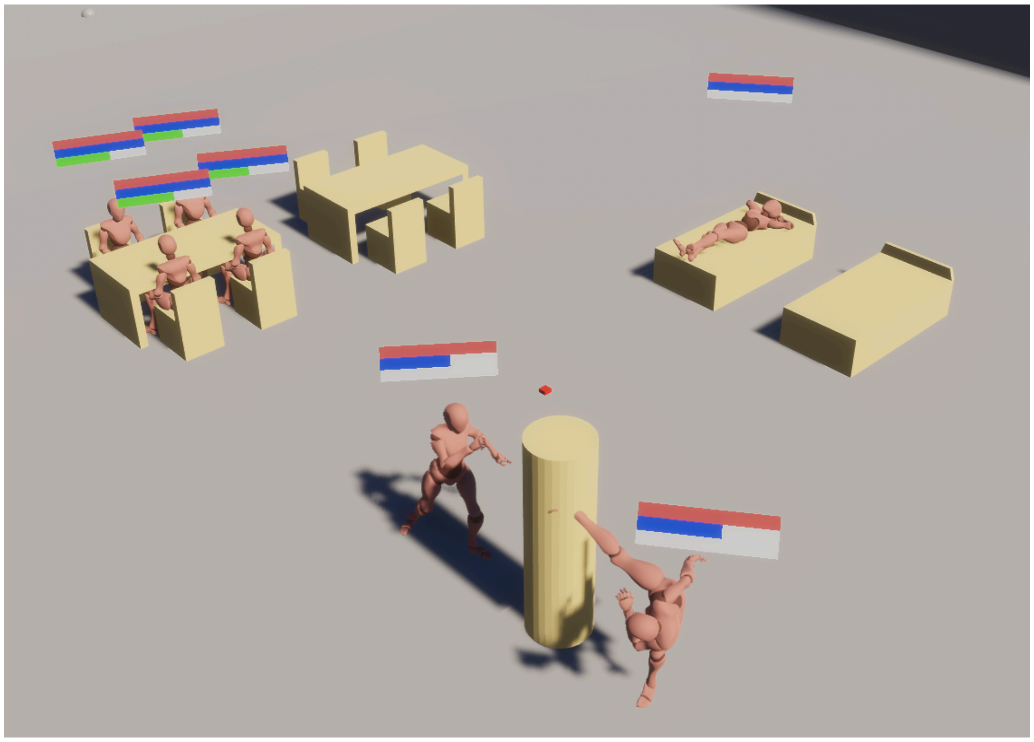
people’s status bar, red bar denotes health value; blue bar denotes energy value which has a relation to the work efficiency and green bar denotes hunger value which measures how much time remains until the people need to eat::



6. Interaction logic between people and buildings:

Completeness: **Layer 3:** All important logics related to the interaction between people and buildings have been finished. This includes animation switching, people finding correct seats within the buildings, effects generated by people in relation to the buildings, and effects generated by the buildings in relation to the people. These interactions create a dynamic and immersive environment for the underground citizens.

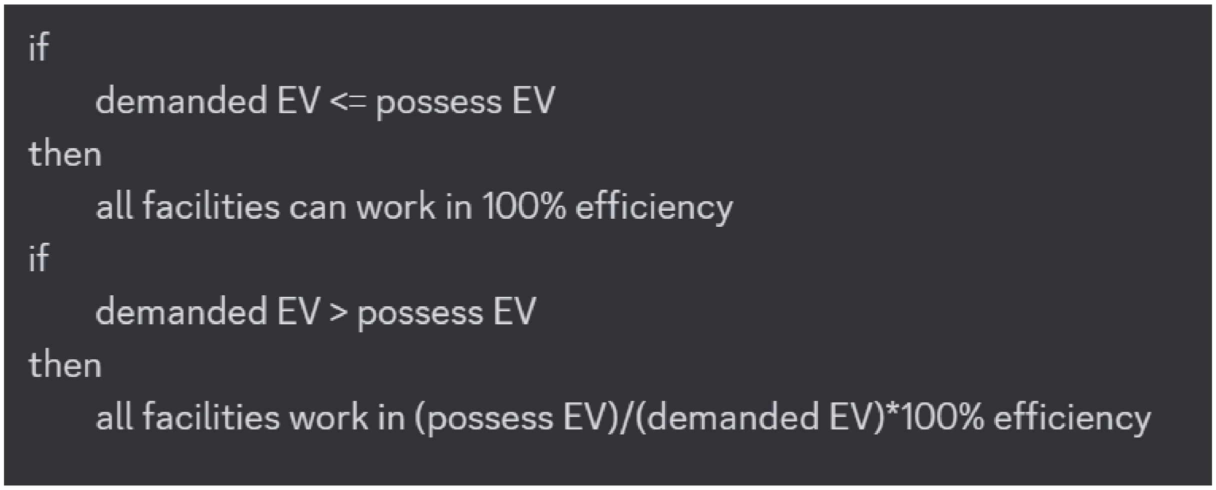
People can locate to the right position and do the right movement and buildings can bring influence to the people's status value:



7. Electricity voltage system:

Completeness: **Layer 3:** The important logics for the electricity voltage system have been completed. This includes handling the effects caused by the voltage system and integrating them with the surface combat. The electricity voltage system adds another layer of complexity to the underground environment, impacting various aspects of the simulation.

Electricity Voltage Logic:



```
if
    demanded EV <= possess EV
then
    all facilities can work in 100% efficiency
if
    demanded EV > possess EV
then
    all facilities work in (possess EV)/(demanded EV)*100% efficiency
```

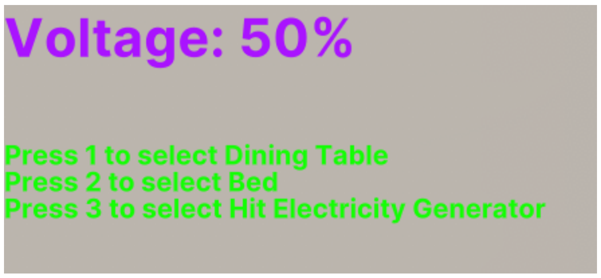
Assume EF denotes efficiency, CE denotes consumed electricity per building, GE denotes generated electricity per building:

$$EF = \left(\frac{\sum CE}{\sum GE}\right)^2$$

8. Demo display of voltage and character status and construction menu:

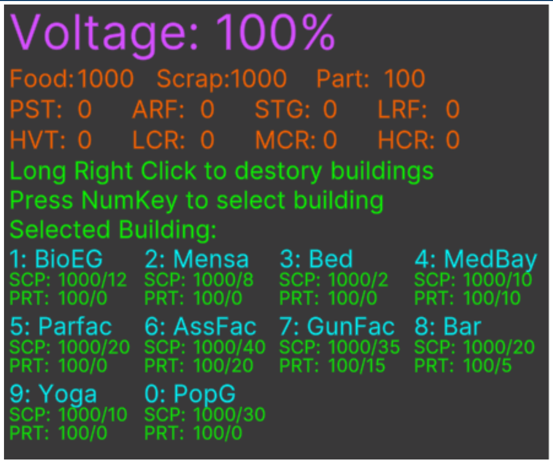
Completeness: **Layer 3:** The demo display, which showcases the voltage system, character status, and construction menu.

demo display of voltage and construction menu:



```
Voltage: 50%

Press 1 to select Dining Table
Press 2 to select Bed
Press 3 to select Hit Electricity Generator
```

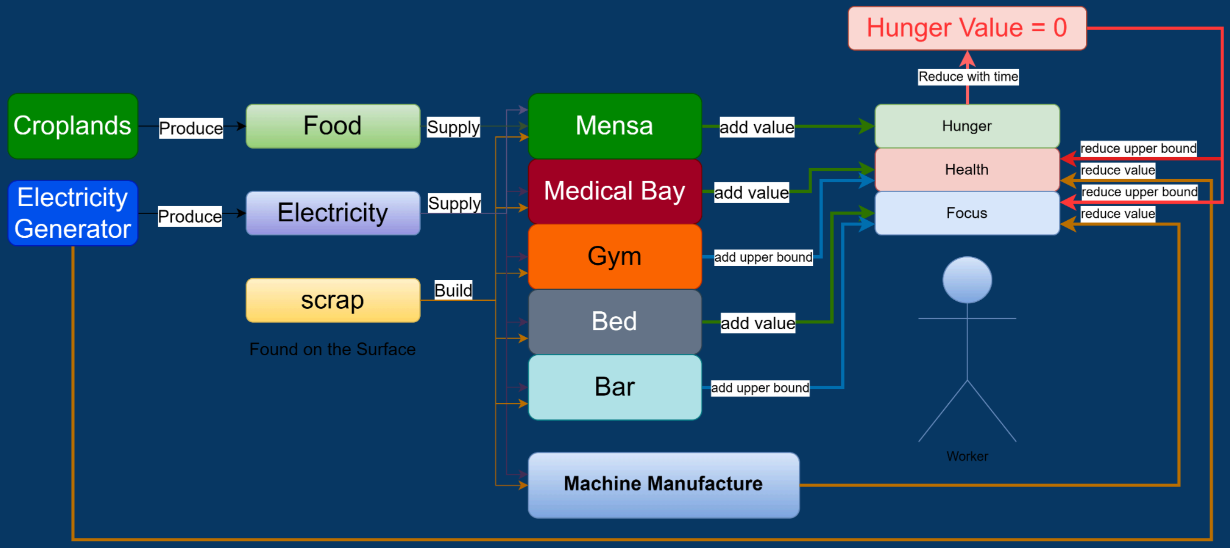


```
Voltage: 100%
Food:1000   Scrap:1000   Part:  100
PST:  0     ARF:  0      STG:  0      LRF:  0
HVT:  0     LCR:  0      MCR:  0      HCR:  0
Long Right Click to destory buildings
Press NumKey to select building
Selected Building:
1: BioEG         2: Mensa         3: Bed           4: MedBay
SCP: 1000/12     SCP: 1000/8      SCP: 1000/2      SCP: 1000/10
PRT:  100/0      PRT:  100/0      PRT:  100/0      PRT:  100/10
5: Parfac        6: AssFac        7: GunFac        8: Bar
SCP: 1000/20     SCP: 1000/40     SCP: 1000/35     SCP: 1000/20
PRT:  100/0      PRT:  100/20     PRT:  100/15     PRT:  100/5
9: Yoga          0: PopG
SCP: 1000/10     SCP: 1000/30
PRT:  100/0      PRT:  100/0
```

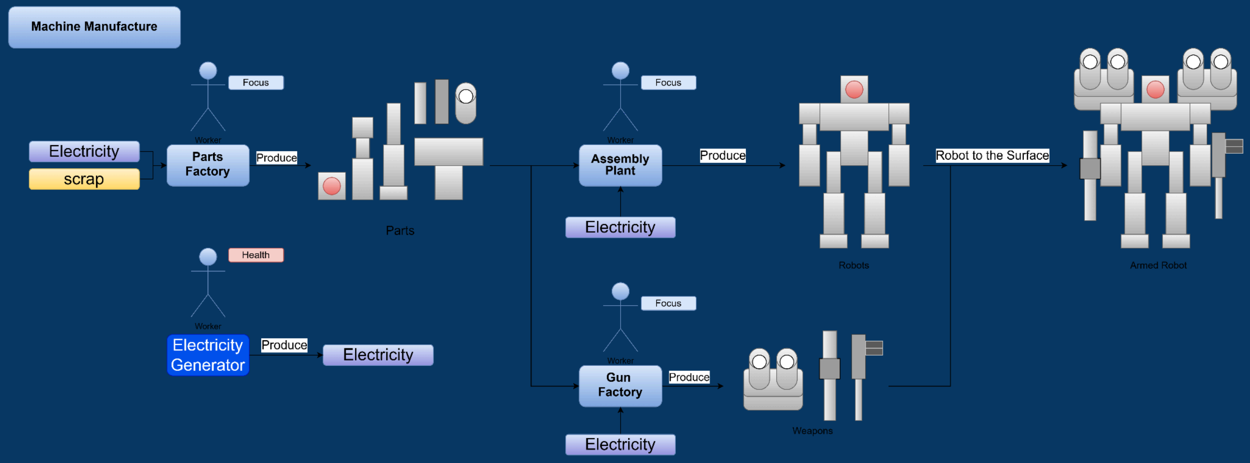
let the player to customize the underground basement that supports the people survive from the doomsday by using RTS construct system to building the underground basement and commanding underground citizens to utilize limited resources to build survive facilities and defense equipments to win the game

- Left click and drag to select people then right click the buildings to deploy them
- Right click the factory when no people are selected to select the equipment to produce at this factory
- Right click the Electricity Generator to release the people who work in the Electricity Generator

Buildings effects and People's status



Equipments produce process

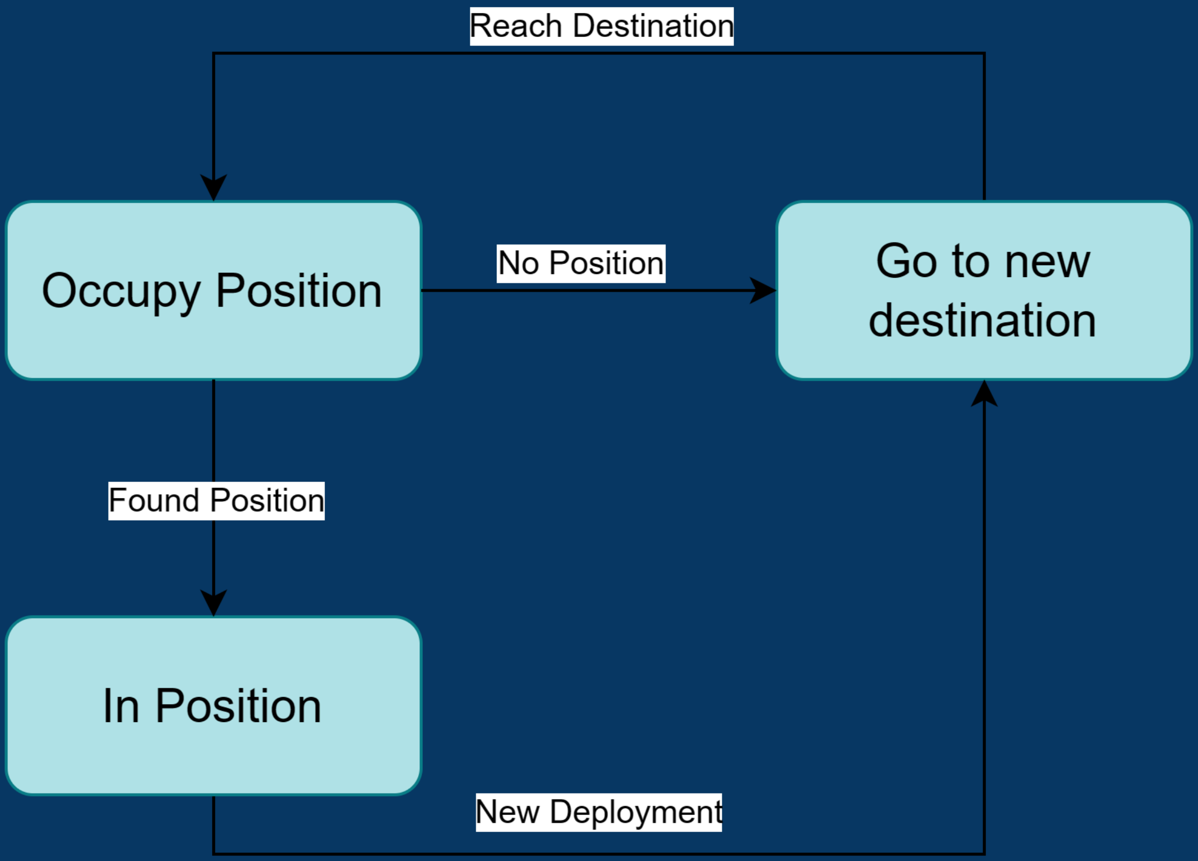


Highlights

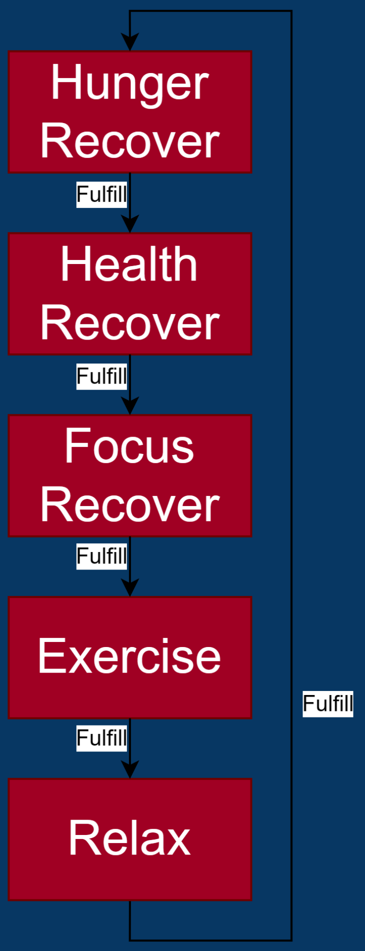
1. Simulating the authentic physiological state of people living in a restricted space which means they will feel hungry and feel tired when keep working and will suffer when don't get enough supply
2. smart deployment logic for player to make the game easier to operate

    a. Players don’t have to sent the task of the people one by one by hand rather just select a group of people then right click the task building, then the selected people will automatically distribute themselves to all the same kind of task buildings
    b. automatically search the same kind of empty buildings when the employed building is fully occupied
    c. automatically replace the tired people when sent to workplace

3. automatic self-supply logic of people
    a. When there is no work for the people to do, then the people will automatically follow the self-recover logic by themselves and don’t need to wait for the player to deploy them. In this method, the complexity brought by the people’s physiological simulation will be significantly relieved
    b. achieve this by using looping status to drive the people to go to the destination



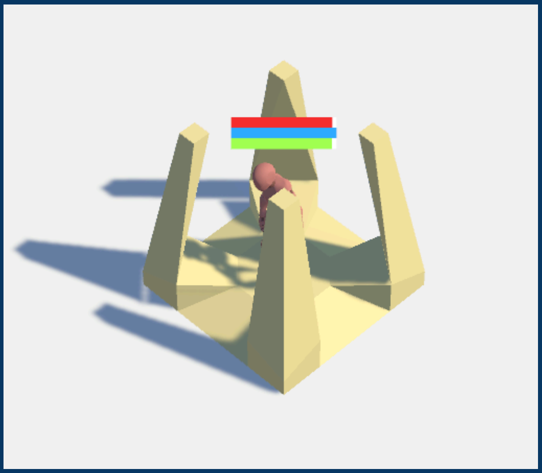
c. using ring queue data structure to store the instructions of self-resupply



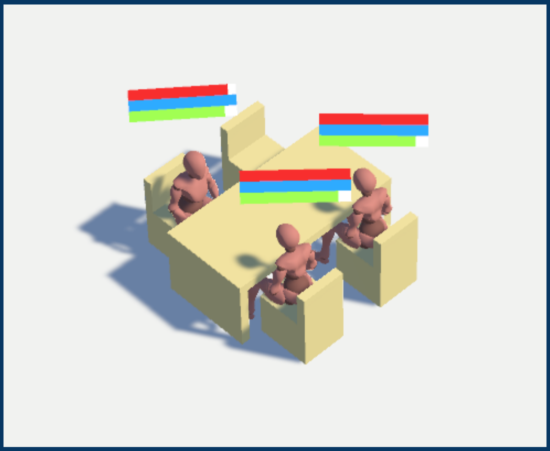
Underground Basement Facilities Modeling and interaction

Implemented the 3D models of the underground basement facilities and the relevant animation controller used for interaction between people and facilities

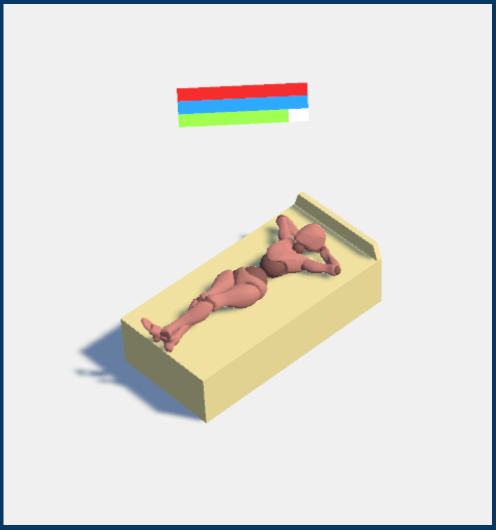
1. Electricity Generator



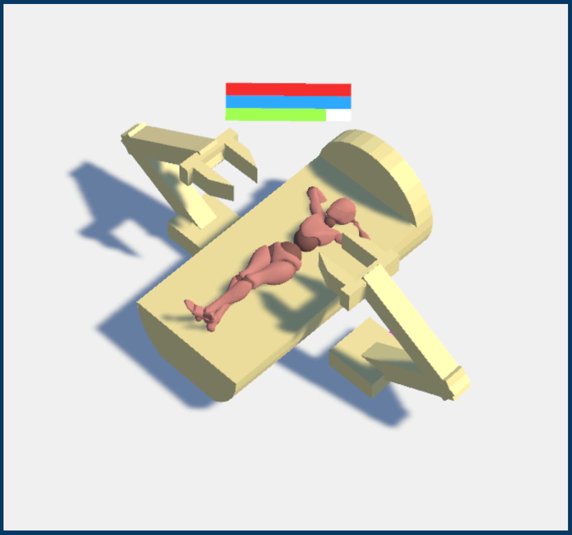
2. Mensa



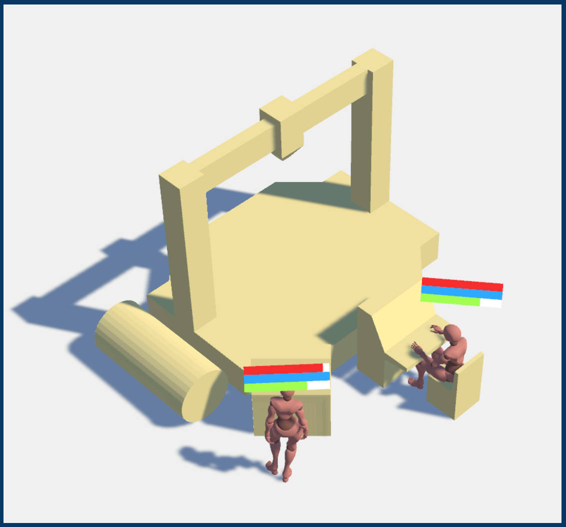
3. Bed

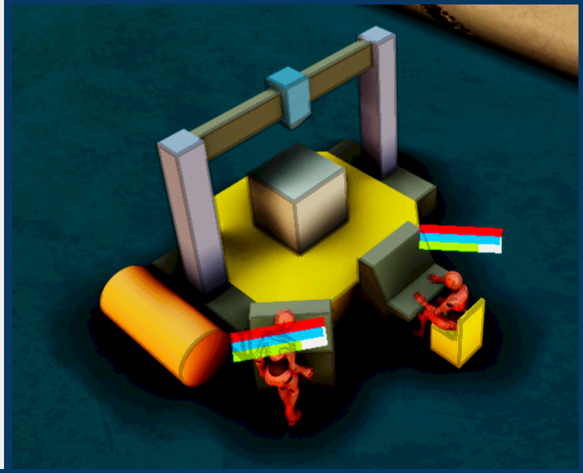


4. Medical Bay

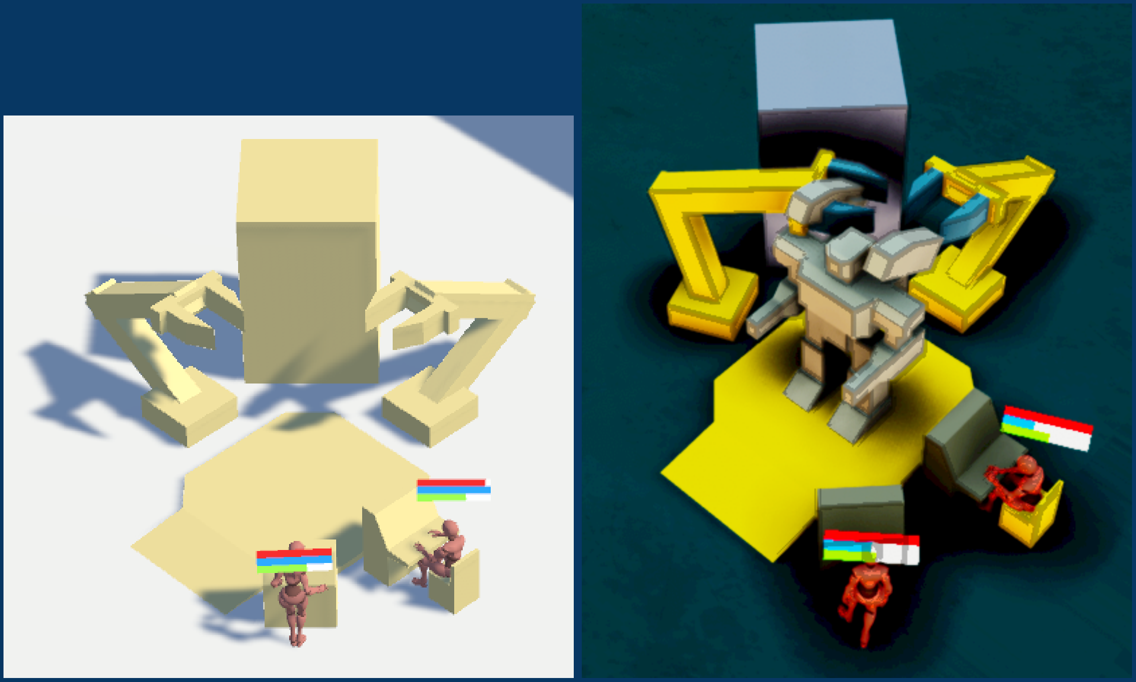


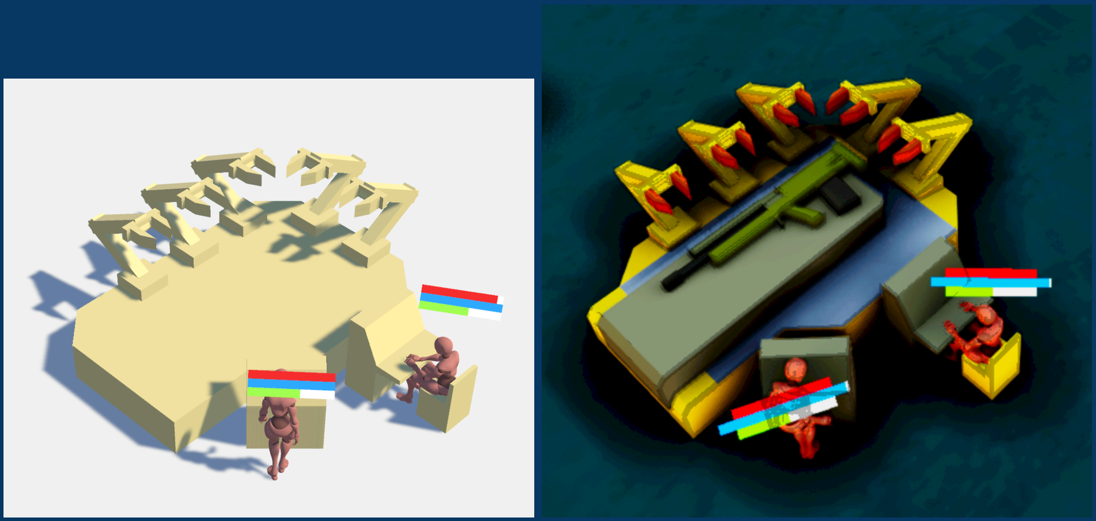
5. Part Factory

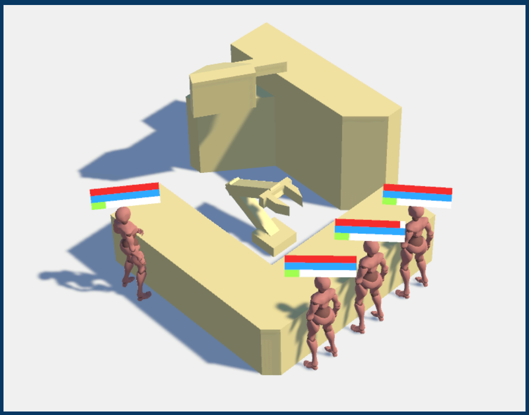




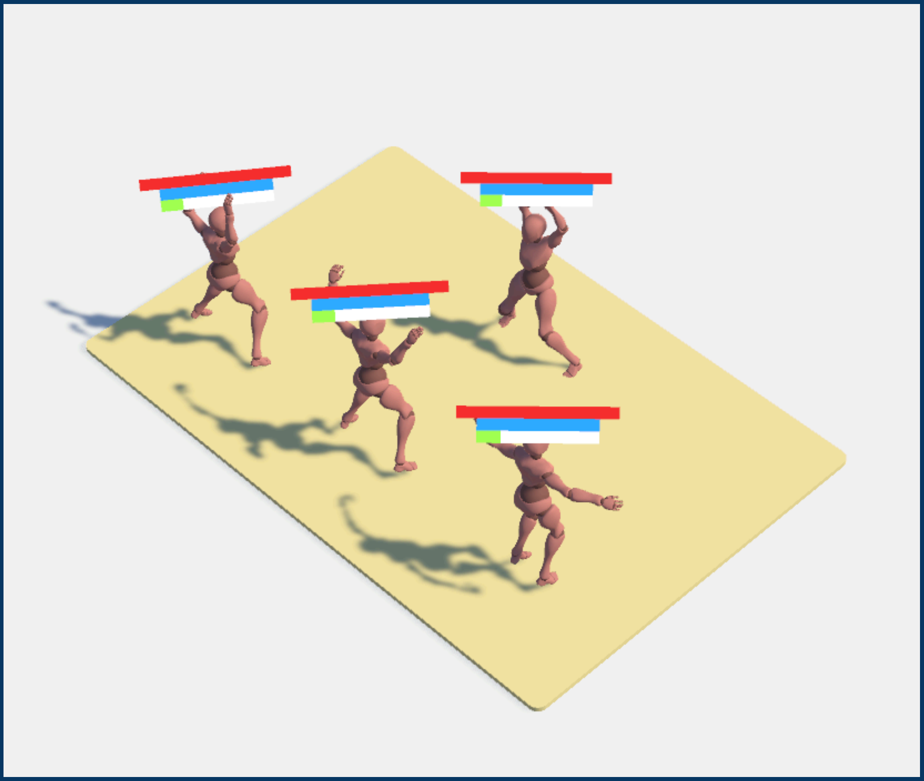
6. Assembly Factory


7. Gun Factory

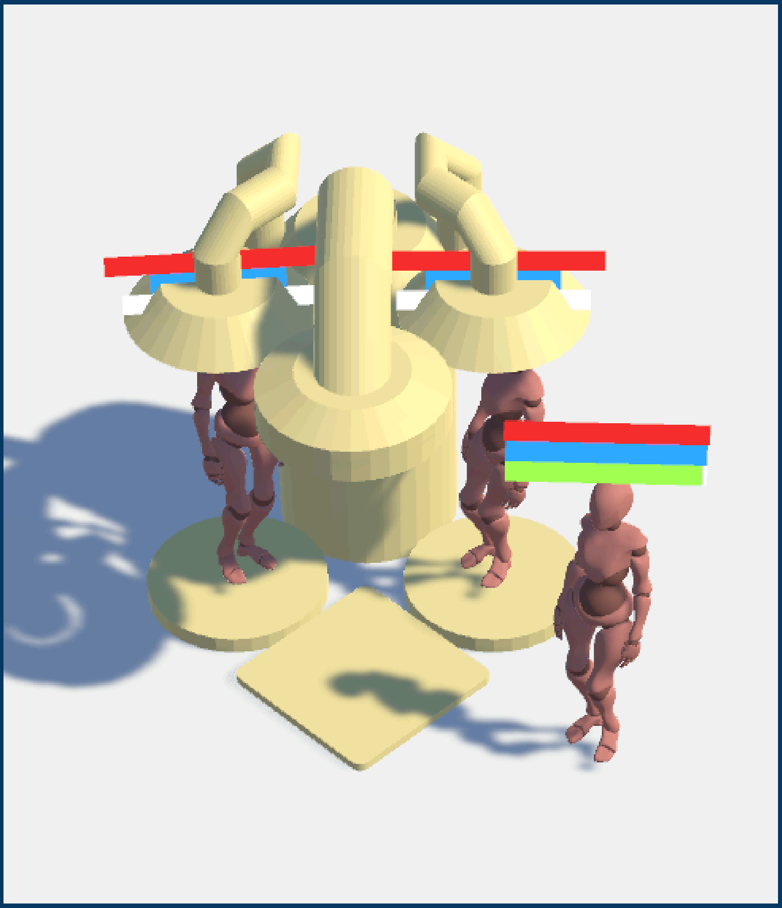

8. Bar



9. Yoga mat



10. Population Generator



Course Evaluation

Expectations

I would have hoped for more tutoring (or explicit content regarding game dev), like what programming patterns to use, how team projects are organized in professional firms, etc.
This was: do it by yourself.
Perfectly Fine learning wise, but I will still be pretty much lost when joining a good company regarding best-practices, git skills etc.
Question is also who would have the experience at the faculty to teach something like this (maybe get a guest lecturer from e.g. Ubisoft).

Questions

1. What was the biggest technical difficulty during the project?

git storage being full, merge conflicts, bugs regarding navmesh
2. What was your impression of working with the theme?
like every theme it is beneficial to have some scope to make the idea finding easier, Up and Down as we can see was interpreted very differently by every team
3. Do you think the theme enhanced your game, or would you have been happier with
total freedom?
yes it enhanced it
4. What would you do differently in your next game project?
actually being paid for it ;) . Planning less features before the exam prep time
5. What was your greatest success during the project?
being able to make the individual parts work together
6. Are you happy with the final result of your project?
yes, still some non-game-breaking bugs but playable
7. Do you consider the project a success?
we learned a lot, so yes
8. To what extend did you meet your project plan and milestones (not at all, partly,
mostly, always)?
mostly
9. What improvements would you suggest for the course organization?

Conclusion

Our game distinguishes itself with notable technical achievements and broad scope. Our dedicated effort enabled us to realize many of our ambitious features. We employed advanced programming techniques, developed visually appealing custom artwork, integrated modern AI for lifelike animations and movements,and blending game genres in a novel and engaging way.
Although we initially progressed slowly to lay the groundwork, we successfully implemented nearly all planned features; we see our game as a success.