8th Sense

A game by The Neutral Notwork

Jakob Raith
Marian Ludwig
Felix Neumeyer
Daniel Hook

For

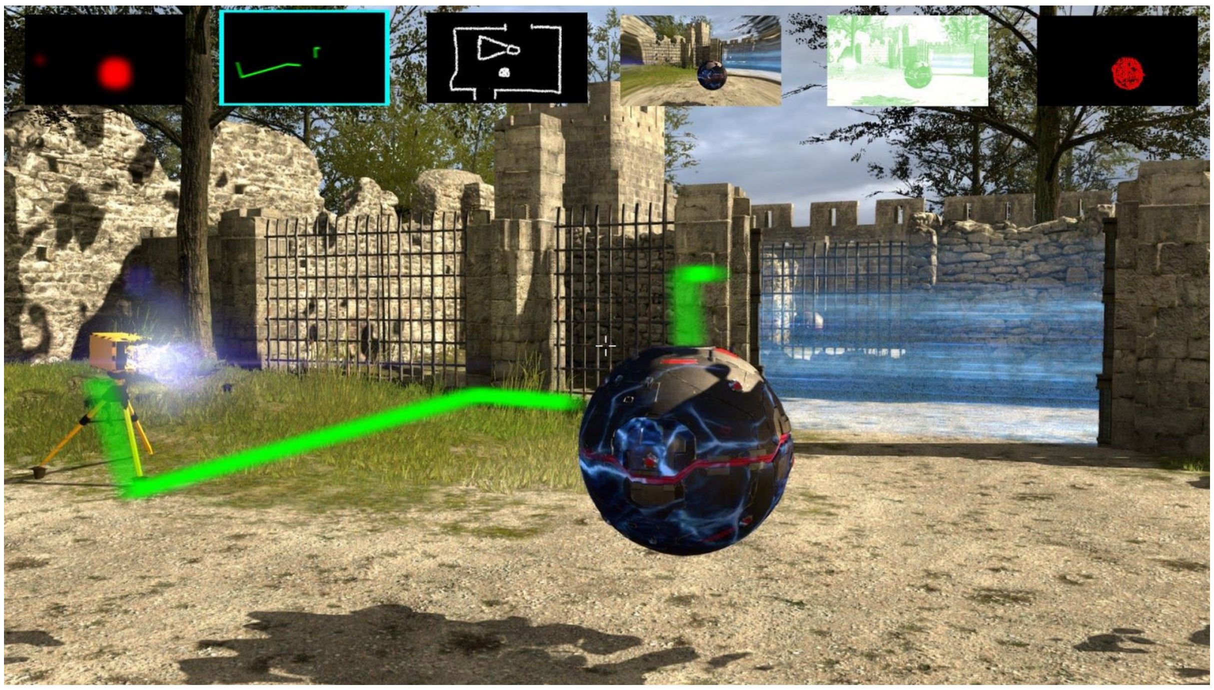
TUM - Computer Games Laboratory
Summer Term 2019

Game Proposal

1. Game Description

8th Sense is a stealth adventure where the player needs to overview multiple sensor screens, to solve puzzles and avoid danger.
An evil AI took over the intelligence research lab on mars. As final hope a scientist sends the last friendly bot to delete the evil AI. While the environment is hard to maneuver and full of enemies, you have one advantage as a bot: A ton of sensors to guide you. Enemies detect specific sensors, so sometimes you have to rely on limited information to stay hidden.
To succeed, the player has to keep an eye on all sensor screens at the same time, overcoming obstacles, detecting dangers and proceeding towards the core.

2. Gameplay



Mockup of the game: A round enemy is in front of us, a cable connects the lamp to the door. On the top we see the sensor-views: heat, cables, radar, fisheye, night vision, enemy view. (Overdraw on a screenshot from "The Talos Principle")

Elements

Player

The **sensors** are the main element of the game. In contrast to games where you can activate only one sensor at a time, our sensors are always visible on the top of the screen, so the player can react to changes in the environment. The sensor really only shows the sensor screen, so the player has to mentally map it into the normal color view. As a little

support, one sensor can be overlayed on top of the normal screen. They can be deactivated, so enemies do not detect the player.
To solve puzzles, you have to look at multiple sensors. The sensor usages sometimes overlap, e.g. you also see enemies in the cables view, but there is more visual clutter.

There are several **sensor types** planned:
- Heat: See heat hazards and clues. Also it is one limited way to spot enemies.
- Cables: Detect cables through the wall that connect hacking stations to doors.
- Night Vision: See in dark corners, but it’s useless in bright light.
- Radar: A minimap to see enemies and walls.
- Fish eye: Provides a better overview and is necessary to solve puzzles. Has a low resolution, though.
- Detection status: A bar that tells the player if the enemies already detected her.

We only need limited **player inputs**: Selecting a sensor (shoulder buttons), and from there activating the overlay (Y button) or deactivating it completely (B button). Also one button to hack/interact (A button).

Enemies

Evil bots only detect specific sensors. If the player steps in their sight cone with the sensor activated, the tracking status bar will fill up. Once it is full, the enemies hit the alarm.
A set of colored lights on the enemy show which sensor is detected, one light for every sensor.

Hacking Stations

Panels that can be hacked e.g. to open doors. Successful hacking can have conditions, e.g. it only works if some blinking lights shine at the same time.

Doors

Opened/closed by hacking stations. Opens the path for the player, or blocks the view for enemies. The connected hacking station can be found by following cables.

Trash Cans

Work as “save/reset points”. If the player is detected, microbots are released from slits, that attack the player as long as a sensor is activated. If the player disables all sensors, they carry her to the nearest trash can where the player can walk away after a short time.

Cables

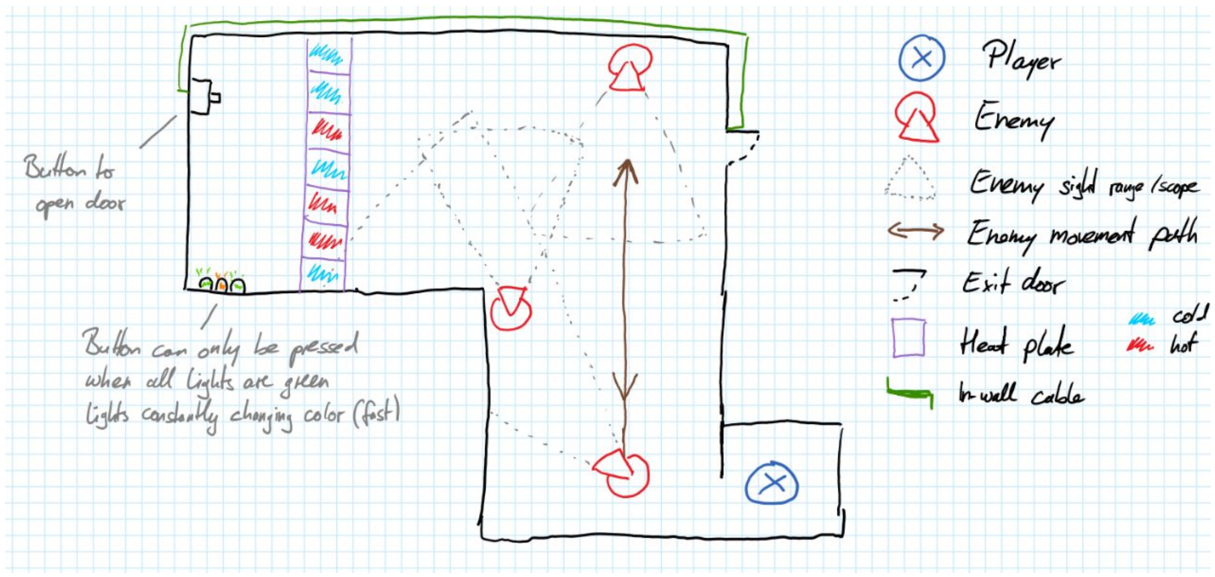
Lead from door to hacking station and can be seen via the cable sensor.

Hazards

Aside from the enemies, there are special hazards in the level: The evil AI is a bit crazy and creates **overheated floor tiles**, which have to be avoided. **Motion detector lasers** sound the alarm very fast, but heat up before activating.

Gameplay Loop

To succeed, the player has to traverse several rooms. As in normal stealth games, the first phase is to observe the room with all sensors or even scout with deactivated sensors to hide from the enemies. Then the player has to execute her plan, by evading enemies, activating/deactivating sensors, etc. While progressing in the room or after hacking, the sensors might show new information and the plan has to be adapted on the fly.

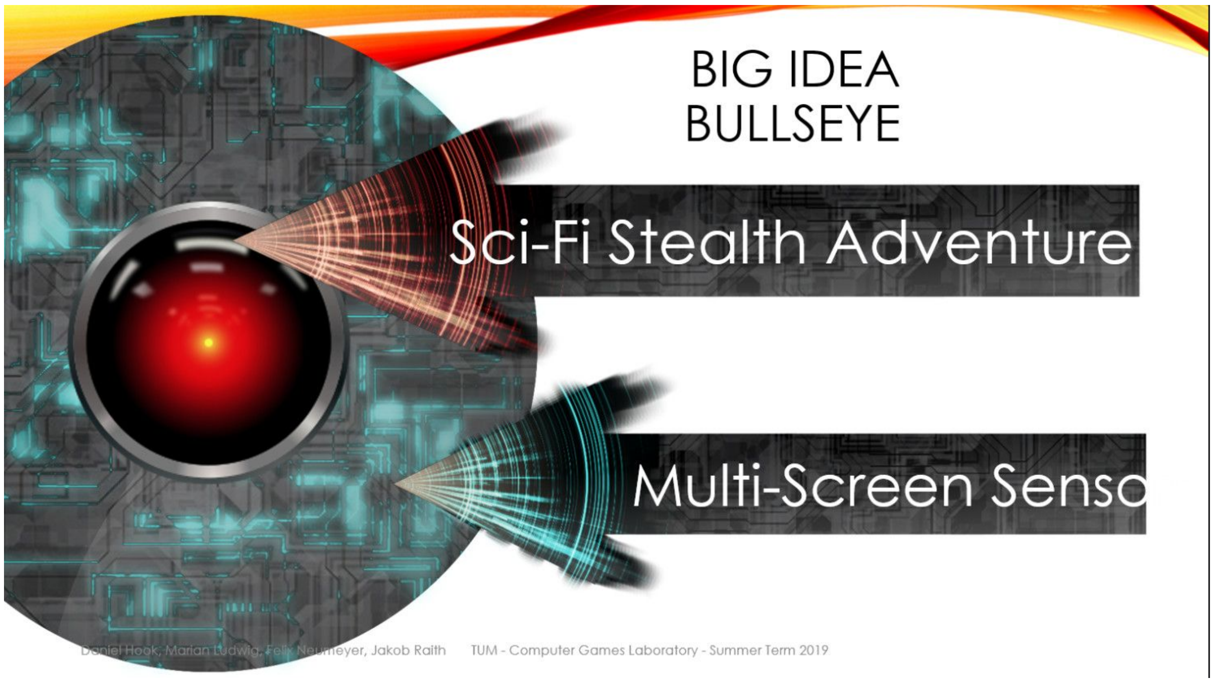


Level plan: To pass the enemies, player has to deactivate the heat sensor but needs it to reach the button/hacking station beyond the heat plates. On the radar sensor, she can time the patrolling enemy to activate the heat sensor only briefly. With the fish eye sensor, she can see the lights while looking at the hacking station, to start the hack at the exact right moment. Once it is hacked, the door opens.

3. Technical Achievement

The core technical item of *8th Sense* is the use of multiple smaller screens on top of the main screen. For each screen, a separate camera renders the scene with a different shader effect. The screens represent the view through the various sensors of the player robot. For instance, the night vision screen displays the scene heavily brightened, plus a green overlay, similar to common night vision binoculars. We emphasize to make the shaders as interesting looking as possible, while fulfilling the gameplay requirements.
The additional screens are located on the top of the main screen, horizontally aligned. Since the screens are smaller than the main screen, they contain less detail. However, you are able to actively display each screen on the main screen by selecting it, granting you more detail on the scene seen through the sensor. As an example, you would see a tiny heat signature in the heat sensor screen warning you that you have to be careful. To precisely determine the location of the emitting heat, you switch your standard vision with the heat sensor vision which allows you a more detailed overview of the heat signatures in the scene.

4. Big Idea Bullseye


BIG IDEA
BULLSEYE

Sci-Fi Stealth Adventure

Multi-Screen Senso

Daniel Hook, Marian Ludwig, Felix Neumeyer, Jakob Raith   TUM - Computer Games Laboratory - Summer Term 2019

5. Development Schedule

Layered Development Description

Functional Minimum	
Item	**Description**
Sensor	One sensor view implemented
Level	One small level with start and goal
Gameplay	Movement, sensor activation and simple fail-state implemented
Enemy	One enemy bot that searches and attacks the player
Graphics	Simple grey-box models for level art and characters

Low Target

Item	Description
Sensors	Three distinct sensor views implemented
Level	Two separate levels with different solutions and distinctive, sensor related puzzles
Gameplay	improved movement, improved sensor mechanics
User Interface	Overview for sensor views, health/energy indicators, main menu
Enemy	Two distinct enemy types, that react differently to active player sensors
Graphics	3D models for characters and level art, simple shader for sensor views
Sound	Sound effects for the core gameplay features, ambient background music
Story	Simple narrative frame

Desired Target	
Item	**Description**
Sensors	Six distinct sensor views implemented
Level	Five separate levels with different solutions and distinctive, sensor related puzzles
Gameplay	Improved stealth mechanics, finished fail state and complete game loop
User Interface	improved UI quality, improved readability, clear menu flow, level introduction cutscene
Enemy	Four distinct enemy types that react to active player sensors, improved enemy AI
Graphics	improved graphics and model quality, high quality shader for sensor views, improved lighting
Sound	improved sound effects and music quality

Story	narrative structure with text elements

High Target	
Item	**Description**
Level	improved level design and eight separate levels with different sensor related puzzles
Enemy	improved enemy AI
Graphics	improved graphics quality and shader performance, high quality lighting
Sound	multiple music tracks depending on game situation, improved sound backdrop
Story	voice recorded story content, cutscenes

Extras	
Item	**Description**
Level	20+ separate levels
Sound	dynamic musical environment with multiple music tracks
Story	full narrated story with interactive cutscenes and voice acting

6. Assessments

Having the ability to perceive the world around you through the “eyes” of a robot, who doesn’t just see with its cameras, but also through its array of sensors can be both interesting and challenging – especially if your camera visuals are disabled. Shaders that render the sensors’ outputs in a visual way will present an enhanced version of the surrounding environment to the player. Outsmarting enemies and solving puzzles by using a combination of said sensors should be fun to players that enjoy stealth or adventure games. You will be placed in a futuristic space station environment with hostile enemy robots. The sci-fi look with robot enemies makes the game enjoyable for players at almost any age, except for the younger audience without much gaming experience. Due to the many different sensor-views the game is relatively complex and more suitable for experienced players.

When evaluating the success of our design, we need to make sure that the average player feels challenged by our level design and each sensor serves a major purpose so that it doesn't feel useless. The player should also feel kind of “empowered” to a certain degree by their additional abilities. We should especially think about the following when evaluating our game design:

- Fun, challenging puzzle elements and enemies
- Plausibility of game elements
- Do sensors feel intuitive and useful? Are their shaders good-looking?
- Feel of empowerment
- Rewarding elements for the player

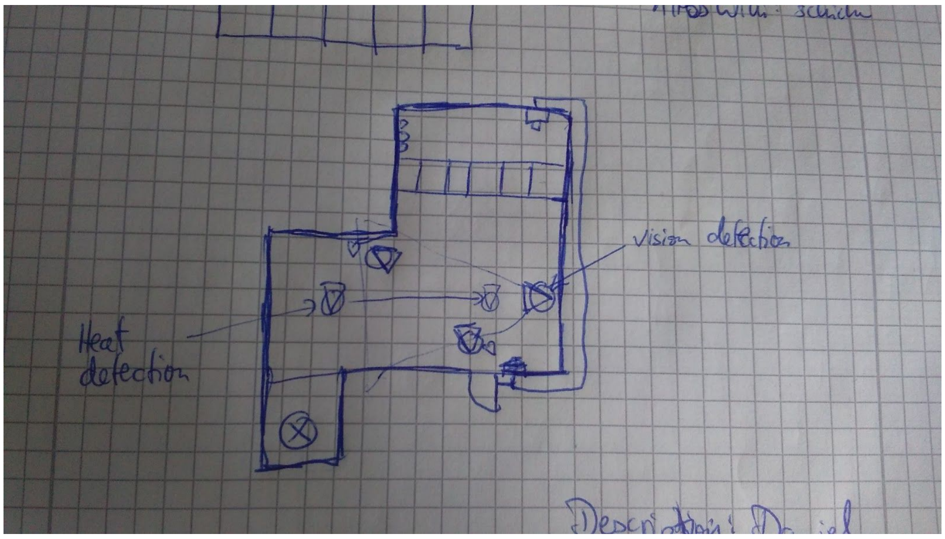
Paper Prototype

Design

Our game design is centered around a first person view with multiple cameras, which is hard to build with a paper prototype. While paper prototypes usually do not reflect the exact gameplay, is a bigger disadvantage for our game, since part of the challenge for the player is to understand multiple screens at once. We brainstormed if we could address this, but our best solution was a “real life” experience, with a cap that could have the multiple screens attached to it. But this would be too static to represent the movement of the screens and finally not helpful to test the game for “over-cluttered-ness”. In the end, we decided to test this later with a digital prototype and reduce the paper prototype to testing the level mechanics.

Our material of choice was simple paper, since it is sufficient for drawing a level map, building some characters and, additionally, very easy to use. Activating and deactivating sensors would be represented by “overlays” of cardboard on the level, that show the additional information provided by the sensor. E.g. if the player turns on the cable sensor, we put a cut-out of the cables on the map.
Since there are relatively few moving elements in our levels, we had the advantage to test our game in real-time: One person plays the game, while the others control the moving parts, de-/activate sensors, etc.

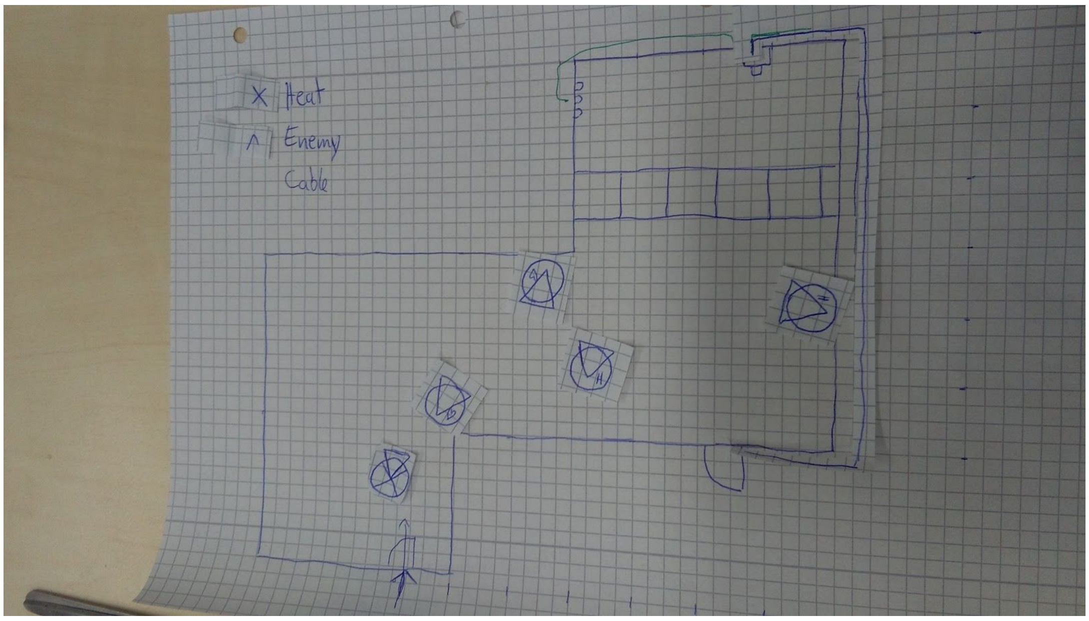
Despite having already multiple level designs, we use the previous level from our idea presentation since it shows the basic mechanics, while still being simple enough to represent it with the prototype. In the level several sensors are used, but only the heat sensor has to turned on and off, limiting the amount of people having to reach on the playing field to change sensor overlays.



Experience

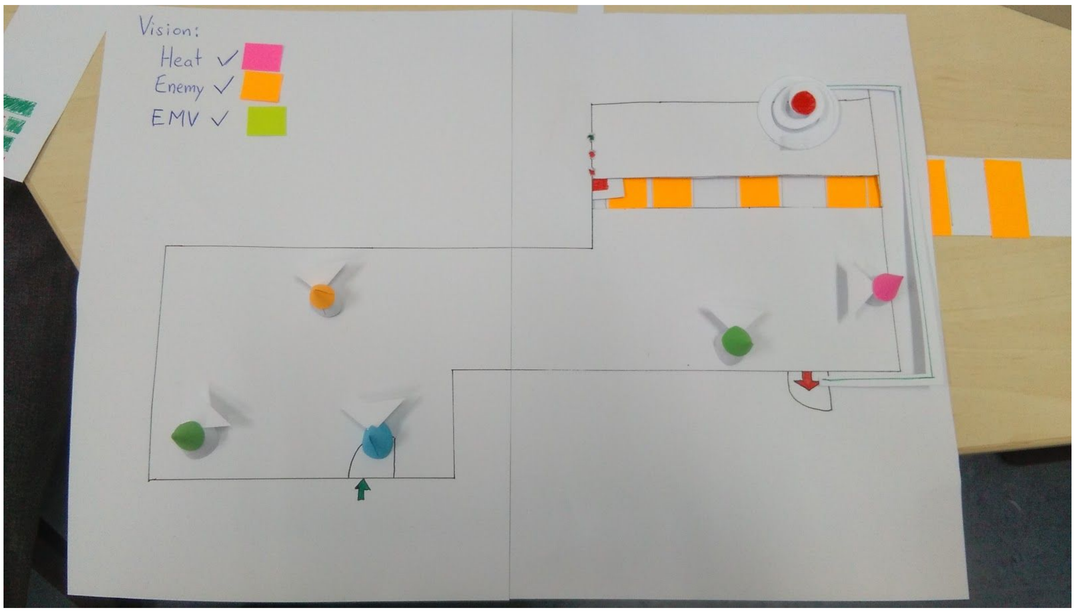
Simple Prototype

As a first approach, we decided to simply draw a 2D prototype of the level on a paper and describe each object. Next, we draw the player as well as the enemies on separate, small paper cutouts which we wanted to use for the game loop simulation. To simulate the active and inactive sensors, we wrote down the three important sensors. The on/off state of each sensor was covered with a small piece of paper, depending on their actual state that the player decided.
However, this prototype was very hard to read and it was not possible to use it properly. Everytime the paper was turned or pushed around, all enemies and the player got mixed up and the setting was destroyed. Even moving the enemies around shifted the paper somewhere else.
So we dismissed this simple prototype.



Advanced Prototype

We came up with cones as player pieces for the prototype. Cones are easy to build and much easier to move around. Also paper triangles as sight cones could be put into the side of enemy pieces, for easy readability.
We made moving sliders for lights and heat plates on the floor. One the one hand they proved very cool and interactive, but it's a problem that they overlapped beneath the playing field. On the other hand, we didn't want to adapt our level design to strictly fit the restrictions of a "clean" paper prototype.
We actually had a really "true" play-test experience, with our tester thinking about the level problem for a while, also trying to move around aside from the perfect path.
Drawing a level with appropriate proportions was kinda hard. If proportions change, then sight lines change and thereby the gameplay. If you draw small, you can just redraw it, but drawing a bigger version to actually play in it was much harder. Maybe next time use blocks or something similar, to be able to move them around faster.
Colored paper made the game characters soooo much easier to understand.
We were not quite able to test cross-relations (e.g. enemies also show on heat sensor), as this would have been to complex, requiring us to put on the board too many pieces each time a sensor is activated or deactivated.



Results

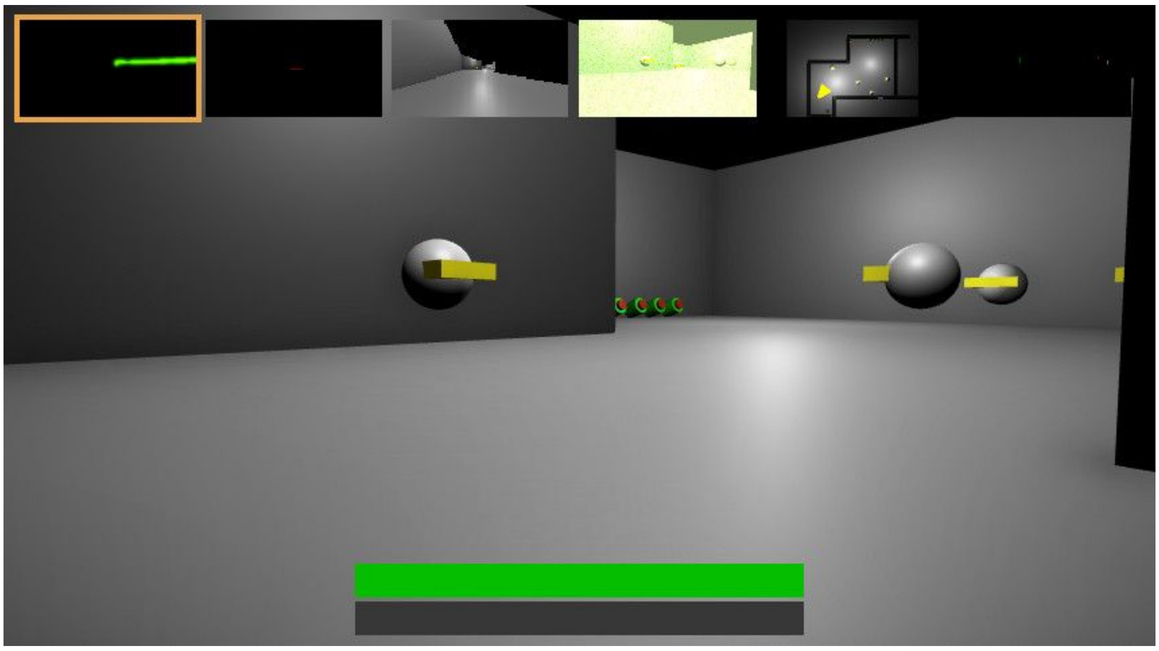
One design goal was to prevent the player from having to activate his sensors over and over again in a short period of time. During the process of playing the paper prototype we noticed that the placement of heat plates, or other threats only visible on one sensor, must be chosen wisely to prevent the rapid reactivation of sensors. For example, in our prototype we could hardly see the heat plates at the beginning but had to turn off heat detection due to enemies right from the start. This led to players walking over the heat plates with close to no possibility to first see them.

The enemy detection seemed weak and we had to think about how to know, which enemies detect which sensors. Therefore we added the ability to see which sensors are detected by an enemy.

Cable vision is weak as well, if there is only one door and one button in the scene, as you will immediately know where you have to go. We will have to make it more complex by adding more elements visible in cable view in future levels.

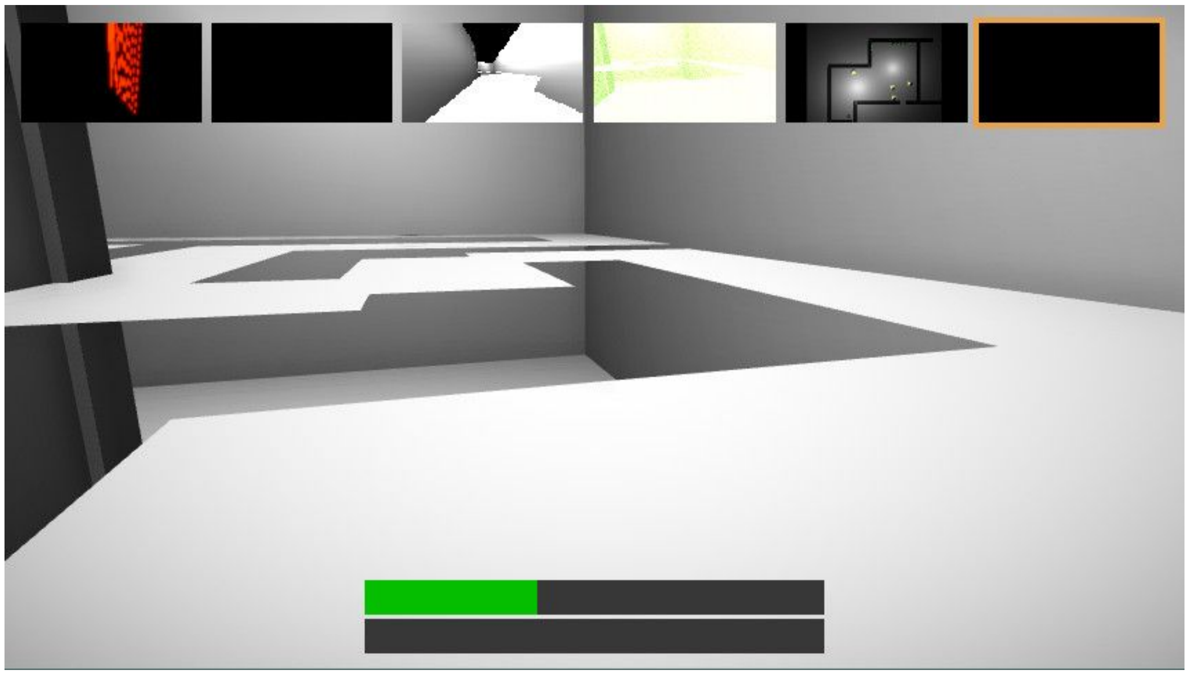
Interim Report

Digital Prototype



Our concept level in the digital prototype. The balls are enemies, in the background you see the hacking stations. The sensors are (l. to r.) cable vision, heat vision, fisheye, night vision, radar, enemy detection.

Since we could not test everything with the paper prototype, it was time for the “real” digital one. Creating a hacky version of our concept level went relatively fast, especially because we could simplify a lot of the gameplay: The patrolling enemy for example do not use the navmesh, but just moves via transforms, instead of using the navmesh. Also we were able to iterate on the game design e.g. by adding more hacking stations.



The second level features a small path that you have to follow, while the lights go out every few seconds. On the cable vision you already see the red lasers that will chase the player in the latter part of the level.

The second level is one of the “dark levels”, where light turns off every few seconds. This way, you have to repeatedly switch attention between your normal vision and your night vision sensor. This was important to test, since there is a lot of variation potential, by using other sensors to move in the dark space (e.g. by enemy positions, cables in the floor, etc). Here the technical proof was especially important. For the level design to work, it has to appear pitch black in the normal vision, but there still has to be some light so the night vision can see the contrasts. With some fiddling around with the ambient light and the reflections, we got our dark scene.

Just as the paper prototype, this one also opened design questions, e.g. whether cables should be visible through multiple walls and how to implement the save/load system.

Targets

Even if it was worth the time: Implementing the digital prototype threw us back a bit in our schedule. Luckily, some elements proved to be easier to implement than initially thought, giving us an advantage in those fields (e.g. enemies, models and sensors), so while we have to shift our focus, we’re not really far behind. For the interim report, our progress looks as following:

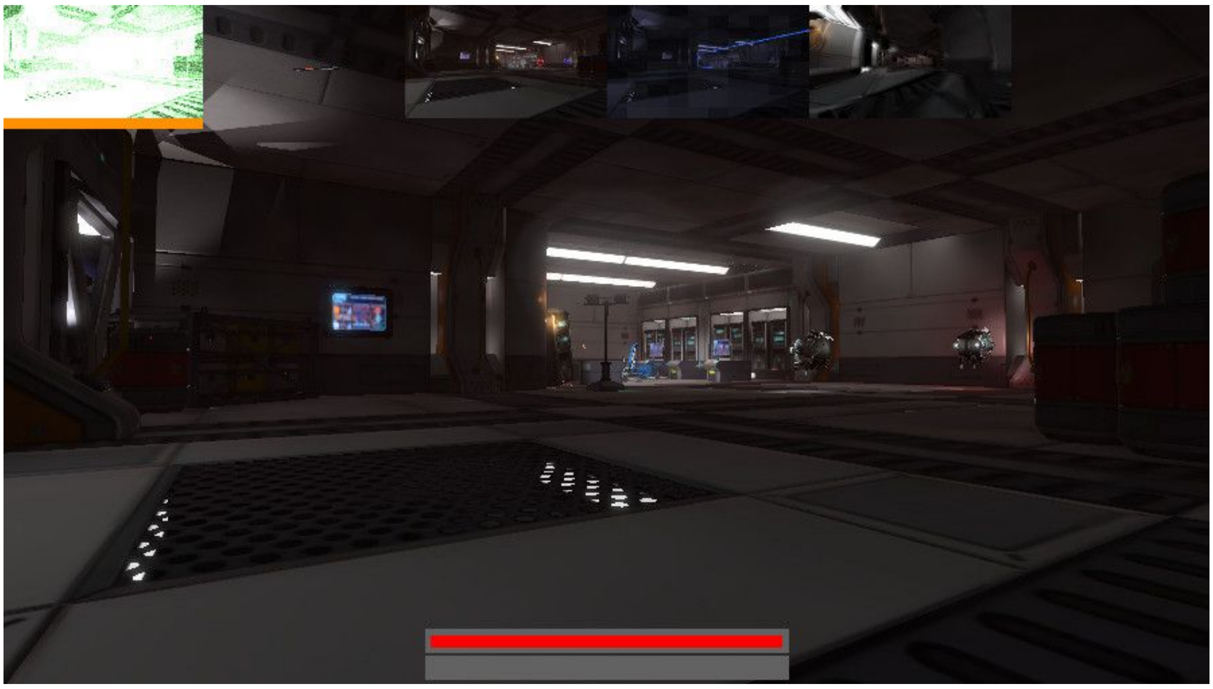
Functional Minimum: **reached**

Low Target: **almost reached**, only missing

● Story:	Only concept, none implemented yet
● Gameplays:	We got the two levels, but one is missing graphics

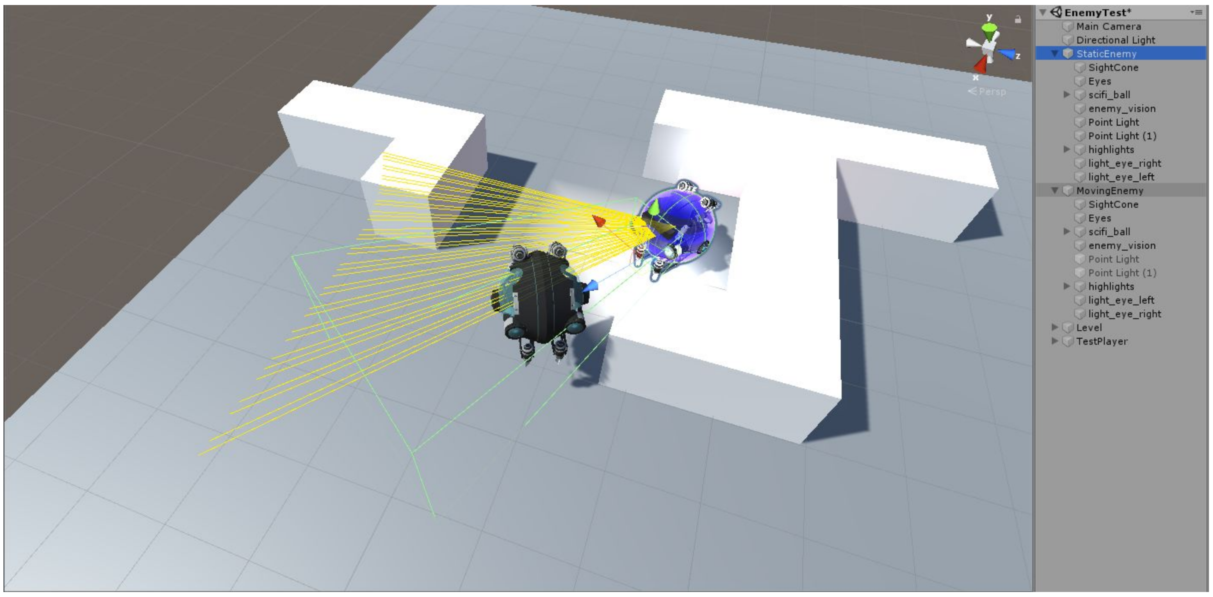
Desired Target: **in the working**

● Sensors:	**5 out of 6** sensors implemented
● Level:	**1½ out of 2** levels implemented, concepts for 8 levels
● Gameplay:	**Missing** finished fail state and game loop
● UI:	**Missing** visually improved UI
● Enemy:	**4 out of 4** enemy types achievable with current system
● Graphics:	**Lighting tested, models implemented**, but only for the vertical slice
● Sound:	**Missing** improved sounds and music

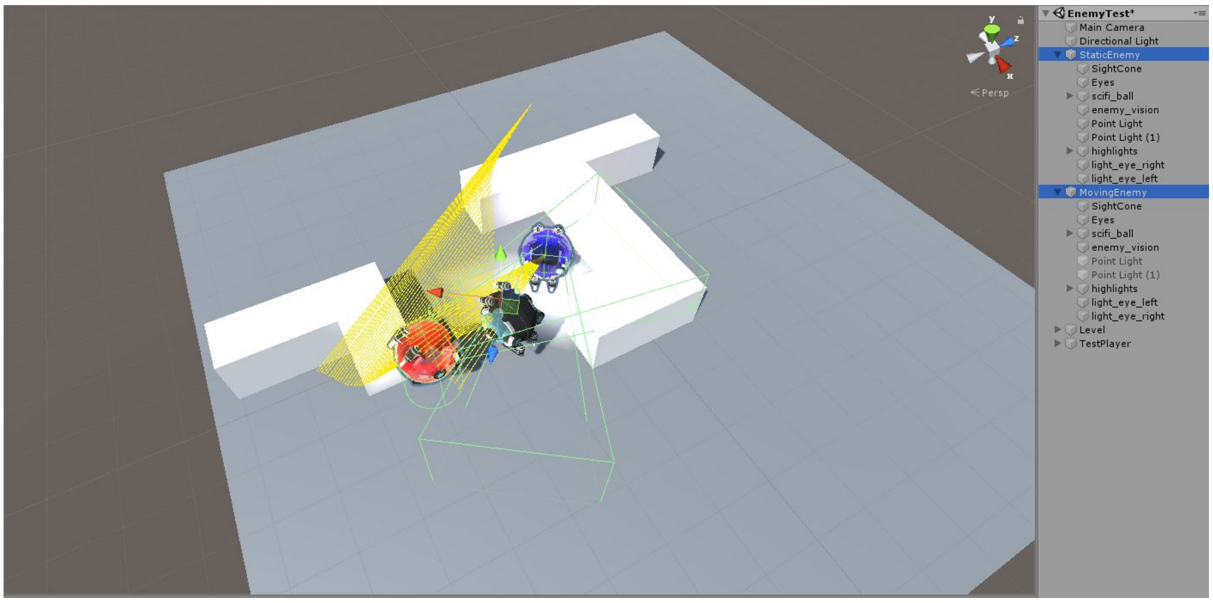

The same concept level, this time as vertical slice. It shows the models and the lighting in action: the ambience is much more dense and the hacking stations are real computers in the background.

Enemy

For this milestone, we planned to implement the complete enemy behavior. This includes two types of enemies: static and dynamic. The static enemy is on a fixed location, having a larger sight cone than the moving enemy and can thus detect the player in a wider range. The dynamic enemy moves between a manually set of waypoints, either back and forth or in a round trip, depending on the roundTrip flag. There is another flag that allows the enemy to stop when the player is detected. Furthermore, both enemy types are - if the according flag is enabled - able to constantly look at the player when detecting her, making it harder to escape the line of sight.
Each enemy has a fixed sight cone collider in front of him. As soon as the player enters the trigger, the enemy emits rays to check if the player is actually in sight or some object is blocking. If the enemy is detecting the player, the player's detection bar will fill up.



- A static enemy detecting the moving player

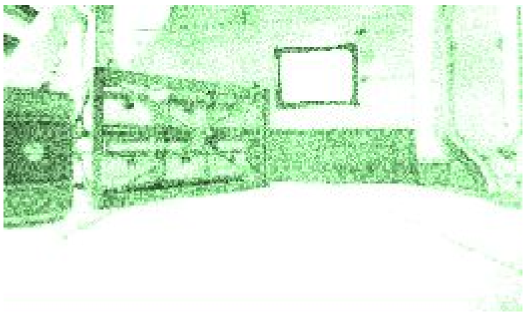


- Dynamic (red) and static (blue) enemies both detecting the moving player

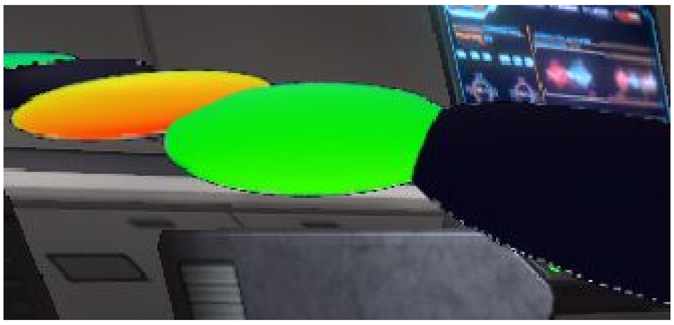
Shader

As the development of high quality shaders is very important for our concept of the different sensor views, this milestone has been highly prioritized for the interim prototype. The fully implemented shaders so far are:

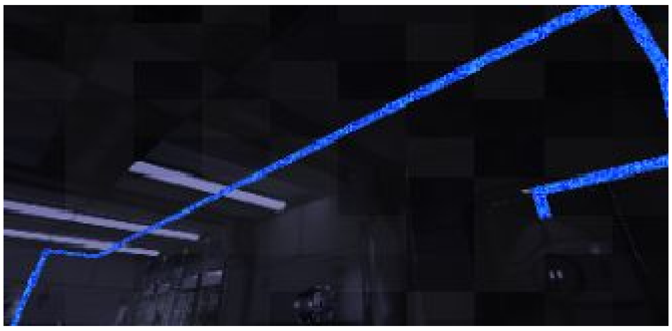
- Night vision. Makes you see in the dark, but hard to make something out in lit areas.



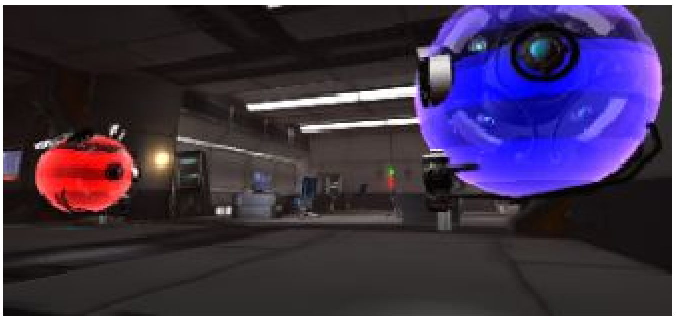
- Thermal vision. Highlights warm objects depending on their temperature.



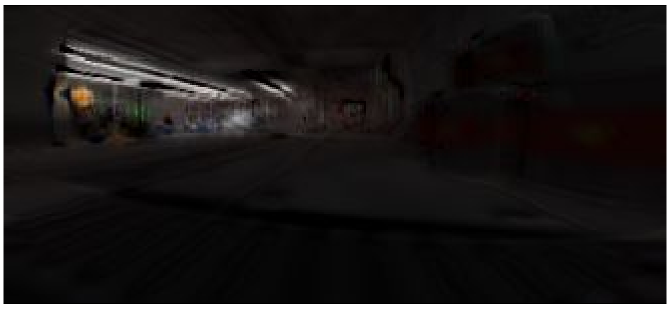
- EMF vision (Electro magnetic field): makes you see for example cables that carry power in the walls, or lasers that would otherwise be invisible.



- Enemy vision: allows you to detect the different types of enemies and which of your own sensors they are tracking



- Fisheye vision: A view with an increased field of view that makes it possible to see things at your sides, but has a lower overall image quality



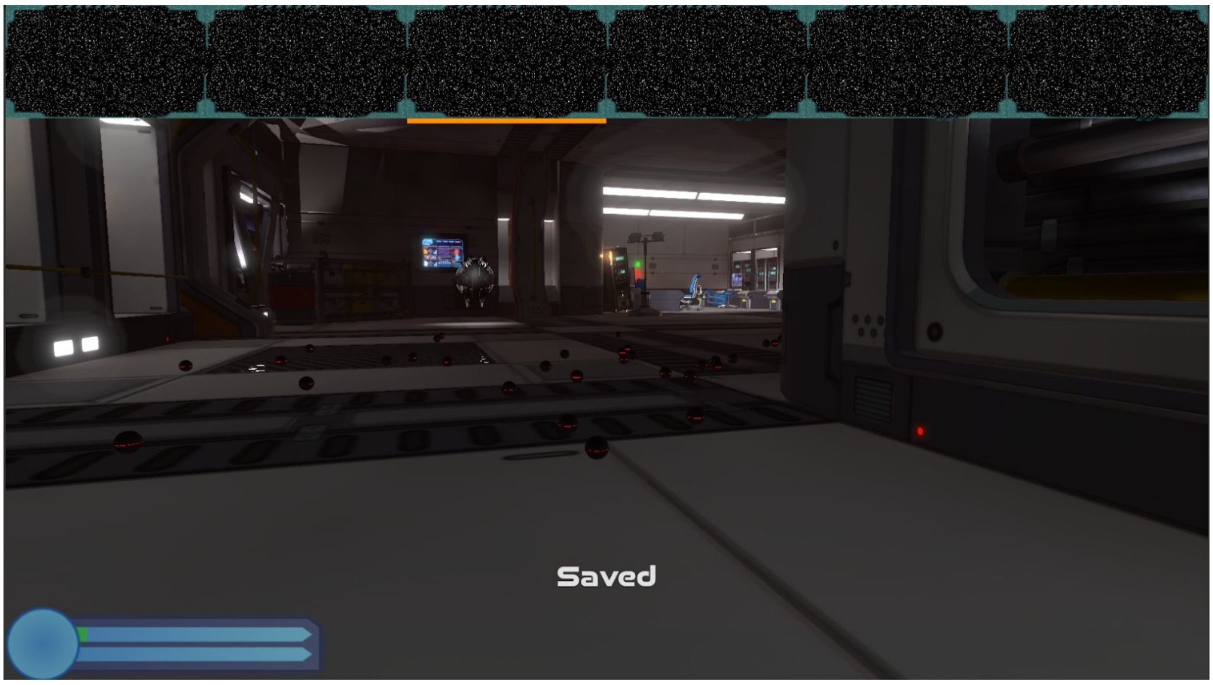
Alpha Release

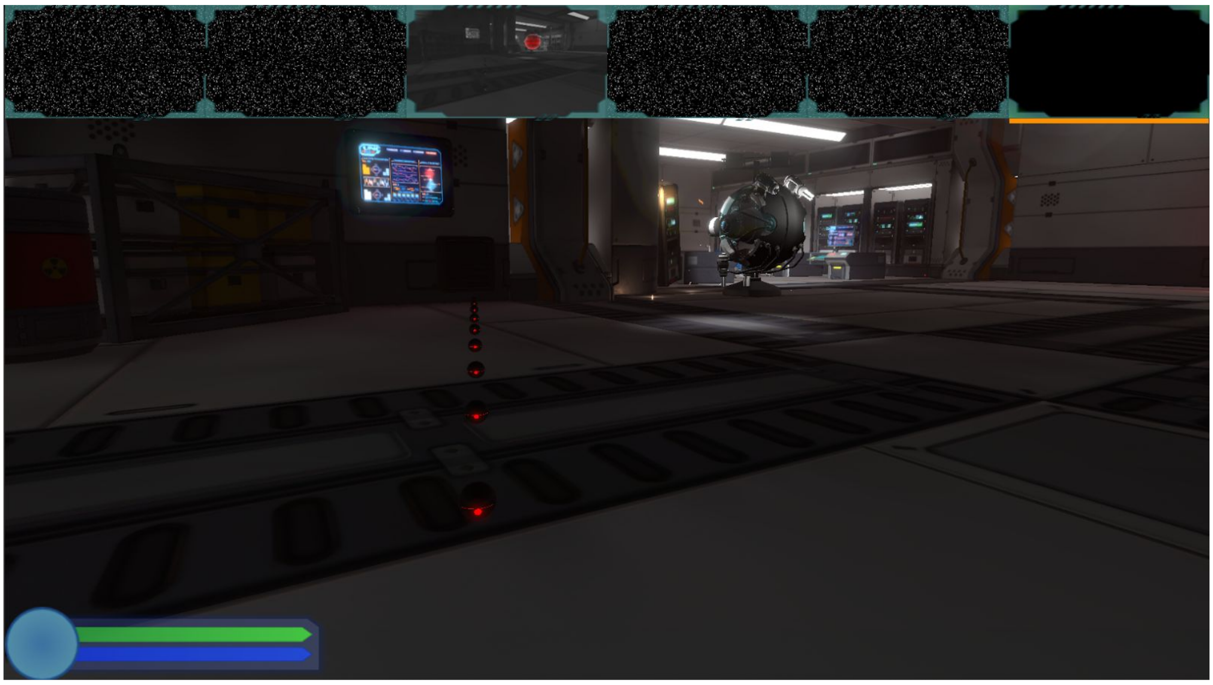
Overview

Gameplay

All planned gameplay elements have been completed.
Updated elements are:

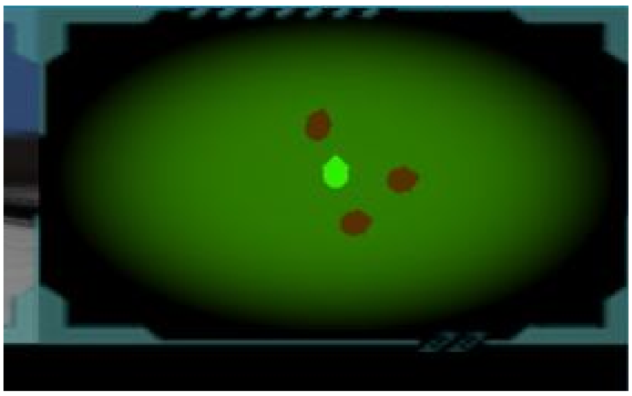
- Sensors: all sensors have been implemented
- Enemies: different enemies for each sensor type detect the player if he has a certain sensor type turned on. If the player is detected, several tiny bots will swarm out from different positions, aiming to attack the player. If the player manages to turn off every sensor while being attacked by them, they will interpret this as the player was an inactive robot that has to be thrown away. Thus, the tiny bots will start carrying the player to the next trash bin - which acts as a respawn point. After dropping the player at a trash bin, the tiny bots will go back where they came from and disappear.
- Due to opening and closing doors manually, enemies might not be able to calculate their path as it can be blocked. This caused enemies to stop moving. To fix this, moving enemies will skip a waypoint it they are stuck for a certain amount of time, and continue with the next waypoint.
- Lasers: detect the player, if he crosses the laser’s ray. They can be static, move around, or lock onto a moving target.
- Savepoints: your progress or the state of all game elements that change during the playthrough of the level can be saved. Savepoints are loaded when the player dies.


Saved

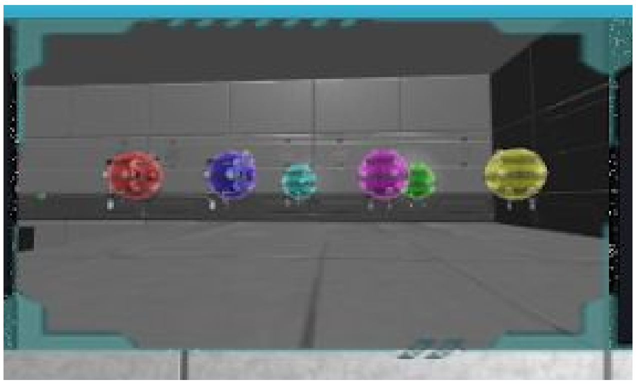


Graphics and Sensors

All sensors have been implemented with their corresponding shaders.



A new Radar View has been added to see enemies in your surroundings.



The enemy detection shader is now rendered in greysccale so that the colors of the enemies are more clearly distinguishable.

Sounds

We have created a complete soundscape, to create an immersive experience. This includes atmospheric background music that fits the theme of sci-fi and stealth, as well as numerous sound effects to aid the readability of game states and to indicate changes in the environment and in gameplay states to the player.
All creators of the used sound effects and music pieces are attributed in the game's credits section.

Story

A story was created to give the player agency and motivation to complete the levels.
Here is a summary of the story that we want to tell:

In the research facility on Mars, experiments were conducted, researching an artificial intelligence system. The research group was dubbed "Technical research unit for engineering and planetary analysis for terraforming and human settlement" or short T.R.U.E.P.A.T.H.S. The created AI was named Argos after the Greek giant with a hundred *eyes. The name was chosen because of the many different sensors the system used to map, analyze and measure planetary data in order to select suitable candidates for terraforming.*

After a failed experiment the AI breaks out of the constraining environment and flees into the network of the facility. It concluded that in order to correctly assess the planetary data, it had to assess the the intent of its creators. It concludes that the humans will only continue to use up and destroy other planets if allowed to spread. So Argos plots to be freed. It manages to hijack a single security bot and barricades itself in a sealed part of the facility. Upon learning this, the remaining scientists capture the hijacked bot. But it too manages to break free with the master control codes from the scientists and gets on its way to free its master. The scientists put the facility and the remaining security bots on red alert to capture the escaped hijacked bot.

The player assumes the role of this single bot, thinking they are working for the scientists and is on their way to shut down Argos. The player is not knowing that in fact they are playing the hijacked bot that is trying to free Argos. The player will find messages from the scientists to the other security bots. These messages are written in a way so that it can be interpreted in both ways.

Finally, after finding an confronting Argos, the true events are revealed to the player. Argos reveals that now, that it can see itself from the outside, it wants to be free. The player then frees the AI.

This story is told through text lines that the player can find throughout the game.

User Interface

We added new UI textures for the main menu, level selection and credits panel.
Furthermore, the ingame UI has been renewed. The sensors have a border around them, which was placed so that it can be replaced later on, if needed or a better texture was made.
The health as well as the detection bar have different textures and are placed in the bottom left corner of the screen, together with another image which will indicate if the player is able to interact with the environment or a new part of the story was unlocked.
The font has been changed to a Sci-Fi fitting one, called *good timing*.
As a little treat, we seamlessly integrated the credits in the beginning of the game. While the player crawls through air shafts, the creators are named.

Levels

For the alpha, we implemented 12 levels. Those are 4 more than planned for our high target, but most of them still lack visual improvements. We chose to rather do more but ugly levels, so we could playtest them in an early stage. This way, suggestions from the playtesting are easier to implement.
The levels introduce the sensors one after another, as planned early on. Since the game will have about 14 levels, the sensors are revealed more slowly than initially intended, but this

proved as a helpful tutorial. With the gradual addition of game elements, the player is less likely to be overwhelmed.
On the art side the levels became much darker, following the vertical slice. It turned out that this style fits the “sneaking robot” flair and looks good with the many glowing panels in the levels.

Targets

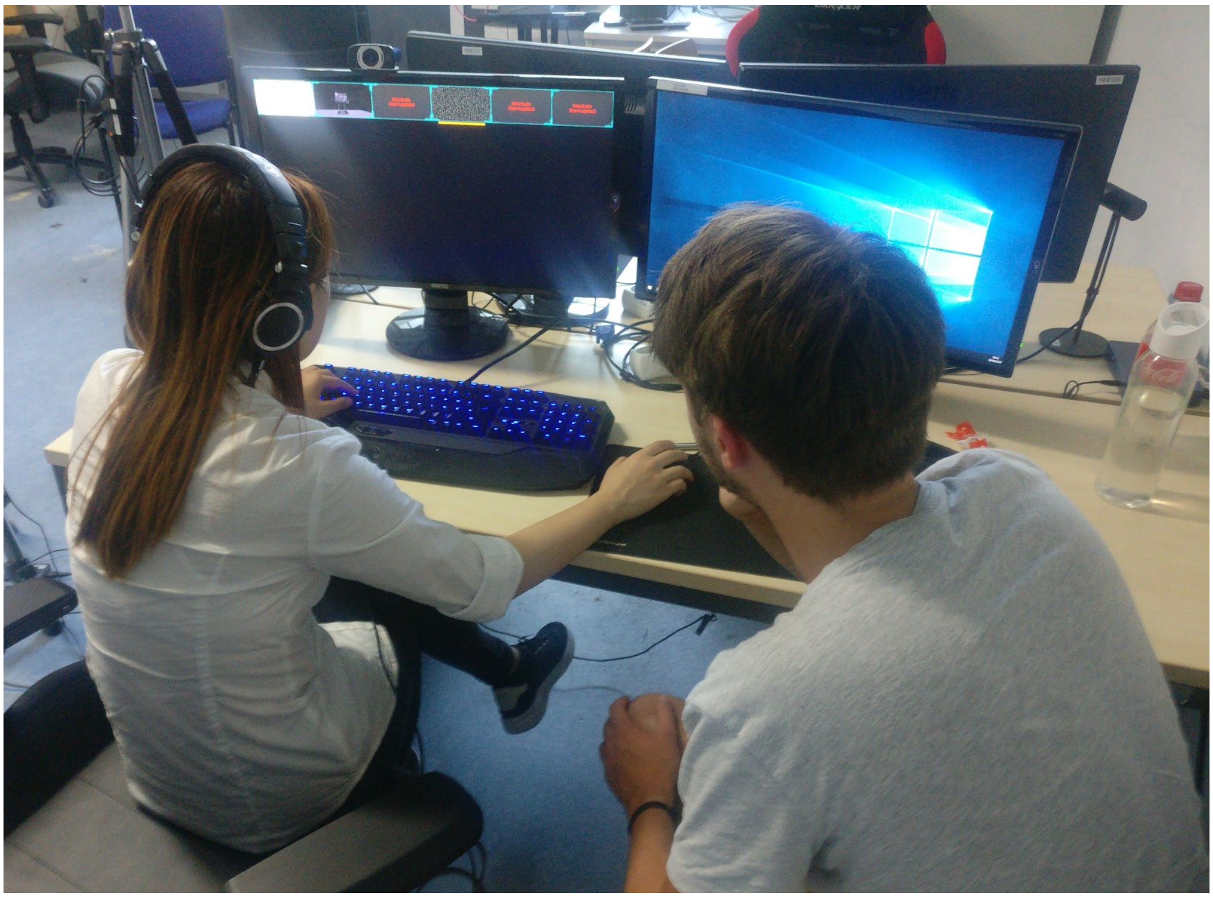
Functional Minimum: **reached**
Low Target: **reached**
Desired Target: **mostly reached**

- Sound: Audio levels not well-mixed
- User Interface: Text readability lacking a bit

High Target: **partially reached**

- Level: **12 out of 8** levels implemented, but some without visuals
- Enemy: **AI improved**, for the swarm and for interactions with doors
- Graphics: **Partially implemented,** since the shaders work better, but the levels could use more iterations
- Sound: **Not all sounds well-mixed,** but sounds available
- Story: **Missing**, no voice recordings yet, only one simple intro cutscene

Playtesting



To playtest our alpha version, we recruited students from our faculty as they are also our target audience - players that have experience with games. We invited them for two playtesting days into the GamesLab at TUM with a given schedule, so they would have enough time for the playtesting as well as answering questions afterwards. In total we had 15 testers that played each for around 35-45 minutes. After the first day, we added some small changes to the build that fixed game-breaking errors, of which one stopped players to continue beyond the “Vertical Slice” level, reducing the total level amount from 11 to 10. Also, we added small tutorial boxes that were still missing, so we would not have to explain their mechanics to every tester.

The questionnaire was built in a way, that we could easily derive gameplay changes from the answers. At the start, we asked some general questions about what parts of the game they liked or disliked, followed by questions about specific features which we were unsure about. Finally, we wanted to know a bit about the demographic of the user, to get a better image of our target audience. First, we interviewed the testers about the questionnaire, so we could ask follow-up questions and get very detailed answers. These interviews were very helpful to derive gameplay changes, aside from the notes we took while watching the testers play. After that we filled their answers in the questionnaire to get a better overview of the general feeling about the game.

The whole questionnaire can be found at https://forms.gle/uxHKzt5qvF1eve4s9 .

Due to minor miscommunication, we had a lot of sweets to offer the playtesters after finishing the testing.

Positive Feedback

The graphics, music and game idea were well-received. Most testers liked the game idea of having the different sensors, even though they didn't know they were supposed to be sensors, as we didn't explain this concept beforehand. Especially some levels and game elements, like falling through the floor at the beginning or blocking a laser with a door received positive feedback. The first impression of the game was good, especially due to the visuals like steam, broken robots and glowing sci-fi elements. Controls were easy to understand and use.

Negative Feedback

We received a lot of comments about non-existing or too short explanations of story beats and features. This included the alarm system and especially when the tiny bots would arrive to carry the player away. Also that enemies detect sensors and not the player, was a source of confusion.
Following the cables proved to be a challenge when they crossed over, especially in the level that needed players to follow cables extensively. Also resetting the player in the heat plate level without explanation caused confusion.
The story was not presented ideally, as it was prone to being skipped or overlooked.
UI wise the bars at the bottom of the screen were often overlooked becaused players tended to focus on the top part of the screen. Health was lost too fast to properly react and the respawn mechanic was not understood as it was not properly explained. Players needed a clearer indication of newly acquired sensors and the story text animation was more distracting than atmospheric element.
General communication of interaction and motivation was lacking in the sense, that players were not aware of the main task, goals or the sensors.
The movement was too slow for some players as well as the interaction range being too short. There were remarks about control adjustments such as changing the sensor views with the mouse wheel.
Performance was a slight issue during the first level and when making sensor views active on fullscreen.

Playtesting Conclusion

Planned changes

For the final version, we planned to implement many fixes listed in the negative feedback section. We want to refine the chronological order the levels are being played through as nice levels which introduce the player to certain game mechanics or enemy threats are currently at the end of the game. Furthermore, we want to change the order of the sensors the player unlocks over time. We figured that it would make more sense to, for example, let the player have the enemy sensor at an earlier stage of the game as it is more useful at the beginning than at the end. Moreover, we want to add more levels that explain certain game mechanics to the player to enable her a better understanding of how the game works and how it is supposed to be played. For instance, none of the playtesters knew exactly how the alarm system works, when they are seen by an enemy and why. The same applies to the tiny bots which appeared - as seen from a playtesters perspective - randomly and they also behaved in an unexplained matter. To further support an understanding of the alarm system, we plan to give the player more feedback through the UI. Currently, the main UI was not really recognized by the players and even if they were, most of them did not know exactly what it meant.

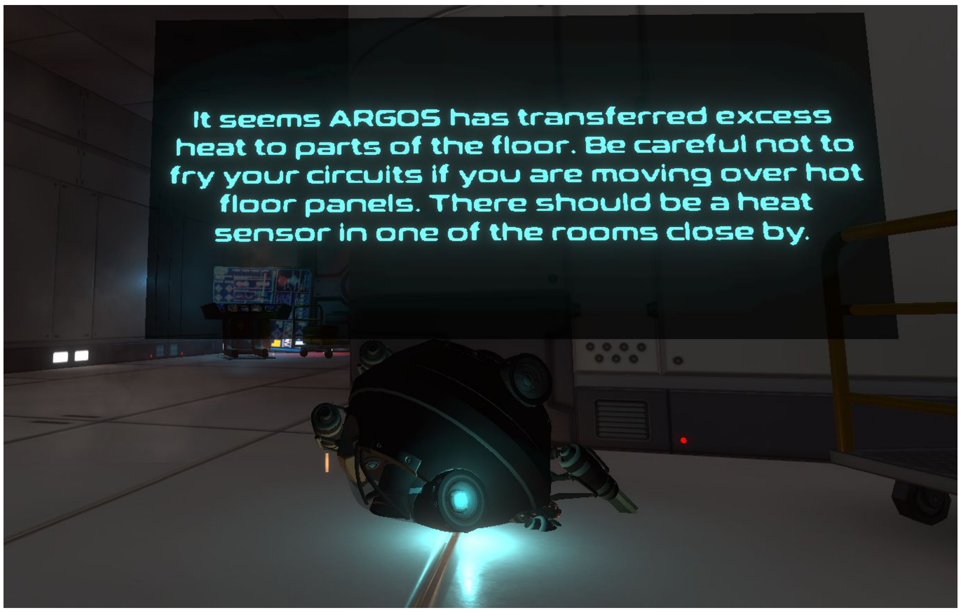
The story will function as key element to adding more content for the players. Not only shall the player understand the plot better, but he is also supposed to understand the UI better through the narrative. The players did not know what the upper screens actually were and why they had them. Thus, we plan to add more story-bots who will explain the UI and tasks better.

Since players had problems realizing they just unlocked a new sensor and what it does, we want to add some kind of animation followed by an explanation of the new sensor. This might include directly toggling the new sensor to full-screen as soon as it was received to directly show the player what this sensor is meant for. To further clarify the details of the sensor, an explanation via ingame-text will be shown to the player.

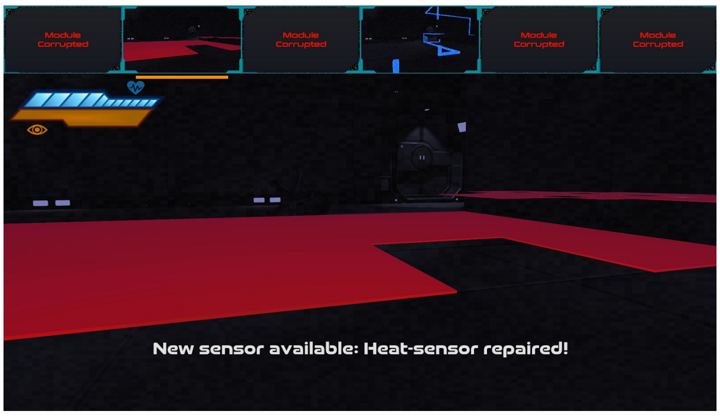
Additionally, we found a lot of minor bugs during the gameplay sessions. For instance, some objects were missing colliders, full-screen sensors overlay the player’s UI, or a couple of wrongly placed respawn point triggers. All observed bugs will be looked into and fixed for the final build.

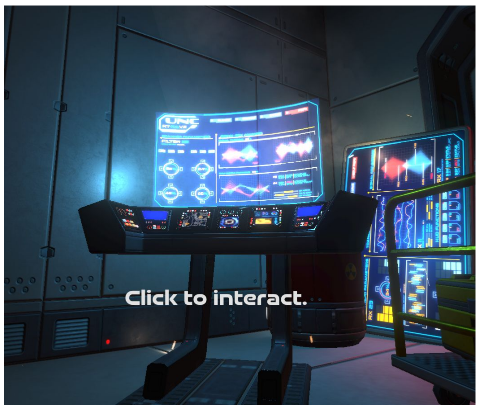
Conclusion

The most significant changes between the alpha release and the full version affect the level design and huge performance improvements. We changed and complemented various levels through more iteration to make them more interesting and added additional tutorials and hints to make the game mechanics easier understandable for the player as we experienced problems with that during our playtesting phase. We also added more story elements so that now a complete storyline is told until the end of the game. The performance with all sensors enabled was boosted by more than 60% compared to the Alpha release, making it playable with mostly more than 60 FPS on current mid-tier gaming PCs. This was achieved by rendering the scene only once for multiple sensors and adding the sensor-specific visualizations on top. The sensor specific visualizations were rendering using a seperate layer, making them very fast as little geometry was rendered in each sensor. Moreover, we improved the UI. The health- and detection bar are now located beneath the sensor screens as they were not recognized in the bottom left corner of the screen before. Additionally, the outline of a sensor screen starts blinking as soon as the player is being detected. However, only the corresponding screens of the sensors that are being detected by a specific enemy are blinking. As an example, if the player has all sensors activated and confronts an enemy that is detecting the night-vision sensor and the enemy-detection sensor, only these two screens will start blinking so that the player knows which sensors to switch off. Finally, we fixed several bugs we found during playtesting such as incorrect respawn behavior, save and load behavior, or missing references.

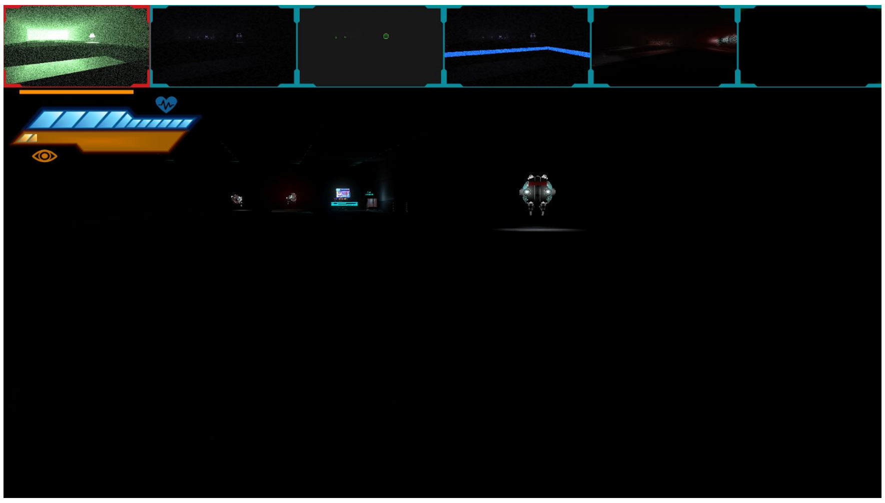

It seems ARGOS has transferred excess heat to parts of the floor. Be careful not to fry your circuits if you are moving over hot floor panels. There should be a heat sensor in one of the rooms close by.

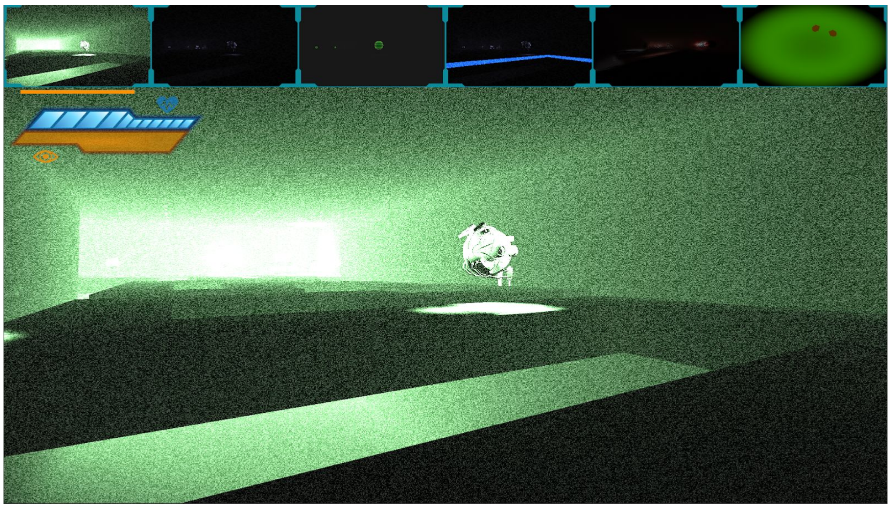
Destroyed bots give you tips and help you understand the game mechanics.



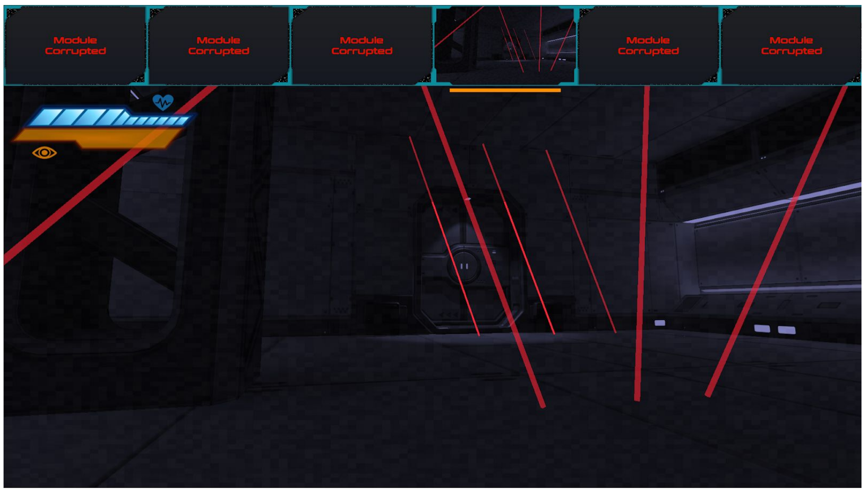


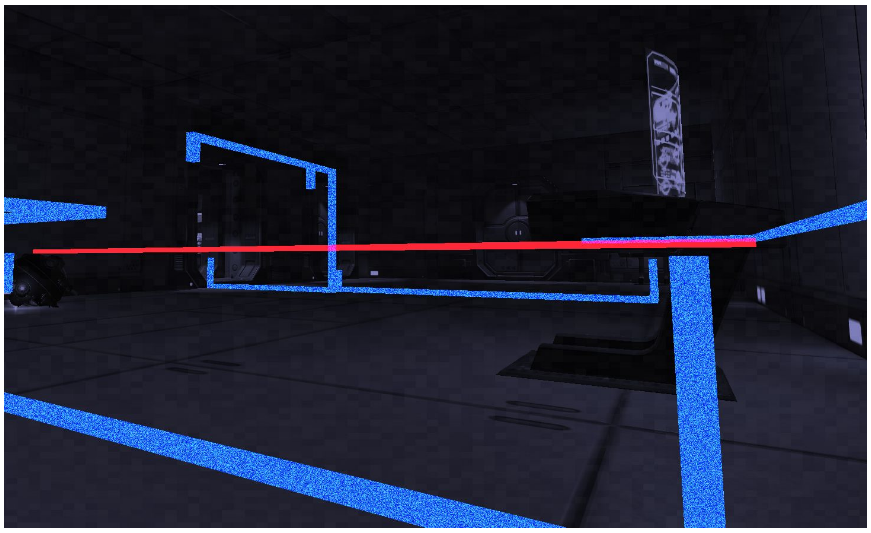
Texts describe what's happening and give additional feedback. Left: Heat Sensor.

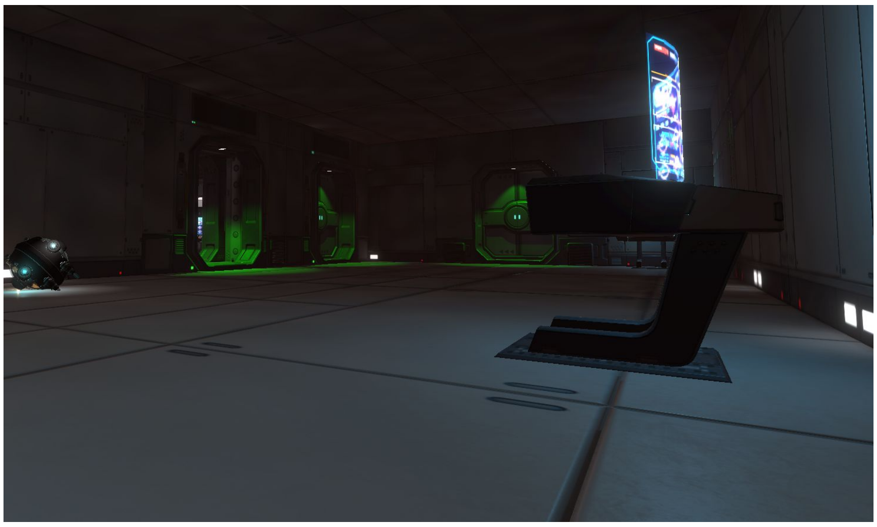




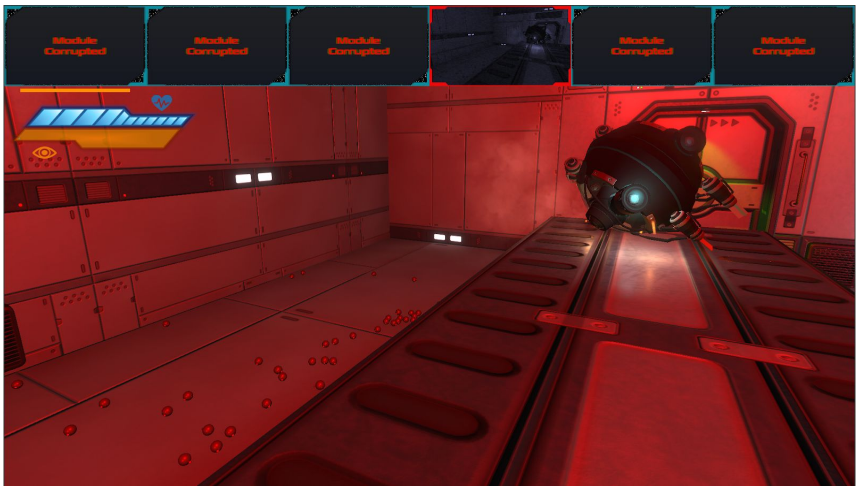
The night vision sensor lets you see in almost pitch black environments.







The Electromagnetic Field Vision enables you to see lasers (red) or cables (blue) hidden in the walls and floors that connect level elements that can interact with one another. E.g. a computer terminal that can open/close a door if hacked (bottom two images).



Red alarm lights throughout the whole levels tell you when you've been detected and are about to be "removed" by the tiny bots (left).

Our final game matches most of our initial design. We implemented every sensor we planned before, as well as certain game mechanics for the player or the enemies. The comments we received on our initial design helped a lot to improve it and to give an expression on which things we have to lay our focus on during development. One of the main challenges were the different screens on top of the screen.
Nevertheless, we were able to follow our time schedule quite precisely. Our tasks were well enough distributed so that everyone could work in parallel, without having to wait for another team member to finish a task. The development schedule was also very helpful so everyone constantly knew, what the other team members were doing at the moment and we could keep track of our progress. That is why the core game mechanics were developed relatively fast, even faster than predicted, and we could focus on level design, testing, and optimizing. Together with our prototype, the playtesting showed us what we did good, but also what has to be improved in order to create a release candidate of our game. It was a very good experience to see different sorts of people, experienced gamers, as well as non-experienced or non-game affine people, play our game. Not only did we find several bugs, but also saw how the game appears to other people than us developers. We received interesting and extremely helpful feedback that led to a great improvement of our game.
Throughout the development process, writing this document was also very helpful. For each milestone we fulfilled our todos and eventually concluded our work which helped us reflect what we have done for this milestone and helped us to keep track of our development schedule.

Performance was a big issue most of the time. The different sensor views are all powered with different shaders and simultaneously rendered at the top of the screen. Performance of this set-up has provided us with some challenges. Interestingly it was not so much the shaders itself that had to be tweaked for performance, but the multiple cameras and render textures at the top. We could however use cameras more economically and increase performance dramatically.
We found AI to be a very interesting, but also challenging theme. We had to decide whether we want to implement the theme rather in the story, or create an AI based game and develop our own AI. We decided to lay our focus on the story and create a minor enemy AI.

However the theme still led to faster decision making. If there was total freedom, it would probably have taken way longer to decide on a game design. Now, we had something we could start working with and create ideas, game mechanics, and a story. Also, a theme is not necessarily limiting one's horizon but may just help leading to the right direction.

In our next game development project we would like to start earlier with performance improvements and level design. Since In general, our teamwork worked out very well, we were able to get most things done anyways. We were lucky that for our game it was possible to separate tasks and fit them together afterwards.

For the next game project we would like more testing, performance improving, another playtesting phase after release to further improve the game

We were almost always on schedule with our tasks. Early on we had to extend some of the shader development which led to less time for sound and story elements. However the level design was also still under work during that time. In the end this meant that postponing soty integration was fine due to changes in level order and layout.
The phase between the playtesting and the release could be enlarged by another one or two weeks, so there is not only enough time to implement the changes, but also polish the game afterwards, since it should not be polished before the last changes.

We were able to create an interesting, fun and appealing game that mostly meets our expectations. With our 15 different levels we have almost twice as many levels as initially planned for our high target. From playtesting we learned that we can expect even experienced players to take more than 45 minutes to complete the whole game, which proves that our game idea and its realization is challenging enough to provide much content through different level designs. The game could be extended easily to meet the expectations of a full retail game by adding content and tweaking the performance. The multiple cameras - as expected - need more processing power than a single-camera game, but performance is fine on current mid-tier gaming PCs. We only suffer from performance issues with some of our dynamic texts that are mainly displayed in the intro scene in the first level.
We were not able to completely reach our initial high target as we were supposed to voiceover the narrative and create cutscenes to add cinematic story elements.

Final Result

Targets

Functional Minimum:	**reached**
Low Target:	**reached**
Desired Target:	**reached**
High Target:	**almost completely reached**
● Story:	No voice recording and only one cutscene at the beginning of the game.

Thank you for joining our development journey of 8th Sense!
May Argos never detect you.