Robo Instructus Art Assets

Overview

Robo Instructus is a 2D PC puzzle game where players program logic to route a robot *(and later a robot + probe)* through maze-like levels made up of interlocking triangle tiles.

The scope of the game is finished, but all the art requires concept work to flesh it out. To understand the requirements you should also observe the current version of the game in motion.

The setting is a mysterious, remote abandoned underground facility, and the scaffolds above it.

Players have 2 different views

- Level: The view of the selected level. Here you write your code solution and watch the robot/probe navigate the level, or fail to.
- Facility: Scrollable view of the facility starting with the scaffolds above. You'll be able to scroll down further as you progress. From here you'll unlock and select levels to play.

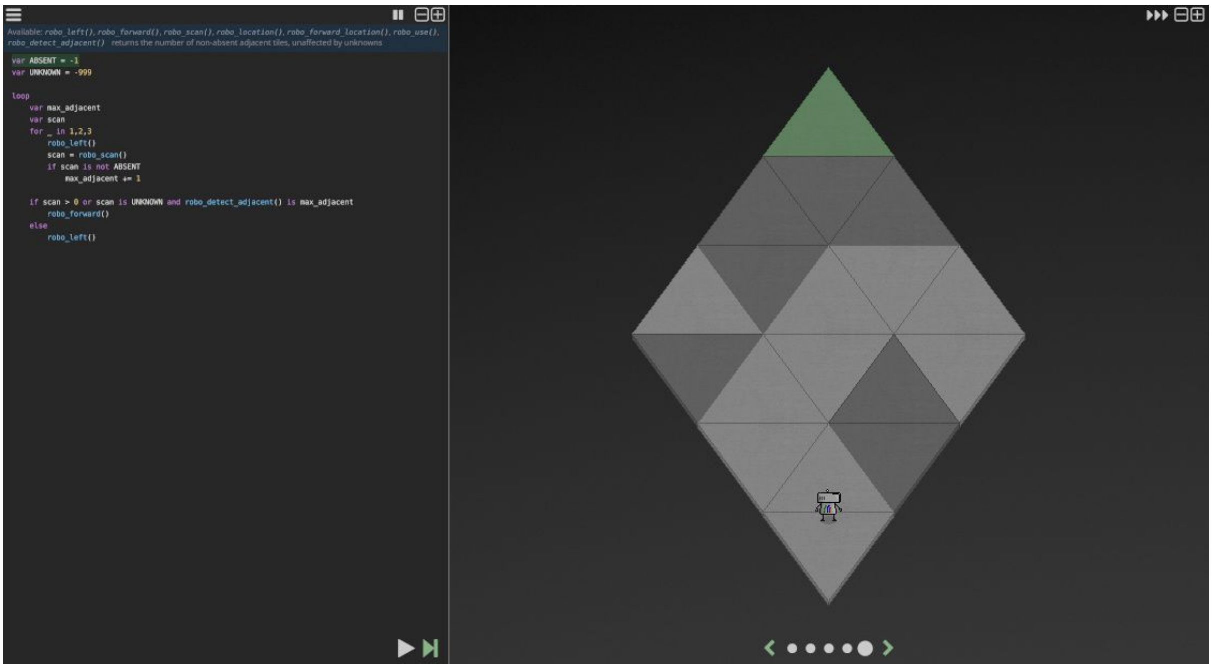
Art Format

The game does not use an engine, it's a custom built OpenGL renderer.

All assets should be high-quality pngs in future-proof resolutions. Svgs/scalable format would work too. Animations provided as frames in a grids, or let me know if there are better formats. Animations should run in a constant linear framerate *(which can be viewed significantly sped up in-game as the speed is configurable)*.

Lossless / original formats should also be provided if possible.

Level View



Available: *robo_left(), robo_forward(), robo_scan(), robo_location(), robo_forward_location(), robo_use(), robo_detect_adjacent()* returns the number of non-absent adjacent tiles, unaffected by unknowns

```
var ABSENT = -1
var UNKNOWN = -999

loop
    var max_adjacent
    var scan
    for _ in 1,2,3
        robo_left()
        scan = robo_scan()
        if scan is not ABSENT
            max_adjacent += 1

    if scan > 0 or scan is UNKNOWN and robo_detect_adjacent() is max_adjacent
        robo_forward()
    else
        robo_left()
```

This is the meat of the game, with the most art assets. On the left-hand side players type in code, run and debug their solutions. On the right you see the robot move about and scan things, and maybe fall off the level or better get to the exit (placeholder green).

Backgrounds

Levels on the right-hand-side are set in 1 of 3 different environment themes. So 3 different background arts are required.

Practically the levels themselves simply float in front of the backgrounds.

Themes

1. Scaffold: The initial puzzles are suspended on the scaffold tower above the facility. This is a cold, mountainous, snowy/icy area. You should see elements of the scaffold itself.
2. Underground near surface: Once we enter the facility the levels are set underground. Near the surface light can still make it into the facility from cracks in the ceiling. Sheltered from the surface cold, some plants grow among the rust & concrete.

   This background is rocky with shafts of light and some signs of facility/lab tech.

3. Deep underground: Later we're deep enough in the facility that no plants or surface light can make it down. The facility starts to look less dilapidated and artificially lit.

   This background should appear a little more high-tech than the last.

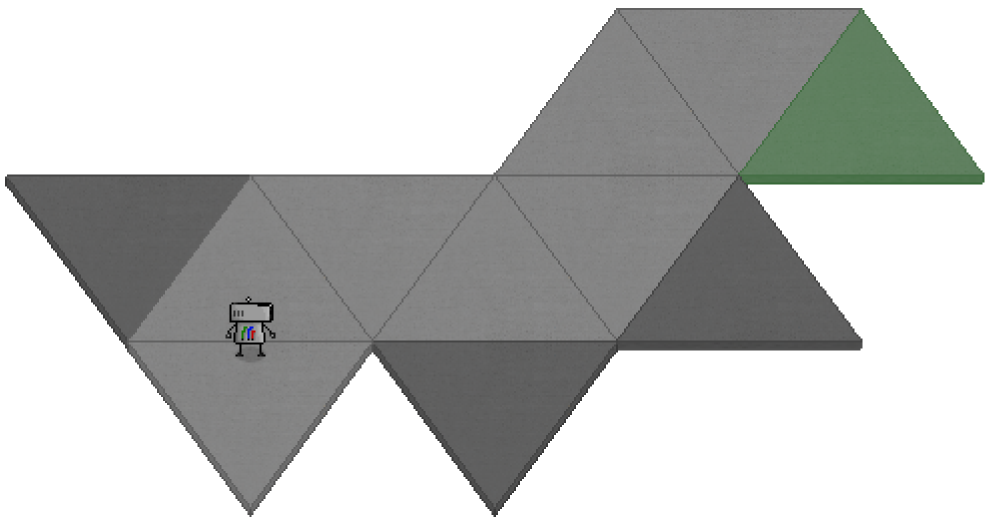
Effects

Some environmental effect concepts:

1. Scaffold: Simple wind & light snow effects to demonstrate the outside cold conditions.
2. Underground near surface: Dust and bits from the surface cracks lazily floating in the air.

Triangle Tiles

Every level in the game is made up of triangle tiles locked together in an isometric view. There are a few variants of tile, currently placeholder colour coded for play-testing.

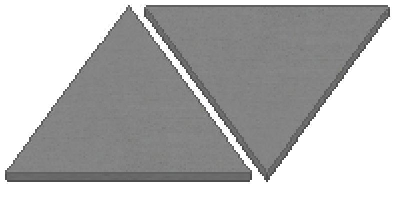


It's important that the type of tile is obvious from a glance at the level.

Tile types

Each type, unless stated otherwise, can appear in a "point-up" or "point-down" triangle form.

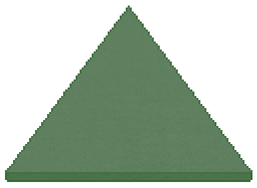
1. Regular tiles: Concrete surfaces that are safe for the robot to walk on. These are the bulk of the tiles in the game.



2. Starting tile: Where the robot starts each level. Wires in the middle of the tile connect to the robot and will detach once the robot starts to move. The robot always starts facing north. Otherwise similar to regular tiles.

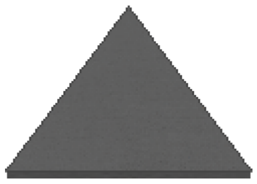
No placeholder art, currently just looks the same as a regular tile.

3. Exit teleporter: Getting here is the objective of each level, allowing further progress deeper into the facility. *(In some cases there are multiple exits).*



So these tiles should contain a teleportation platform, or glowing portal of some kind.

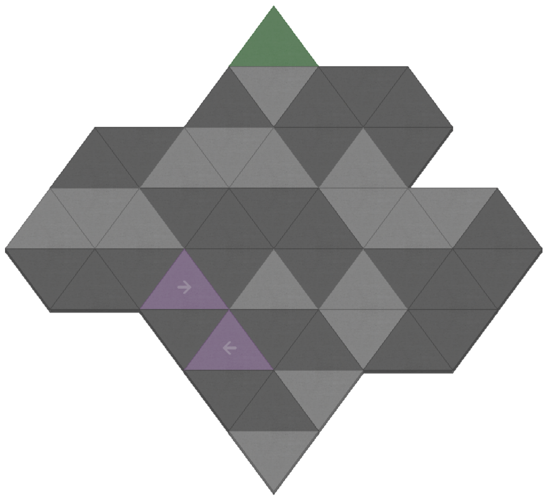
4. Unknown tile: This is an area where we don't know if there is an actual surface or if it's a hole. These crop up a lot in levels.



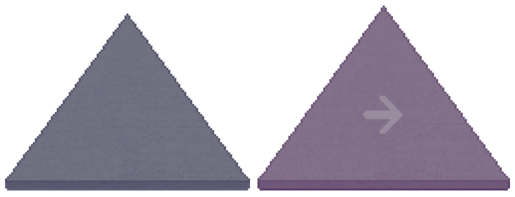
This could look ephemeral. Or be a kind of "fog of war" over these areas.

Currently just coloured darker, open to ideas here but the clarity of where they are should remain.

For example a later level with lots of "unknowns":



5. Data points: Regular tiles that also contain some data the robot can read.

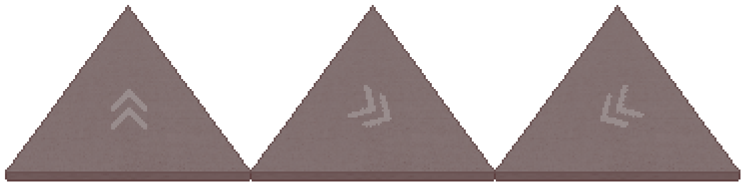


There are 2 of these, one provides location ID information. The other provides direction information.

They should look fairly similar to regular tiles, as other than the data they contain they act the same way as regulars. Some sort of interface in the middle that the robot can connect to from any side.

The two kinds don't need to look too dissimilar from each other. For the location-id data points the player should simply be able to tell what it is. The direction ones should also make clear a single direction (up, down, left, right).

6. Launcher tiles: These are tiles that can launch the robot into the air and in a direction.

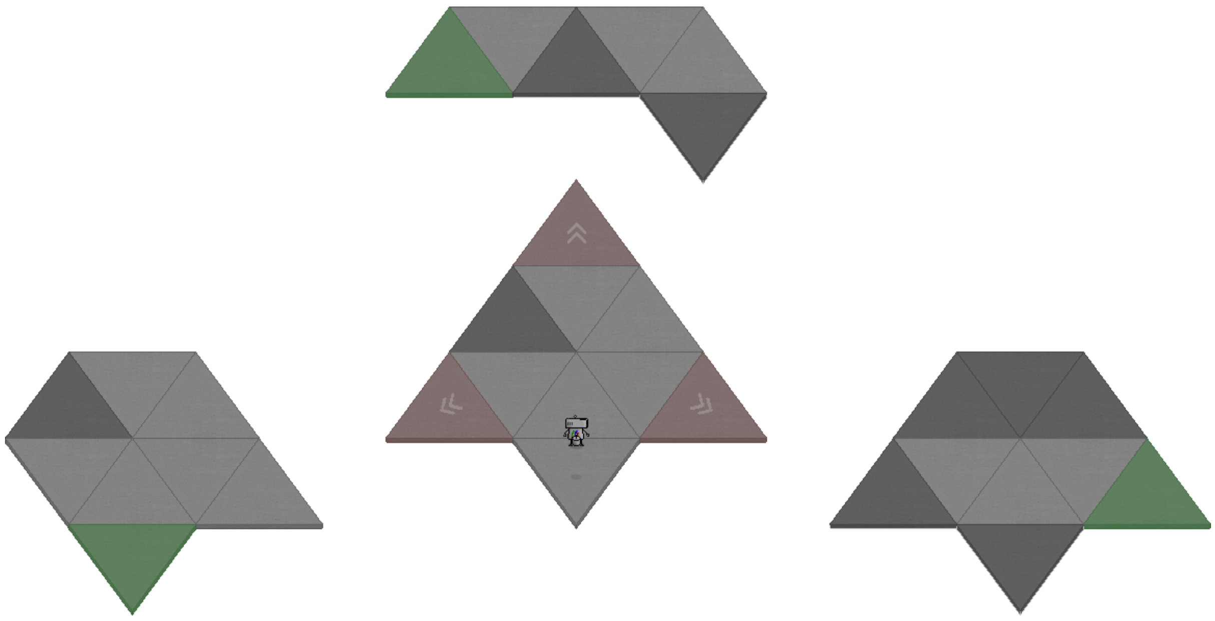


These, unlike the others, **can only be "point-up" triangles**.

They charge up while the robot stands on them, after a while they launch the robot. They should show their direction (as the placeholder does). But also show how much they've charged and some effect when the launch the robot.

The charge could be just lighting up, or lights changing colour.

Example level with launchers:



Tile thematic variations

The game has 3 different puzzle environments providing variation to the look. So each tile needs versions in line with the theme.

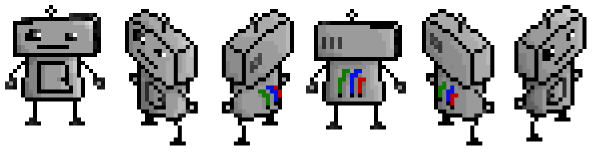
1. Scaffold: On the scaffold tower tile appear icy & weathered
2. Underground near surface: Plant life grows in the cracks, metal is rusted and concrete is somewhat decayed.
3. Deep underground: Concrete is in better shape & less rust.

Actors / Characters

There are just 2 actors that can move about the level. They are both programmable by the player.

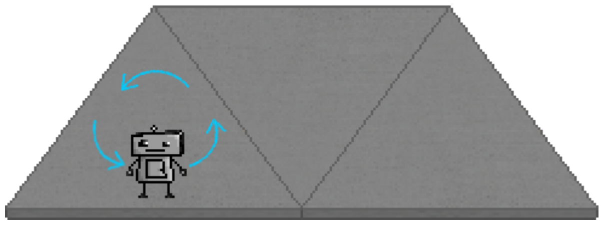
Robot

The main character & game mascot that players program and moves around the levels.

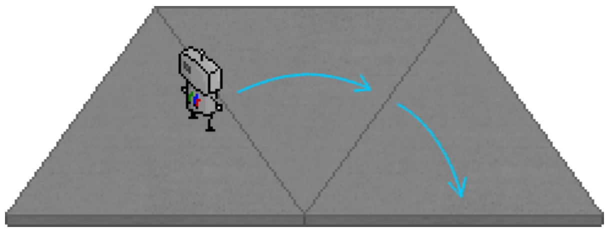


Moves to and from near the flat edges of the triangles so rests facing 1 of 6 directions.

Moves about levels according to the player program:
- ***robo_left()*** move leftwards from one side to another



- ***robo_forward()*** move forward (and rightwards) to the tile it's facing



Can be "launched" by the launcher tiles.

Concepts/constraints

- Should be fairly "charming"
- In general we can follow the placeholder concept, or do something totally different.
- Wires at the back to connect to the starting tile.
- Should be grounded, ie physically fall if moving over an edge to an absent tile.
- Should have some sort of antenna, as will communicate with that later

Since the game can be sped up and slowed down all animations and effects should bear this in mind and look ok at the slowest speed upwards. Make sure to view the game in motion to help see what's needed.

Animations

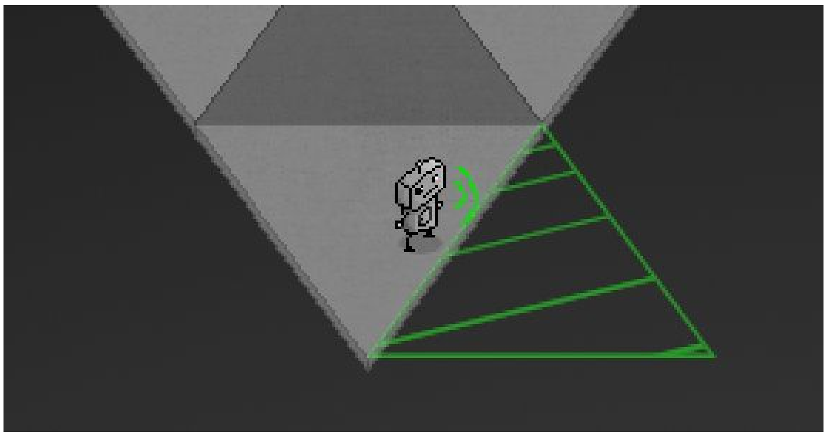
- Moving from side to side leftward of either type of triangle-tile.
- Moving forward (and rightwards) from one triangle-tile to another.
- Moving (either left or forward) and disconnecting from the starting tile
- Falling off an edge of a tile
- Teleporting in & away
- Being launched in 1 of 3 directions (a distance of ~2 tiles)

Function effects

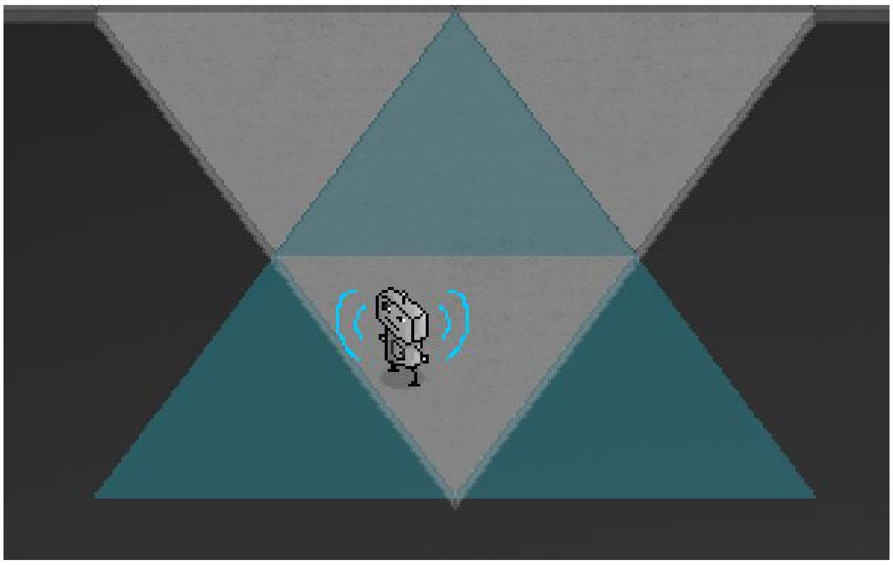
To allow the robot to navigate through the levels it has functions that return information generally about stuff near the robot.

When the robot is doing such a function it will not be moving. An effect highlights the tiles that are affected, another animation/effect shows that the robot is doing something.

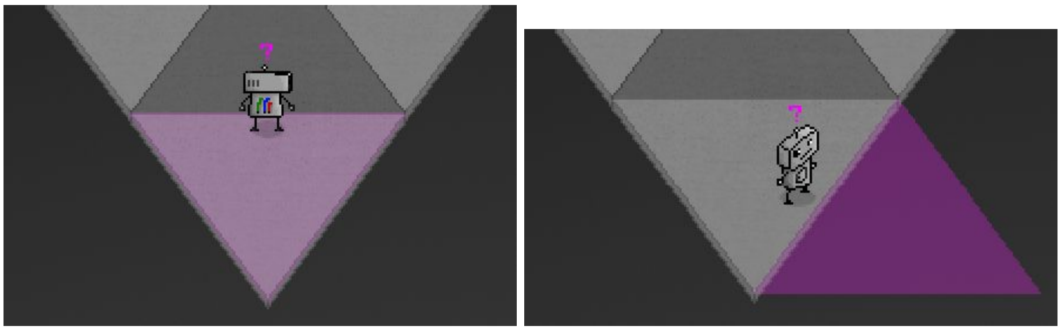
- ***robo_scan()*** scan the tile in front



- ***robo_detect_adjacent()*** detects adjacent tiles

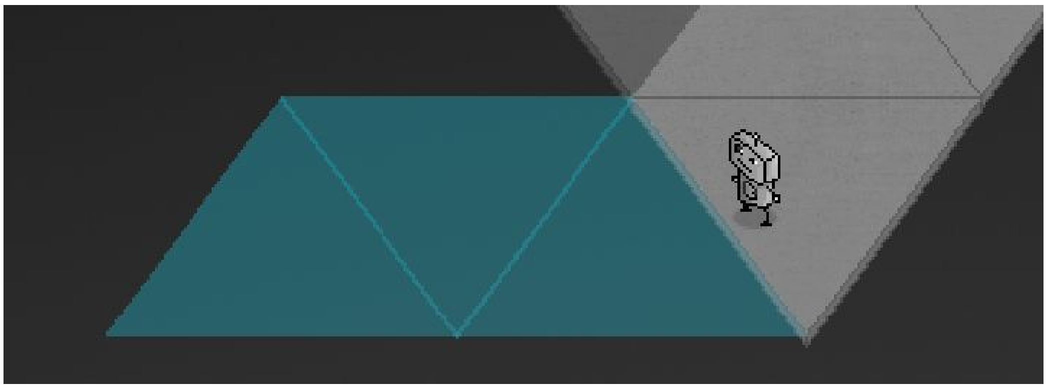


- ***robo_location()*** gets current tile id
  robo_forward_location() tile id of tile in front



These functions are both very quick.

- ***robo_detect_3()***, ***robo_detect_3l()***

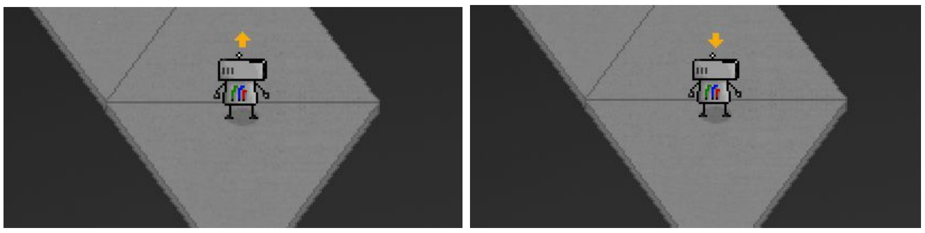
  - Should look similar to ***robo_detect_adjacent()***

- ***robo_use()*** Access "data point" data, or not if the current tile is not a data point.

  Here should connect with the data point somehow. If not on a data point do an animation of "trying" to connect.

  Consider that the probe also needs to be able to do this, so should avoid a physical interaction with the data point.

- ***transmit*** & ***receive*** communicates with probe

  Should show if sending or receiving near the robot antenna.

  These functions are both very quick.

Probe

In the final 3rd of the game a probe also becomes available, and players must program both the probe and the robot.

The probe hovers off the ground, so cannot fall, collide with the robot or be launched. Similarly to the robot it faces 1 of 6 direction, and can move left or forward. However, it sits more in the middle of the tiles rather than the edges where the robot is.

Unlike the placeholder art you *should* be able to see which direction the probe is facing.

Concepts/constraints

- Should hover
- Should look like similar tech to the robot

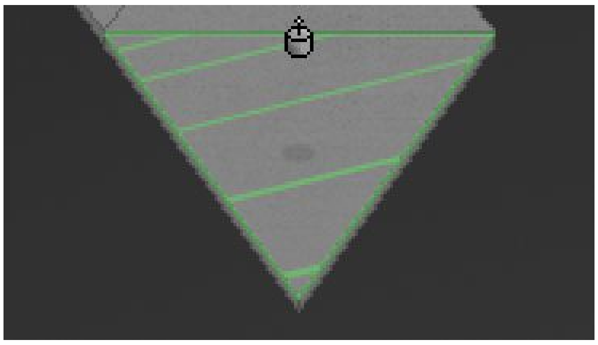
- Should have some sort of antenna, as will communicate with that later

Moves about levels according to the player program:
- ***probo_left()*** rotate on the spot leftwards to face the next side
- ***probo_forward()*** move forward (and rightwards) to the middle of the tile it's facing

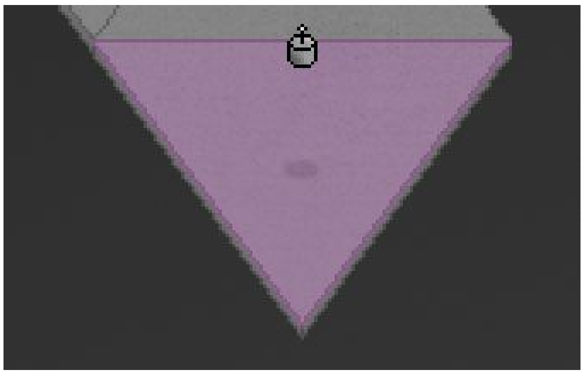
Animations
- Movement left or forward
- Teleporting in & away

Function effects
The probe has access to somewhat similar functions to the robot.

- ***probo_scan()*** scans current tile
  Similar to *robo_scan()*

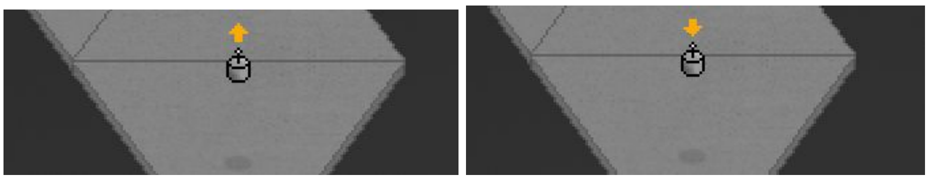


- ***probo_location()*** gets current tile id
  Similar to *robo_location()*



- **probe_use()** Access "data point" data, or not if the current tile is not a data point.
  Similar to *robo_use()*

- ***transmit*** & ***receive*** communicates with robot
  Same as robot functions.



Facility View

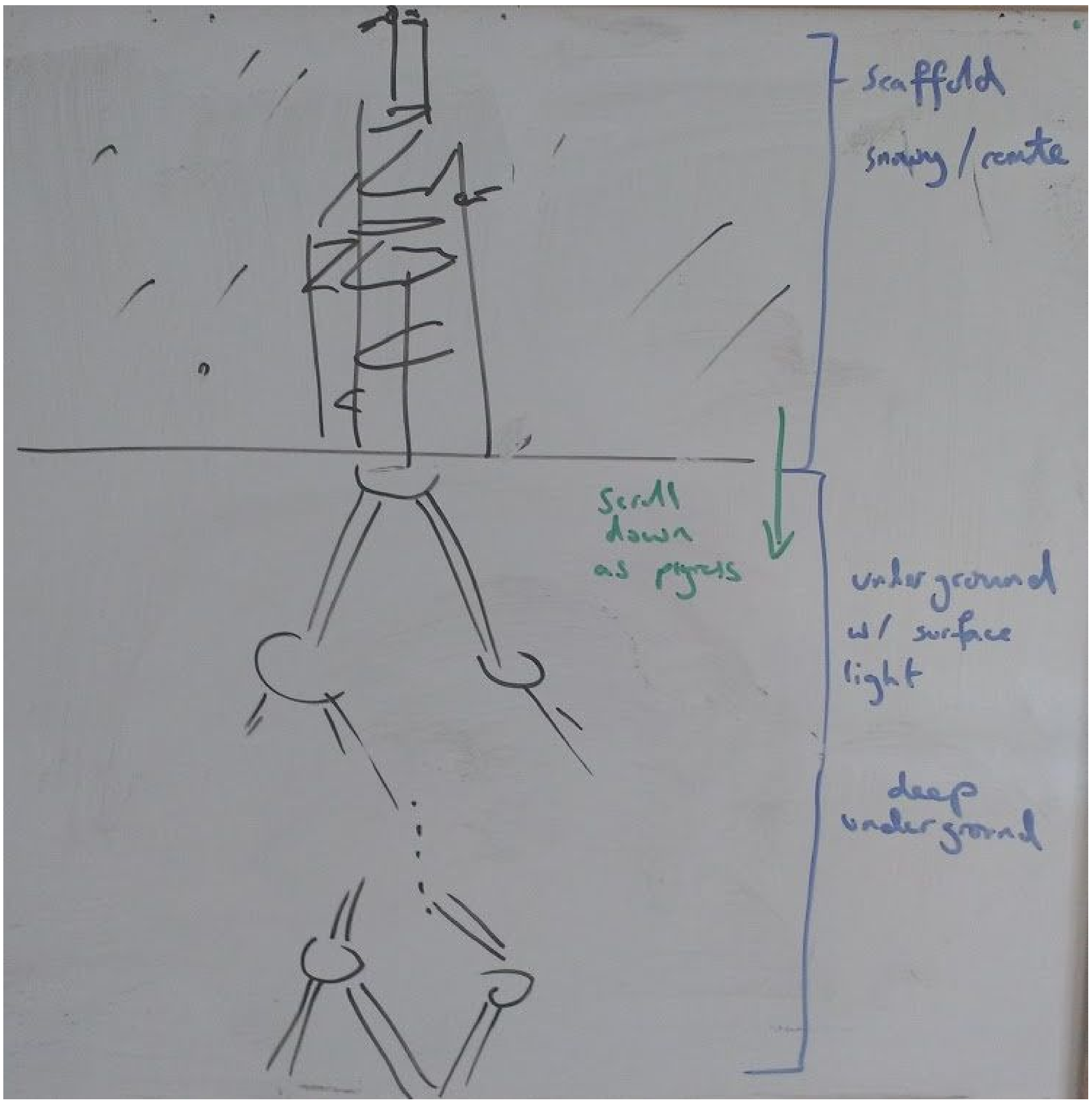
This is where you select levels to play.

The art is one long scrollable background taking up the whole screen with some environment effects.

Rendered over the background will be clickable level nodes and information I'll implement myself with simple geometric shapes for level nodes and lines between them.

Initially players can see only the top part of the scaffold tower, and can select the first level. As players complete levels the view will scroll downwards to the facility entrance then steadily underground until the final level can be selected.

Sketch:



As you progress you'll go through the 3 thematic zones *scaffold tower, underground near surface, deep underground.* These should be visible here too (though underground / deep underground is more subtle in the wide view).

Resolution

This view will take up the whole screen, and should repeat in all directions to allow large screen resolution support.

Effects

Similar to the level background effects we could have wind/snow effects above the surface & dust light beams below to provide a little movement to the scene.

Level selection nodes

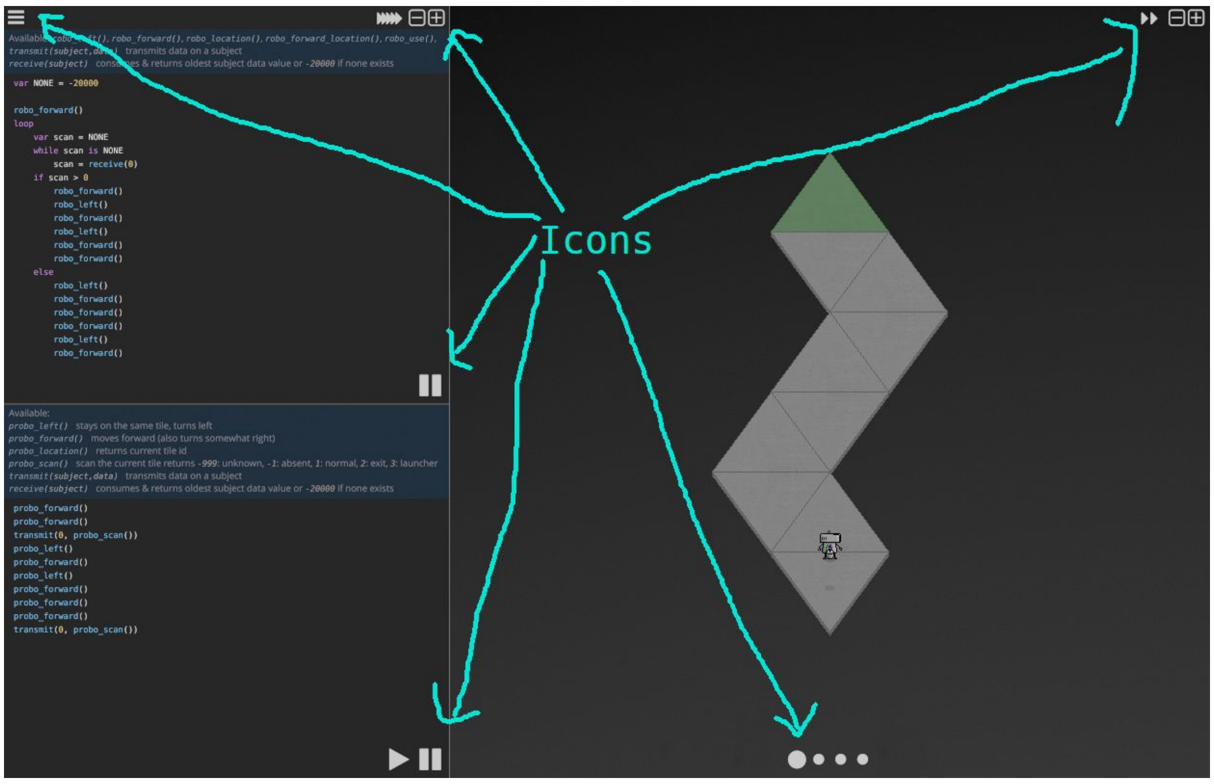
On this screen you select levels to play. The nodes you click on will be floating on the view background with lines placed afterwards dynamically.

I want the level nodes to be separate from the background so that adding / re-arranging levels can be done without reworking the background art.

UI

No specific work is planned for the UI.

However, some icons may be better replaced depending on how they blend with the final art.



Available: robo_left(), robo_forward(), robo_location(), robo_forward_location(), robo_use(),
transmit(subject,data) transmits data on a subject
receive(subject) consumes & returns oldest subject data value or -20000 if none exists

```
var NONE = -20000

robo_forward()
loop
    var scan = NONE
    while scan is NONE
        scan = receive(0)
    if scan > 0
        robo_forward()
        robo_left()
        robo_forward()
        robo_left()
        robo_forward()
        robo_forward()
    else
        robo_left()
        robo_forward()
        robo_forward()
        robo_forward()
        robo_left()
        robo_forward()
```

Available:
probo_left() stays on the same tile, turns left
probo_forward() moves forward (also turns somewhat right)
probo_location() returns current tile id
probo_scan() scan the current tile returns -999: unknown, -1: absent, 1: normal, 2: exit, 3: launcher
transmit(subject,data) transmits data on a subject
receive(subject) consumes & returns oldest subject data value or -20000 if none exists

```
probo_forward()
probo_forward()
transmit(0, probo_scan())
probo_left()
probo_forward()
probo_left()
probo_forward()
probo_forward()
probo_forward()
transmit(0, probo_scan())
```

I'd be interested in suggestions on improving the UI / blending with final art while keeping it simple.