Game Concept

What are the basic game concepts of our game “Purpose and Despair”

General

- 1st person
- Puzzle Game
- No dialogs
- No encounters with NPCs
- Intro = drop on the planet with no explanation of who you really are
- Target = build a rocket to get to an other planet because there is no butter to push (and that is his purpose)

Player

Rick and Morty’s robot passing over the butter –> The player doesn’t know that.

Setting

People are highly developed and have learned how space travel is possible.

Story

A small robot is thrown away by its owners and comes via “garbage collection” to a planet which is used as a garbage dump site where only garbage/scrap is lying around. The initial goal of the robot is to leave the planet to show the owners that it is not so useless. He has to build a rocket. At the beginning he can only push and pull things. He now walks through the levels and encounters currently unsolvable problems which he can only solve by learning other things. The learning consists of two parts, on the one hand the robot has to find himself through the world battles and new parts for himself, these parts he can

build on himself. In order to be able to use the parts however its AI must be extended, this happens with the help of “manuals” which must be found in the game world. Once a manual has been found, a puzzle can be started which, if successfully completed, will unlock the upgrade for the robot and learn a new skill. He must now venture through the tricky levels to find parts for himself and the rocket to build them. In the end he flies away from the planet and the end.

Puzzle

Card puzzle

On the map you can find robots, rocket parts and manuals. These are hidden by puzzles that have to be solved. These puzzles can be for example:

- A way has to be found to get to a place with parts, to do this objects have to be moved which operate switches and buttons.
- Large things have to be moved with the help of a lever, for example.
- open doors

User manuals

The manuals contain riddles that are 2D puzzles printed on the manual. When the robot has solved the puzzle his AI develops further and he has understood how it solves a piece. The puzzles have the following structure:

- Everything is in a grid
- You have to close a circuit and supply all objects with power.
- There is one power source and several objects per level.
- Between these objects are fields with wires that you can rotate.
- By turning the wires you have to close the power grid.

Progress

Find new parts and learn the appropriate skill to build a rocket.

Basic skills

- pushing
- pull

learnable skills

- lifting
- Place and activate the lever
- be able to climb small steps
- Use ladders from e.g. old construction machines
- maybe use switch on power and supply “circuit diagrams” with power
- …

Map

A central place where he can upgrade the robot and build the rocket. It is a large map where there are certain areas and different parts to find. In the background you can see mostly scrap and garbage, but there are also Easteregg in the background that reveal the original owner. (e.g. Hint: “You pass butter” on sign in the background or mirror where he actually sees his look)

Time invested

Hermann 2h Tim 2h

Written on March 1, 2019

First Game Prototype

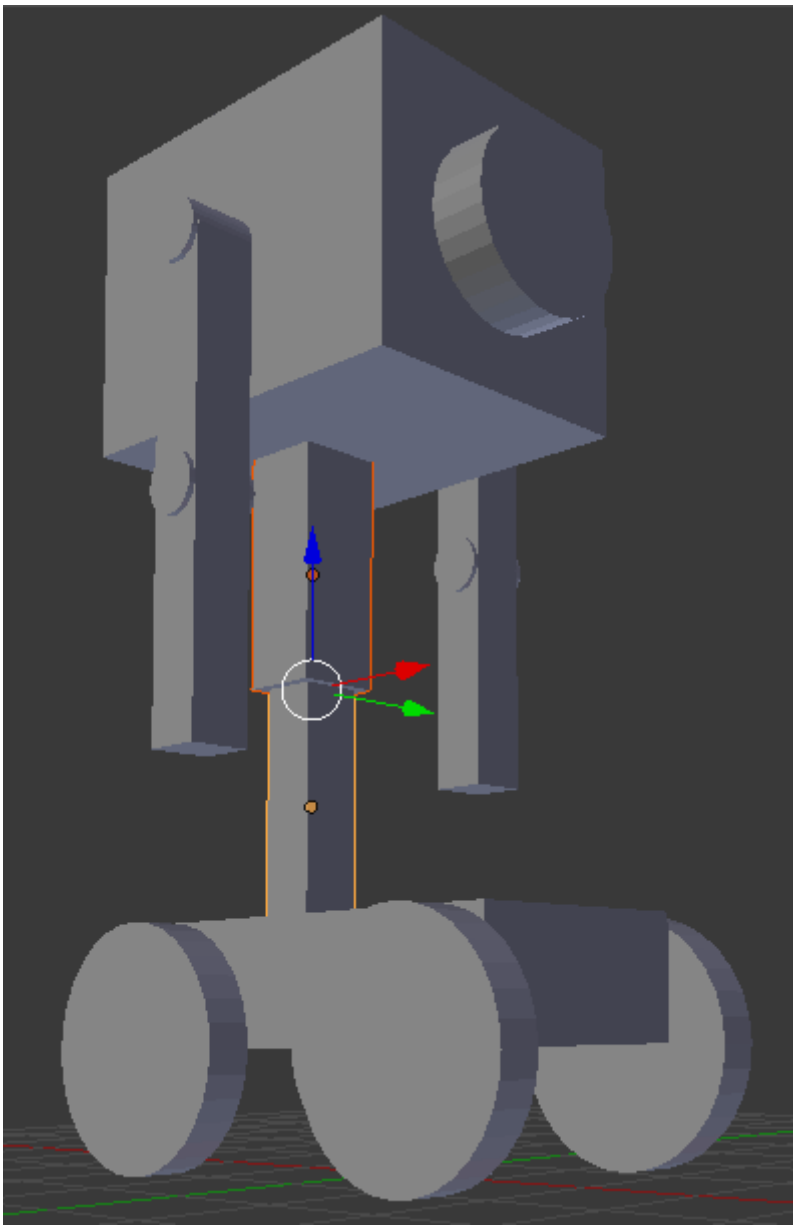
A prototype of the character and it’s movement.

General

The game will be build with Unity3D. This will allow us to use default functions from Unity like physics and light.

Character

The player character is a robot with wheels to traverse terrain. In the first model seen in the picture below, he consists of four wheels, a body, head, and two arms. The head has a big eye which will be the place of the first person camera of the player.



The arms seen in the picture might not initially be on the character when you start the game. Rather it will be one of the first upgrades you can get.

Movement

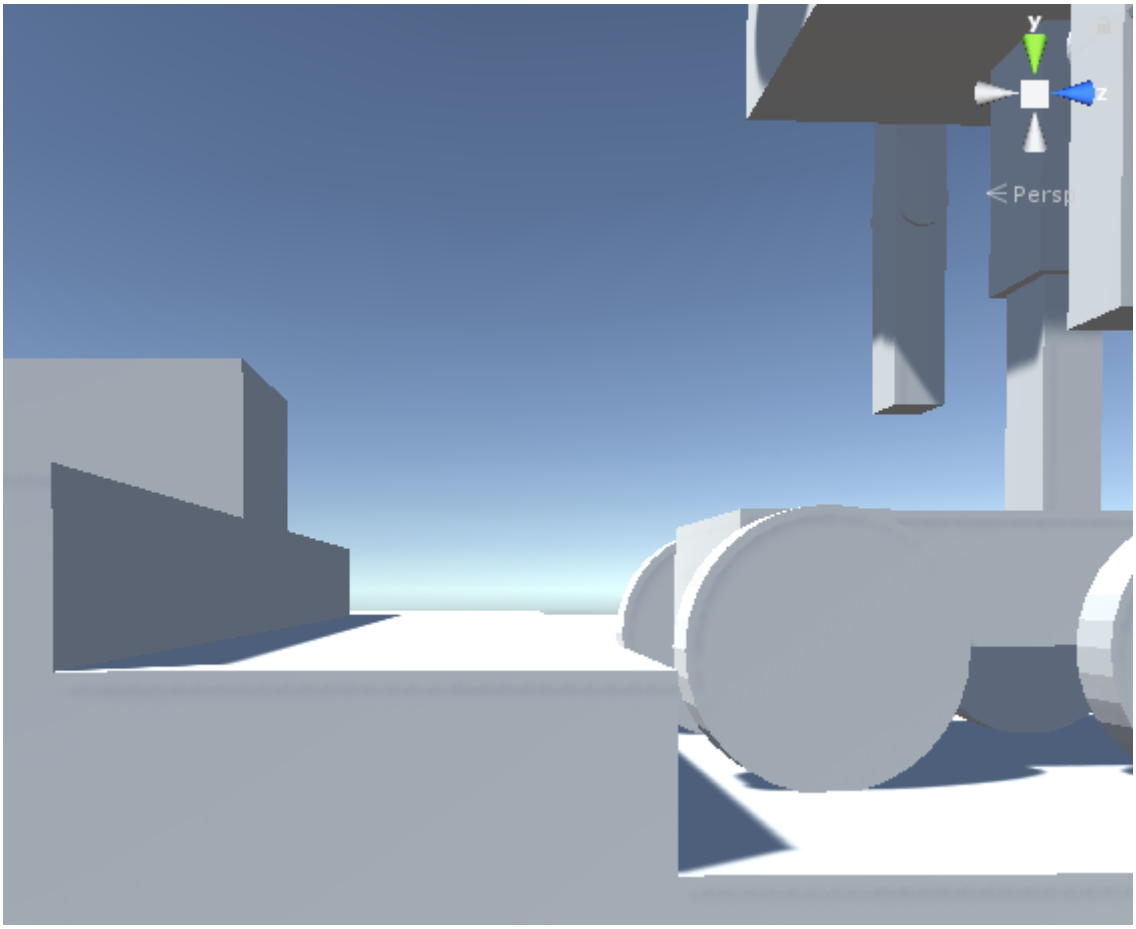
The movement of the character consists of simple forwards and backwards movement combined with rotation. It can be controlled with the arrow keys.

key	movement
up	forwards
down	backwards
left	rotate left
right	rotate right

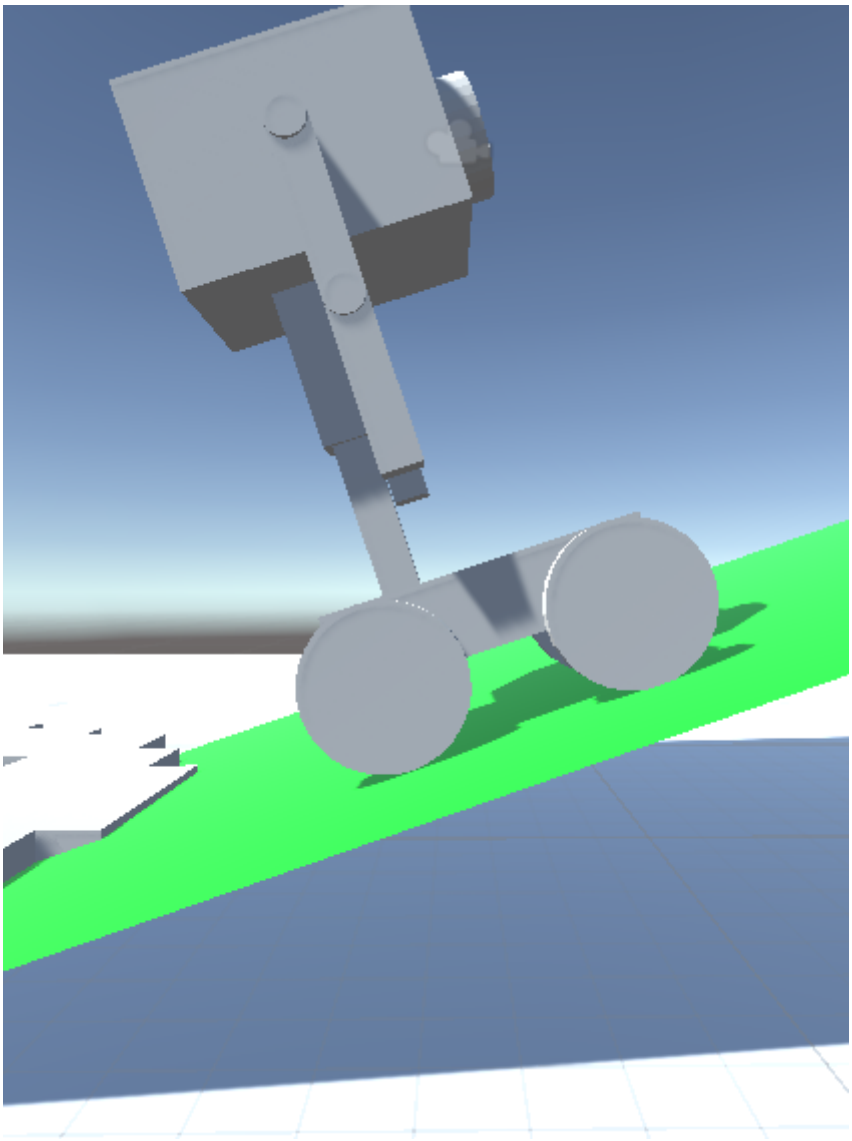
The camera is static and can’t be moved independently of the body.

The player is affected by gravity and will fall. Unlike in other games there is no fall damage since the player can’t die anyways.

The player is able to drive around the world and will automatically perform certain actions. If the robot hits a small wall, this will be seen as a step and the robot will move over this step. This ability is limited by the height of the step, he can only move up a step that is about half of the size of his wheels. The picture below shows steps which the player is able to climb.

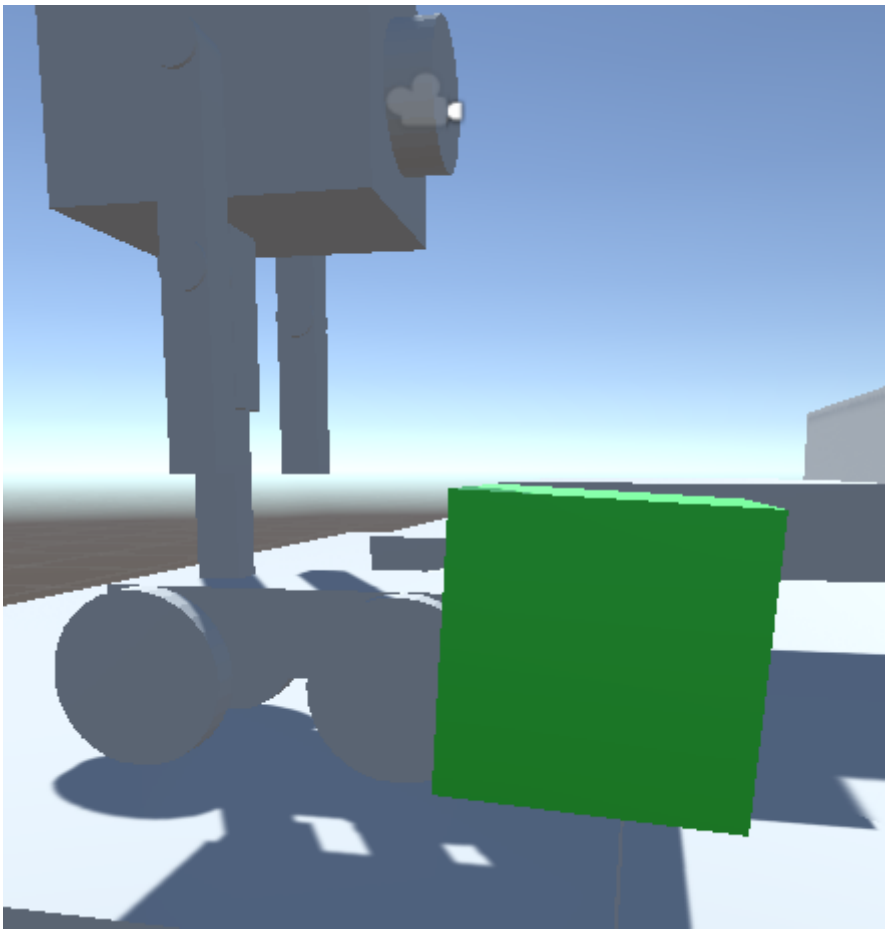


The player is also able to drive on slopes. This will tilt the character. Currently the only limitation of the slope is given by gravity which pulls the player down if the slope is to steep.



As stated in the blog post about the game concepts, the player will be able to push objects from the start of the game.This is realized through the collision with an pushable object. The object can be pushed around and is affected by normal physics at all time. This also mean he can push it down stairs or on slopes but he can’t move it up stairs.

Even though the player is able to push the object while moving forwards and backwards, he will only see the object if he is pushing it forwards, since the camera will always show the area in front of him.



Time invested

Tim 3h

Written on March 3, 2019

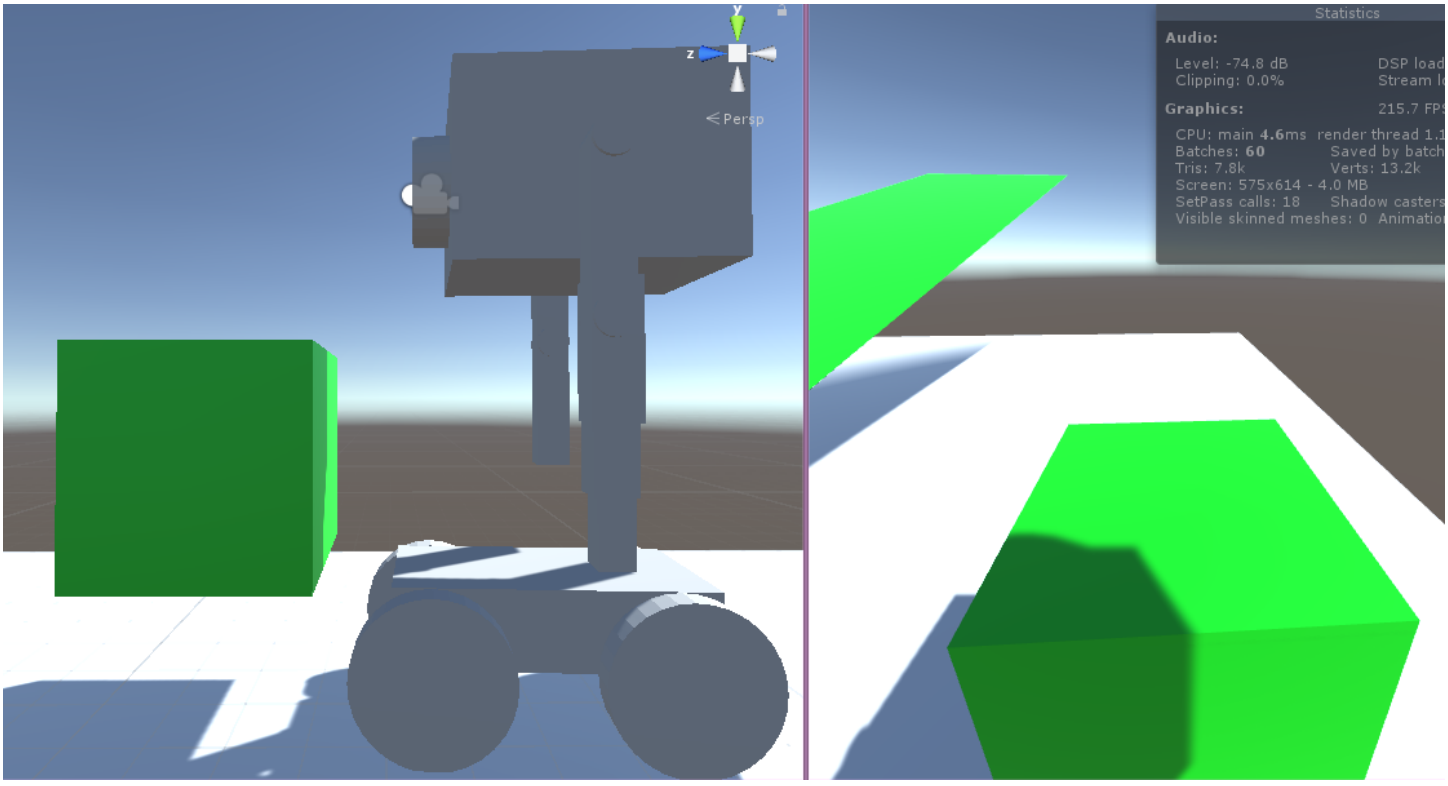
Environment interaction

Option for the player to interact with the environment.

Pickup of objects

The player will be able to unlock new skills and abilities. One such ability is to lift a object and carry it around. Such an object can be used to solve riddles and activate buttons or such.

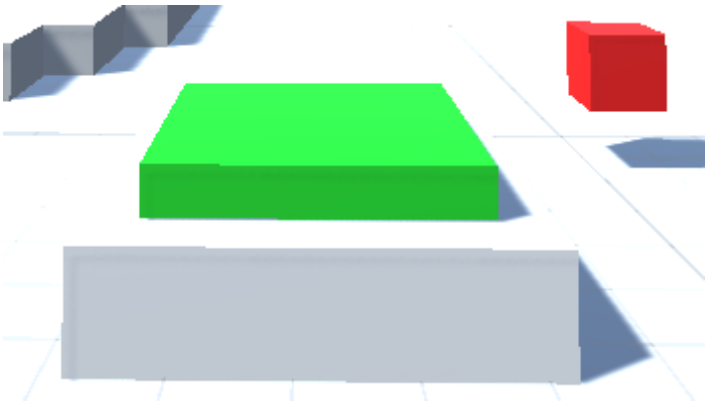
When the player is close to an object tat is pickubable he can click the “E” key and and pick it up. He can carry ot around and drop it when another press of the “E” key.



On the Right you can see the player from a third perspective and on the left you can see the view of the player.

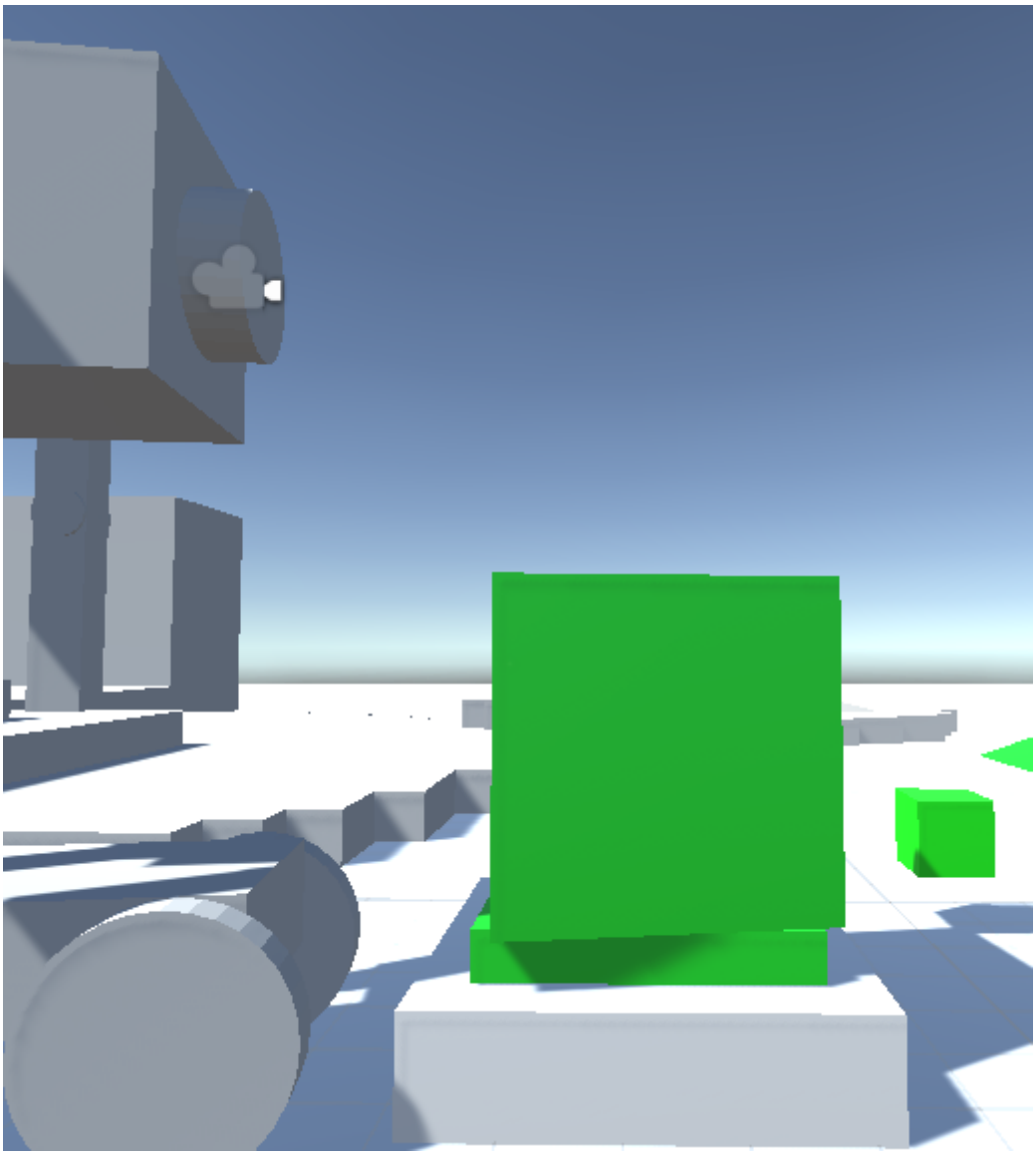
Button

The pickupable objects can be used to activate a button the red cube next to it show that the button is not pressed.



The button consists of a base and a top that can be activated. The button will change into an activated state if and object is on top of it. This can also be the player itself.

In the image below you can see the button is activated by an object, seen in the green color of the cube which changes color accordingly to the state of the button.



Time invested

Tim 2h

Written on March 7, 2019

NetWalk - fix the circuit, fool!

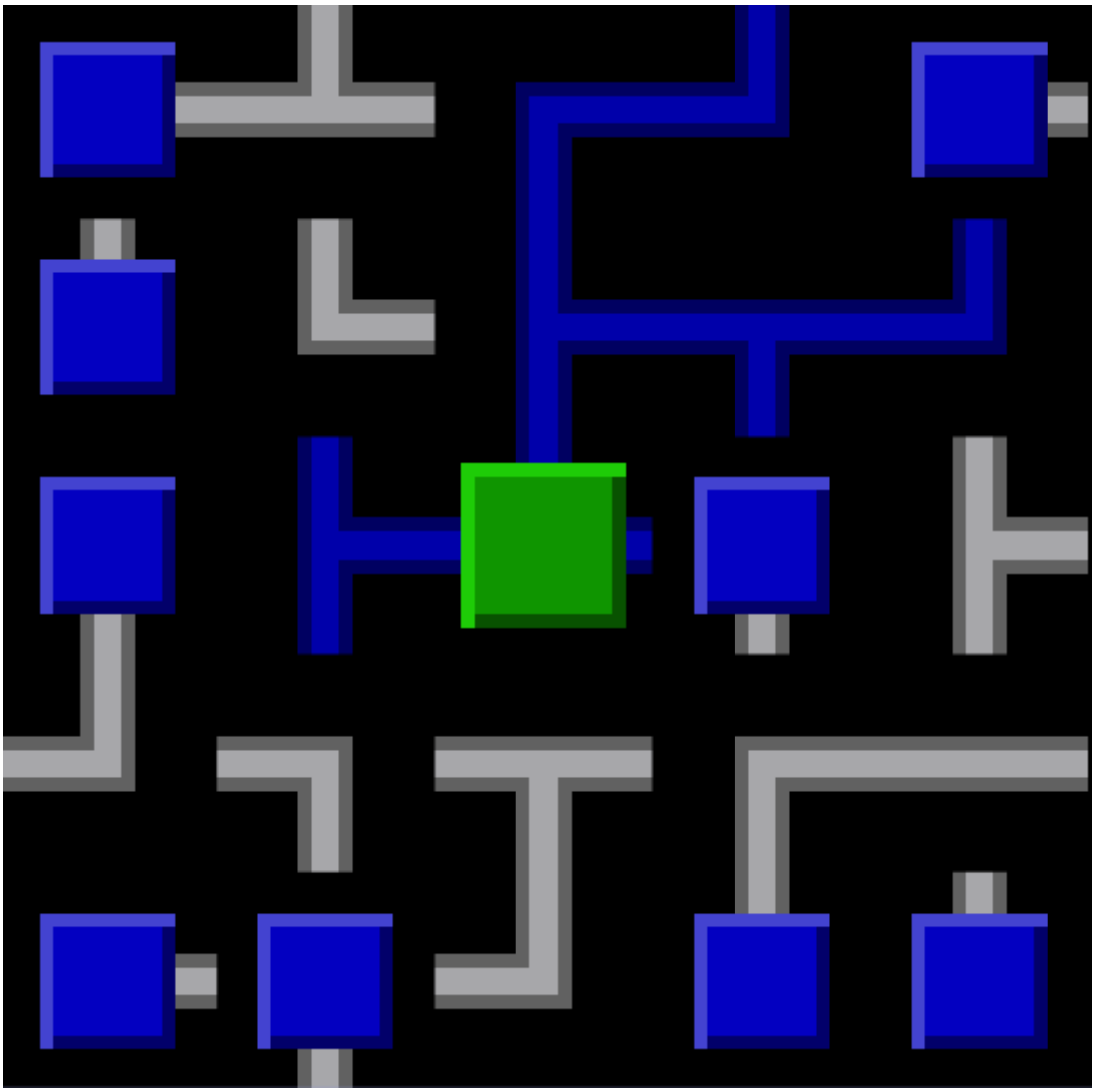
Introducing our 2D puzzle, which the robot needs to solve in order to upgrade itself.

Why?

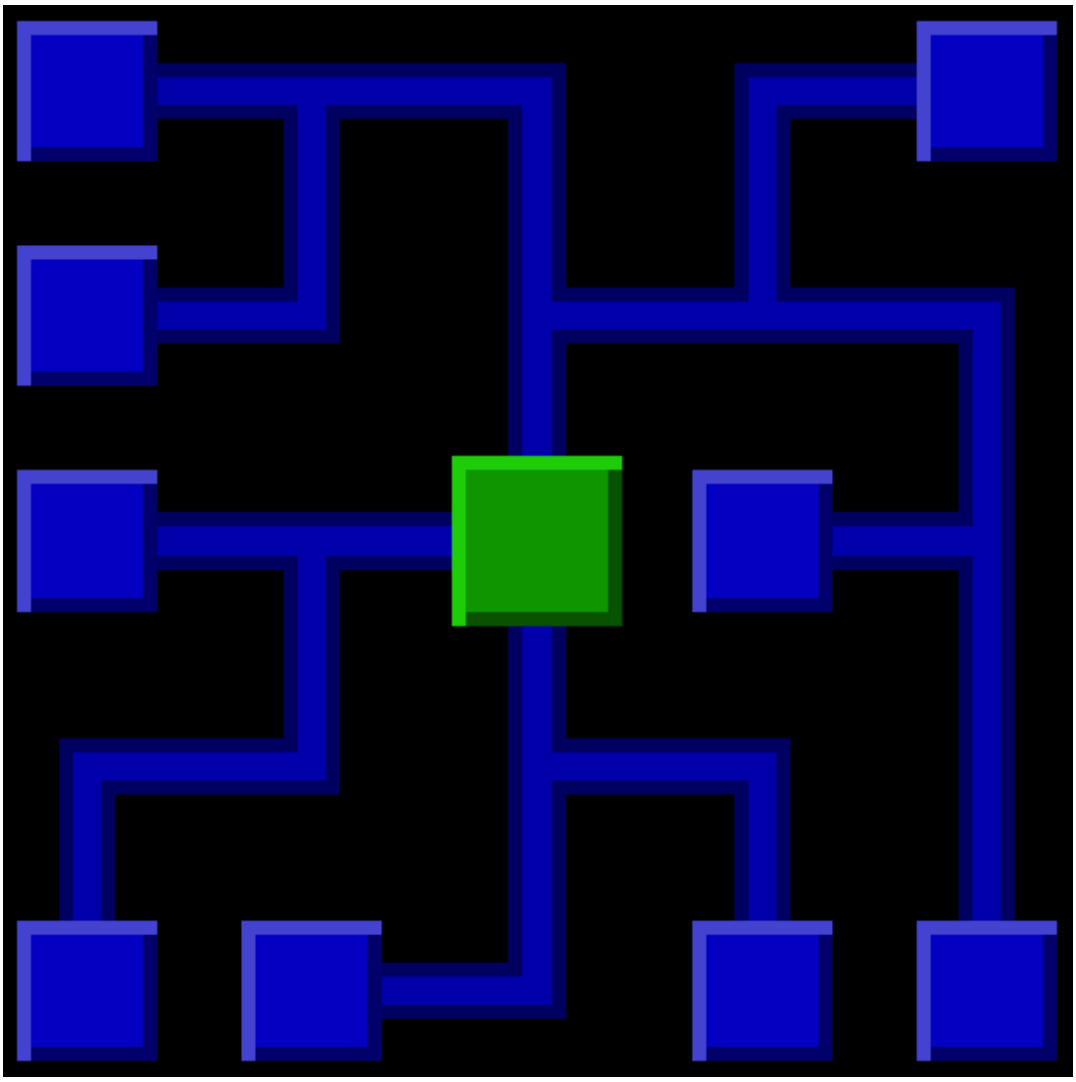
In order to be able to repair a rocket and escape from the garbage planet, our little robot needs to upgrade itself. This involves finding parts and training its brain to put these parts together. For this brain-training a puzzle needs to be solved, one that requires the player to basically *connect* all the dots; all pieces are already there, they just need to be put in the right order for it to *click*.

What?

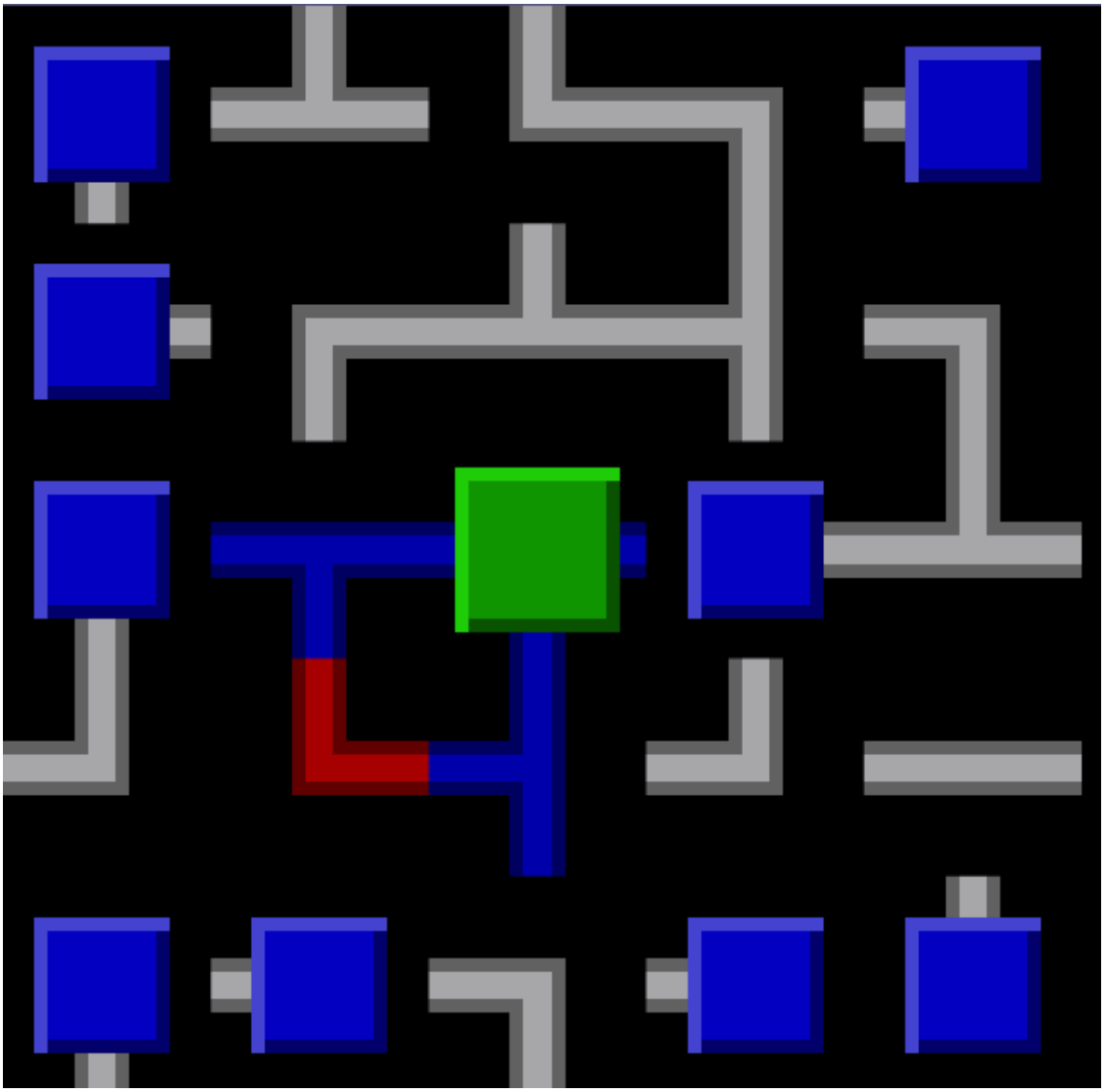
The puzzle to be solved is a game called “NetWalk” originally developed for Windows 95. You have a grid consisting of wires, terminals (blue square) and servers (green). Power runs from serves through the wires to terminals. The goal is to have power flowing through each terminal and each wire without creating cycles. In order to achieve this, the player can rotate elements. So at first the player is confronted with something like this:



A real mess of a network, there are some wires powered as random decides to rotate them in this way. With a bit of time and thinking the player can then produce the one unique solution which looks a little something like this:



But pay attention! Don’t short your circuit by creating cycles:



This isn’t healthy and won’t solve the puzzle!

How?

The game in its current state (at the time of this post) can generate any size of puzzle with one or more servers in the puzzle, the player cannot yet play as their only interaction - rotation pieces is not yet implemented. It is planned to be put as panel somewhere in the 3D world of the rest of PurposeAndDespair.

Game logic

The game logic is implemented as completly separated piece of code, it can be ported to any C# platform and thus one could create e.g. daily puzzle for all players as challenge.

The generation algorithm creates tile for tile a spanning tree by placing servers and then randomly selecting a new place for a branch to grow on. It is guaranteed to never have

tiles that are without any connection or with connections to all side (as they could not be rotated)

Unity integration

The current unity integration uses a tilemap renderer in an offscreen environment. Thus the game should not lag disregarding how big the puzzle gets. A small script converts the NetGame tile types and their power status to Unity tiles consisting of a sprite (out of a small sprite sheet) and its transform. Lattern can be used to animate the rotation by the player. With the camera rendering offscreen, the resulting texture can be applied to any old 3D object in the “real” world of PurposeAndDespair.

Time invested

Hermann 8h

Written on March 7, 2019

First Game Level

The first level of the game.

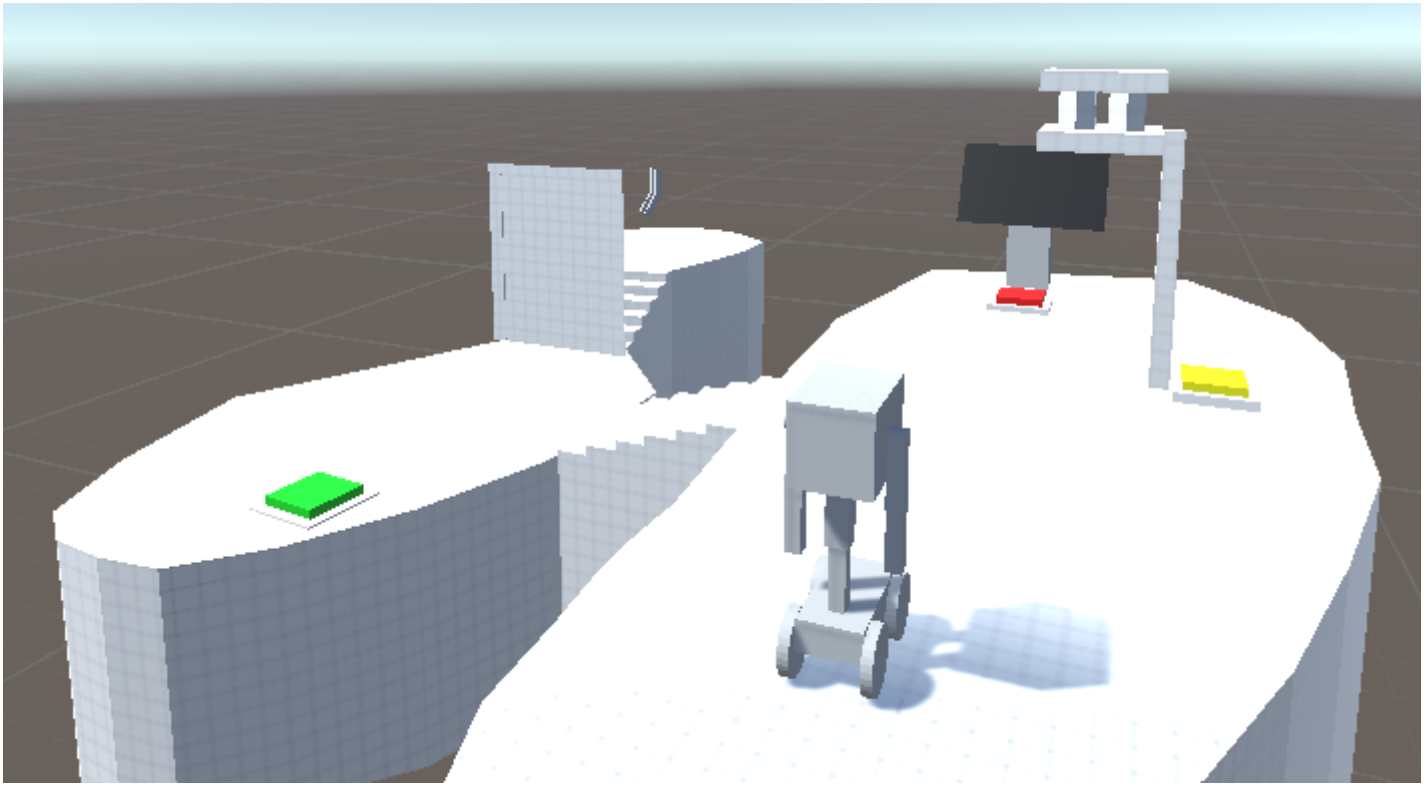
Map

The first level of Purpose and Despair will teach the player fundamental skills and interactions without telling him anything.

The player sees 3 Button which he might try to use. The green button is ready for use, a yellow button can also bea activated but will spawn a new objects that can be pushed around or picked up if the skill is learned. A red button is currently disabled and can not be used.

In the first Level the player can see a terminal which is diabled (indicated by the red button). Also a spinning object that is locked behind a door. The player now has to push the green button to open the door. In order to do the he need to put an object on it to be able to drive to the arm upgrade. Afterwards the terminal button color will change to green and is therefore enabled. The player can now trigger the NetGame on the terminal and get the “pickup” skill.

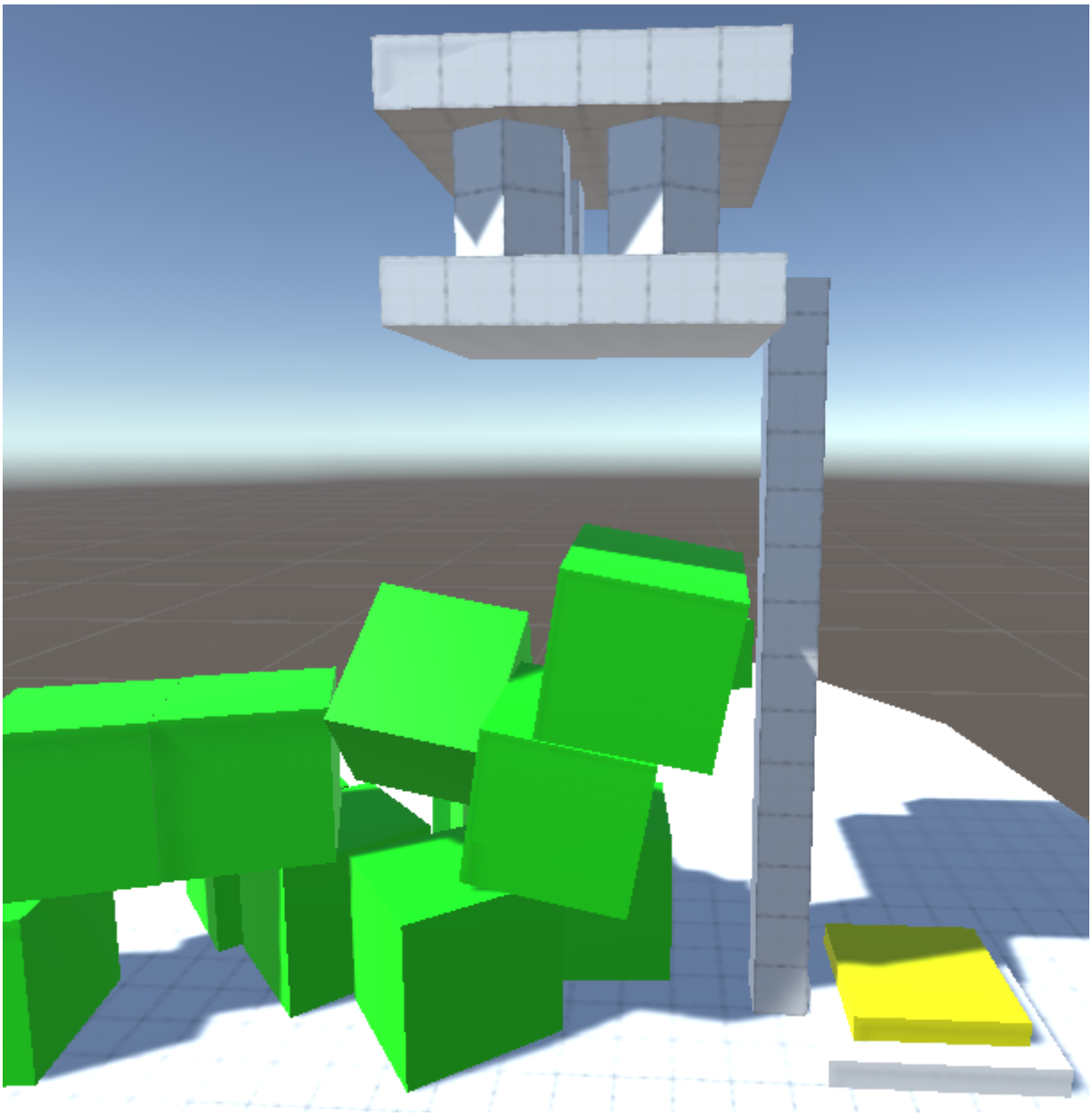
Currently there is no notification that the player has earned a new skill or that he ended the level.



Cube Spawner

We realized that it is currently possible to push the cube of into the void. o avoid the a deadlock of the level the player can spawn as many cubes as he like with the cube spawner.

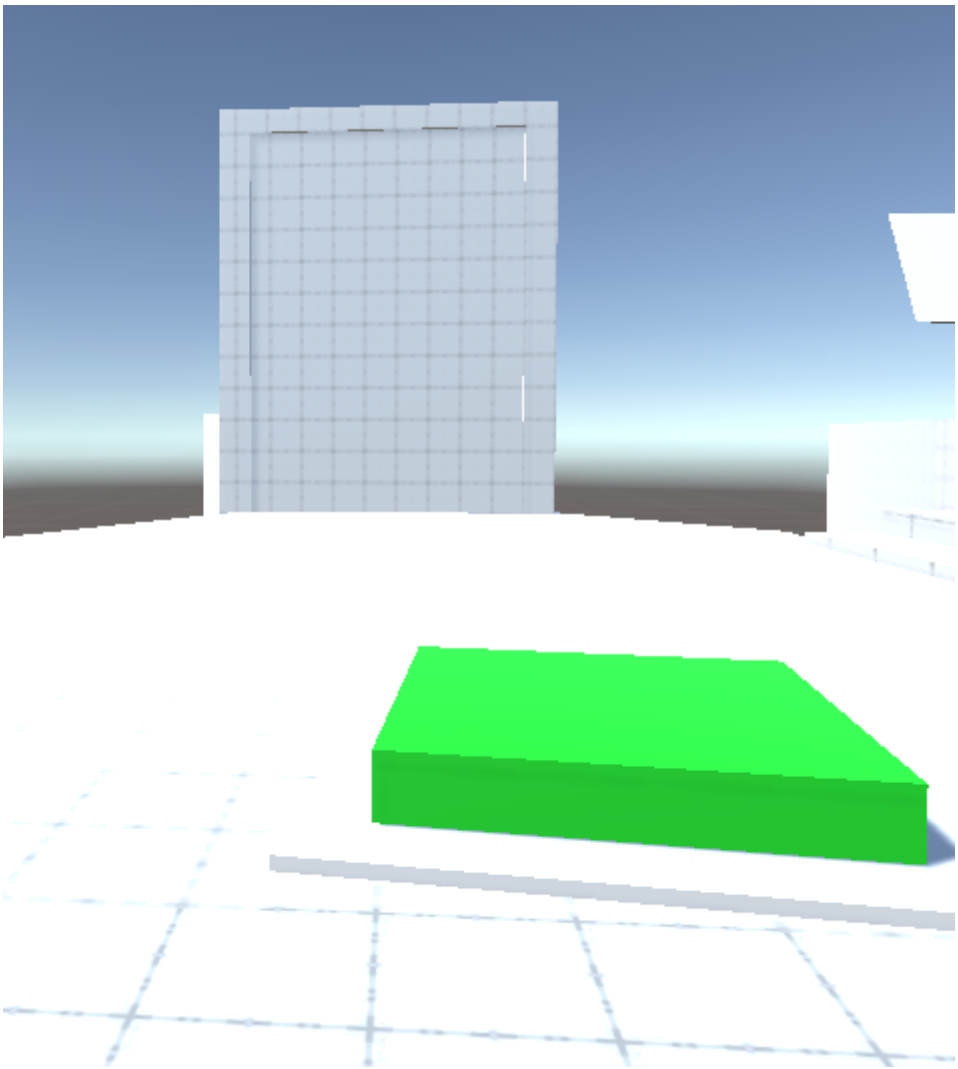
The cube spawner has a yellow button that the player can push, that will spawn a new cube that will fall down from the top of the spawner.



The spawner is also used in the second level.

Door

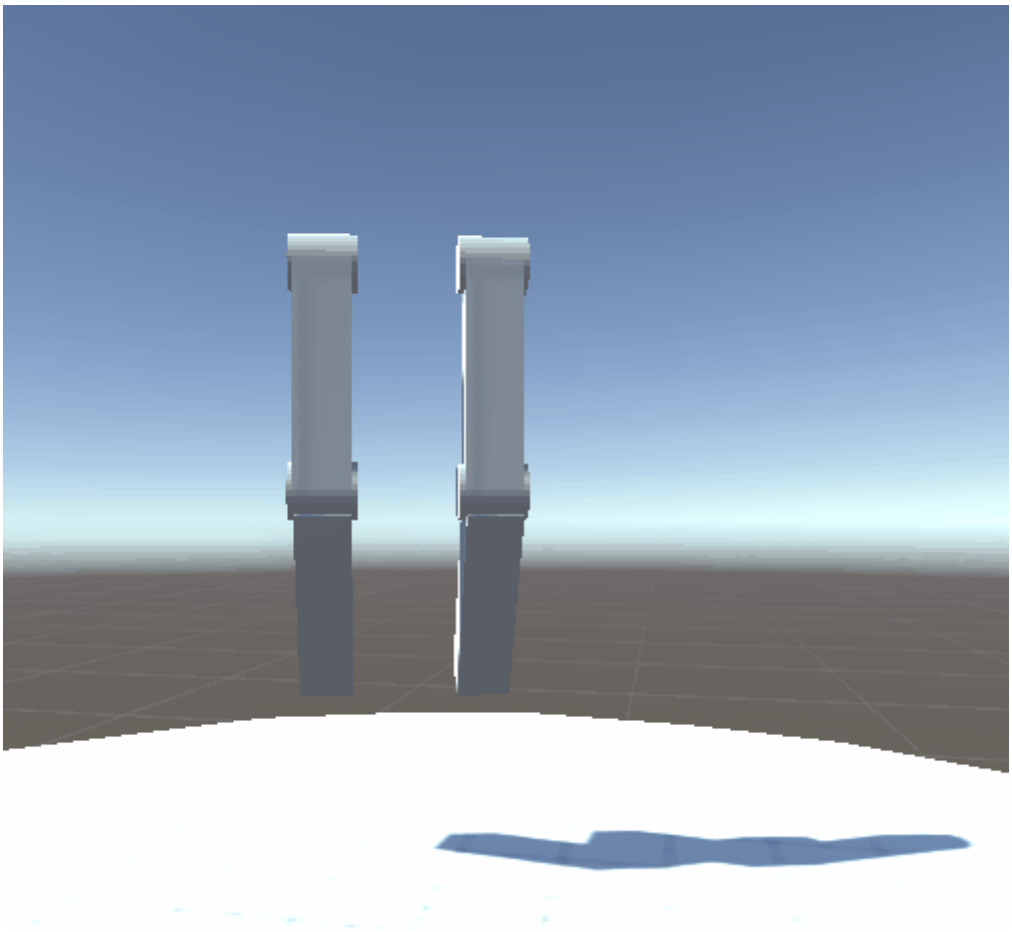
A door can lock areas behind it. To open the door a button has to be pressed permanently. The player can also open the door with standing on the button but as soon as he moves off the button the door will close.



Upgrade

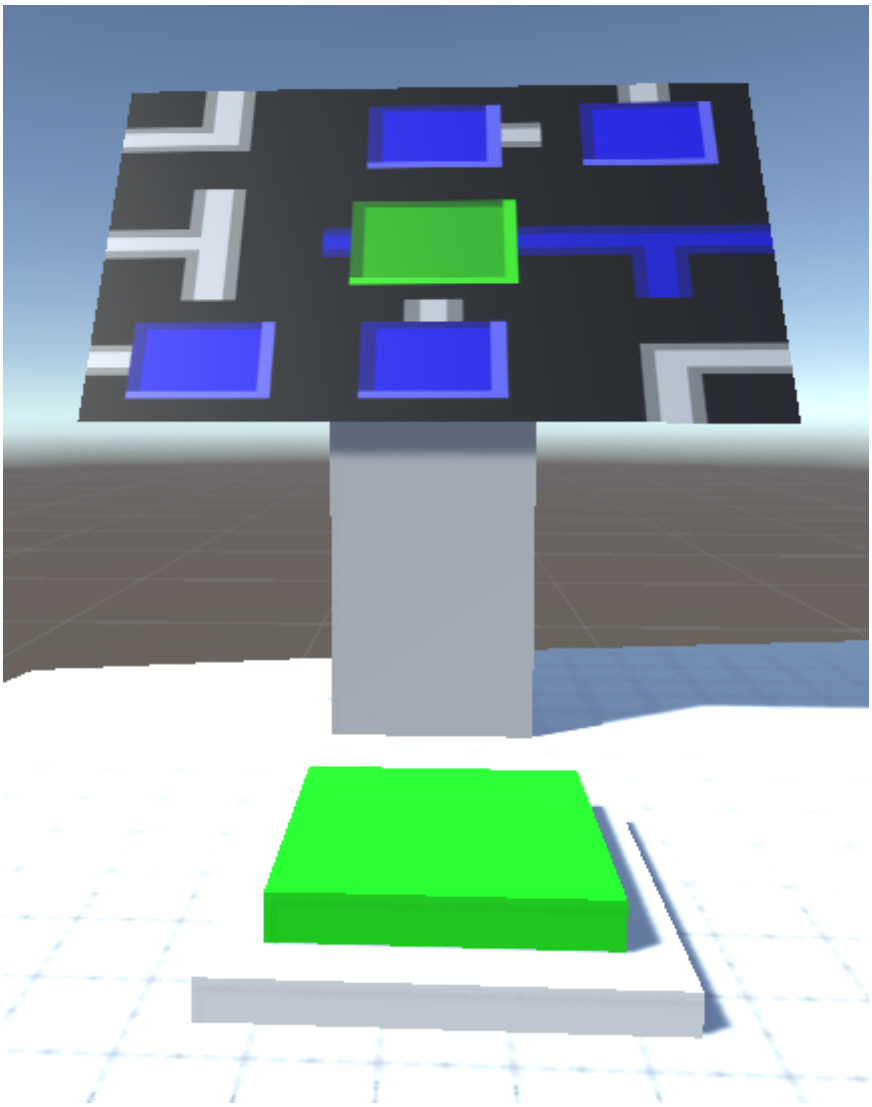
A upgrade is a physical object that the player needs in order to earn a new skill. He will have to pick up the upgrade before he can learn to use it with the NetGame.

The upgrade moves up and down and rotates in order to get the attention of the player.



NetGame Stand

In order to learn a new skill then player has to enable the use of an upgrade in its AI. this is done in the NetGame which is now directly playable from the gameworld. The stand is like a display stand that shows the game. The player can trigger a game with moving onto the button in front of it. The button can also be disabled which and it will change its color to red.



Time invested

Tim: 5h

Written on March 18, 2019

What do you think?

0 Responses

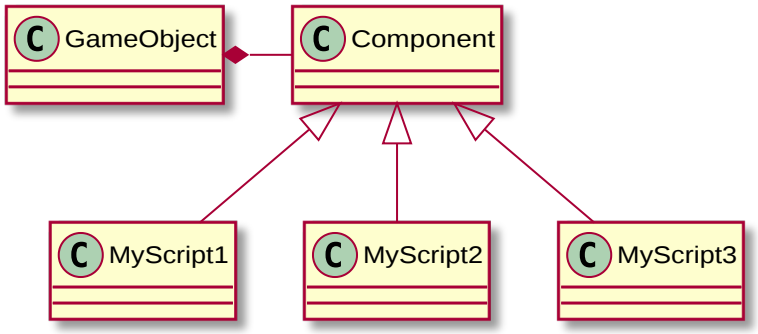
Architecture - exciting!

It is about time we talk about the architecture of our game.

Unity GameObject and components

We develop with Unity, that is why we can take advantage of a pretty great architecture to begin with: the GameObject and Component system.
A scene contains several GameObjects which have several base properties like name, layer (for rendering/collision masks) and tags (to find them based on categories). A GameObject can contain components, truth is they can not do much without components as they are the driving force behind your game.

Components control the behaviour and appearance of a GameObject, for example the `Transform` component which basically every GameObject has contains the position, rotation and scale of it and provides functions to change them or to act upon the local space these properties define.



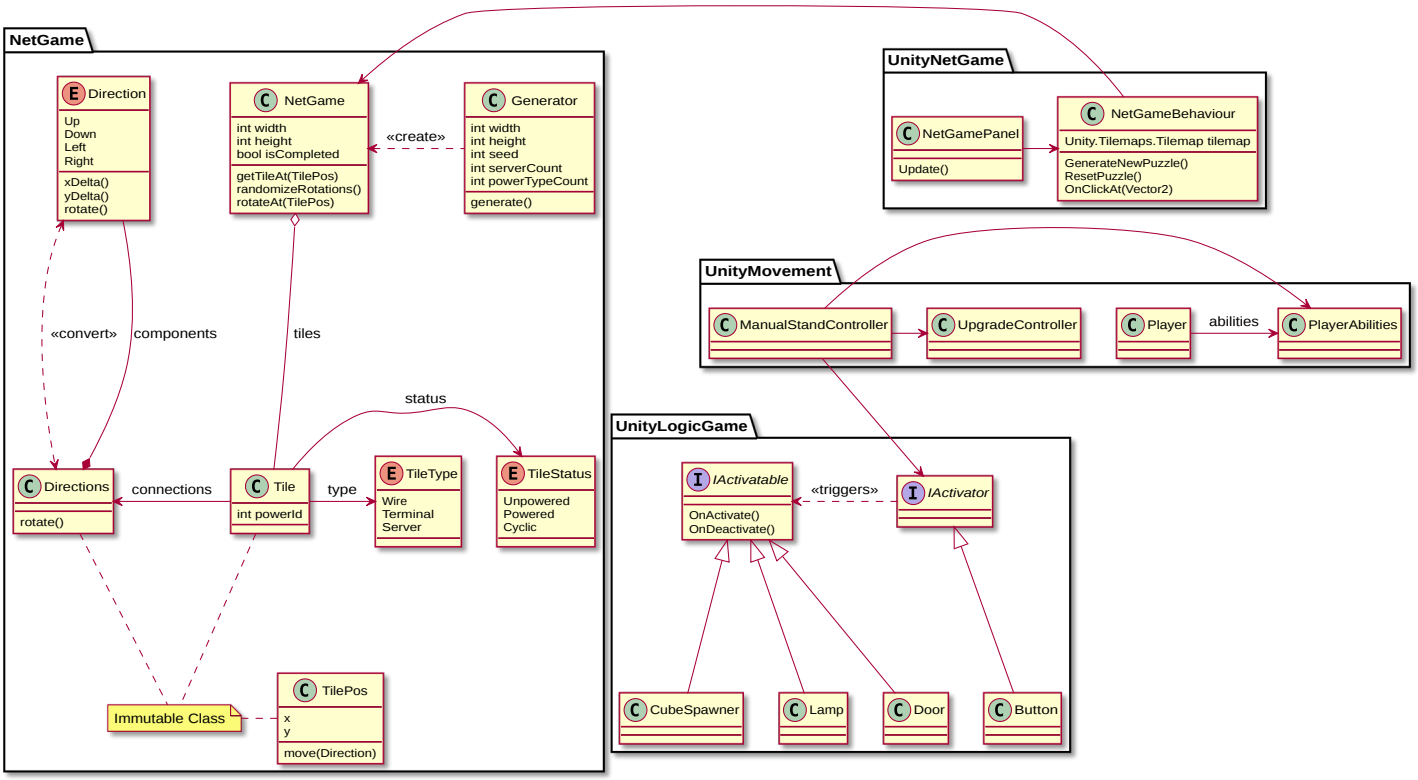
A Unity developer can either attach built-in components to their GameObjects like `MeshRenderer` and `RigidBody`, or they can write their own components in the form of a class which inherits from `MonoBehaviour`.

There are several special functions in a `MonoBehaviour` instance which are called by Unity at the right time like `void Start()`, `void Update()` or `void`

`OnCollisionEnter()`. With this the components act in a event-driven way and can easily be isolated from another.

What about us?

Now Unity brings many of the base infrastructure but how does *Purpose and Despair* fit into this? Like this (open in a new window to freely dive into it):



(every *class* in the “Unity[…]” packages inherit from `MonoBehaviour`)

The cute robot and its little companions

The robot in of itself lives with only two components: `Player` and `PlayerAbilities`. The first component controls the most important part: the player interaction and thus the movement of the robot. Every time the robot speeds against a wall or picks up a cube - the `Player` component knows about it (and propably is responsible for it). The second component contains information about what the robot has already learned, it will be used by e.g. `ManualStandController` to check whether the player is allowed to continue into further level progress.

The logic and physic puzzles one has to solve is based on a very simple concept: An *Activator* activates *Activatable* things. For now this means a `Button` activates a `Door` or a `CubeSpawner` when the robot steps or drops a cube onto it.

NetGame - fuzzy circuits but clean architecture

The logic of the NetGame is a bit more complicated, there are wires and power running about, things rotating out of control (Spoiler warning) and colors flashing. That is why we took that nasty logic and pushed it away from the Unity-related classes.

First we declared a `Direction` enum which in of itself can do things like tell how many steps in a certain dimension one should walk for that particular direction. But a tile in a NetGame can have connections in multiple directions, so we are in need of a container we called `Directions` (quite obvious naming, isn't it?) which now can represent any combination of one or more `Direction`. Because that is our main (and only) interaction, the rotation is already present here. After this we can write our `Vector2Int` clone, we call it `TilePos` and give it helpful extensions like move in a certain `Direction`.

The main game logic is implemented in the `NetGame` class. Itself is created by another class, the `Generator`. You pass in the parameters like width and height into it, call the `generate` function and can play, right? No games are generated in a finished state (so we can gurantee there is a solution to every puzzle), but after you randomize the rotations of every tile you can go and call the `rotateAt` function to play.

On Unity we have two more classes (components to be correct): one (`NetGameBehaviour`) to translate the abstract NetGame state into a rendered version of it and the other to put this into an actual 3D environment (`NetGamePanel`). You may notice that the `NetGameBehaviour` class has a `OnClick` function taking a `Vector2`. That is the second level of abstraction - here we take a normalized vector as position, translate it into a `TilePos` and pass it to our `NetGame` instance to update the game state

(and then update the rendered version). The `NetGamePanel` is the simplest version of a 3D integration, one could also imagine a half-broken screen on a robot arm swinging about to make the puzzle harder to solve (maybe a bit extreme but it shows the third level of abstraction in this structure).

So we can summarize the NetGame architecture:

1. Domain logic (in `NetGame`)
2. 2D renderer and 2D interaction (in `NetGameBehaviour`)
3. 3D integration (with `NetGamePanel`)
4. 3D rendering (with the Unity component `MeshRenderer`)
5. Graphics API (also handled by Unity, on Windows most likely DirectX)
6. Graphics card hardware

Time invested

Hermann: 4h

Written on March 21, 2019

What do you think?

1 Response

NetGame Rotating About

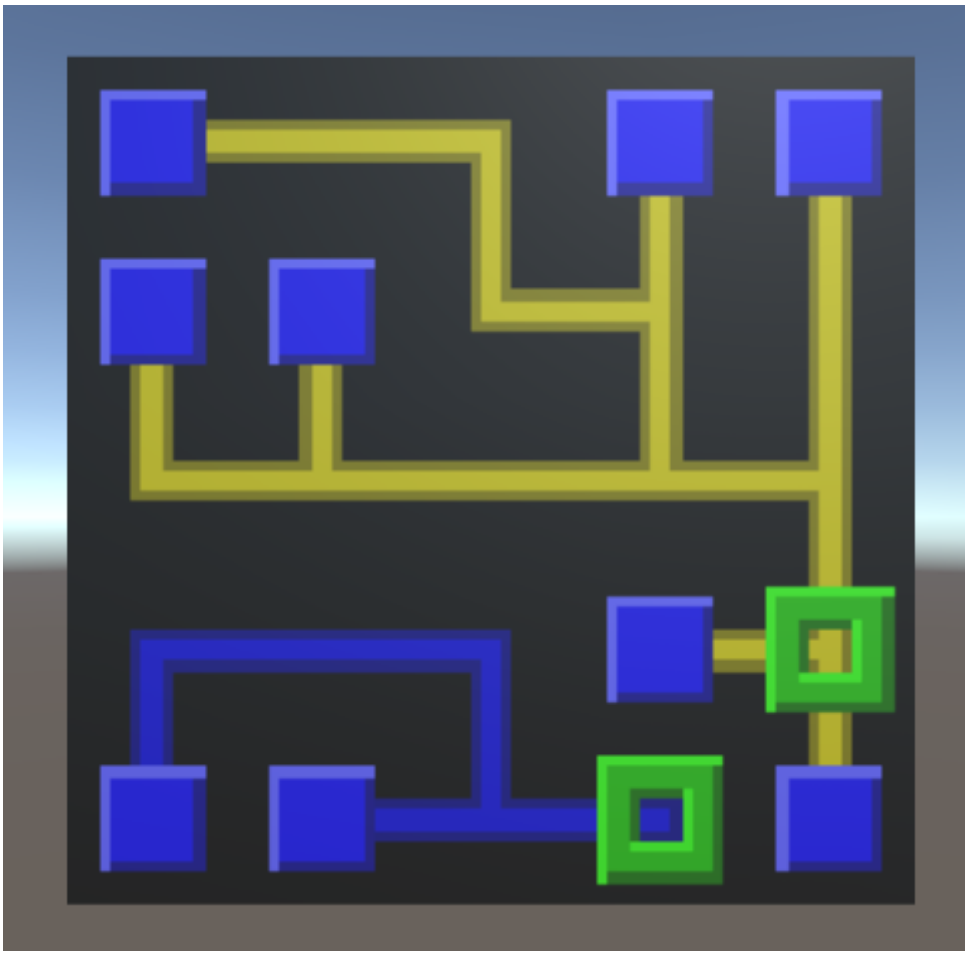
There are things spiraling out of control, find out what and how here!

Improvements

You remember the NetGame right? (If not read this). It got some improvements we want to show you:

Multiple power types

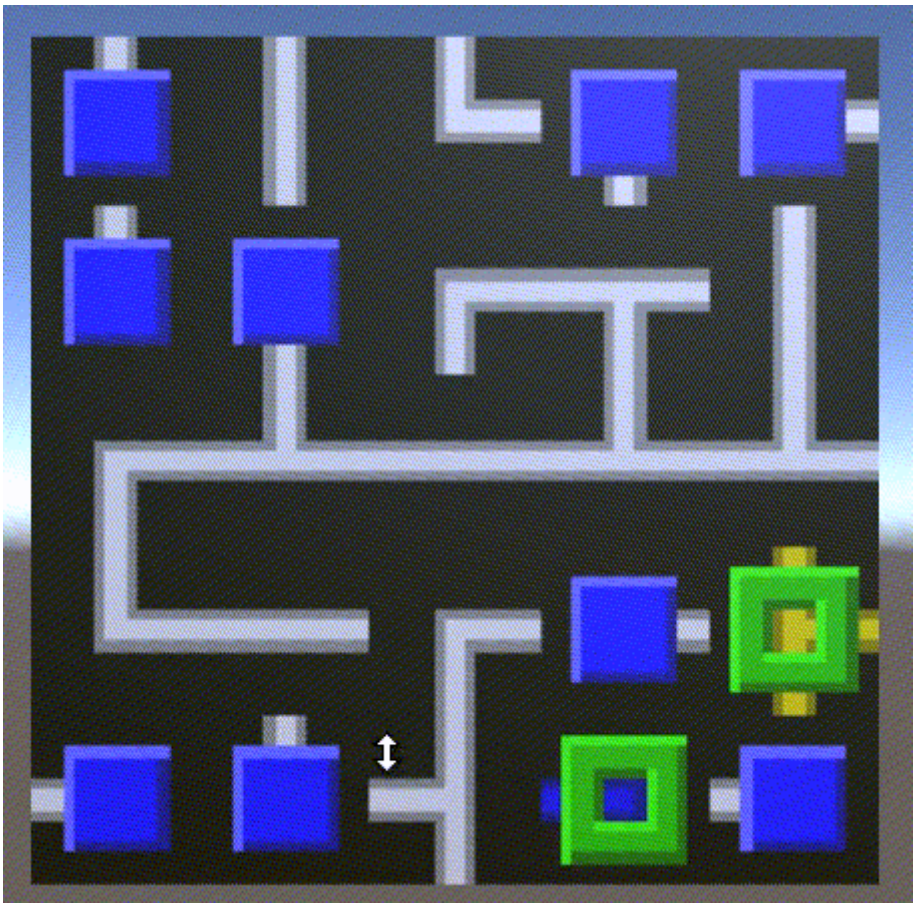
To confuse the player a bit more when he starts a new puzzle we have now multiple power types (for now the colors are blue and yellow). The terminals can be connected to any server any color but of course wire ciruits different colors may not be connected just like wire circuits of the same color (no short circuits!)



And yes this is the first screenshot with servers, it works just as well. It got a bit confusing which server got which power type that is why we have cut a window into the server block. This ought to make later puzzles a bit more interesting.

Animations

Interactions might have been a bit difficult to interpret, tile connections just jumping about when clicked. This we have changed as well with fancy animations helping the player understand what is happening:



Time invested

Hermann 3h

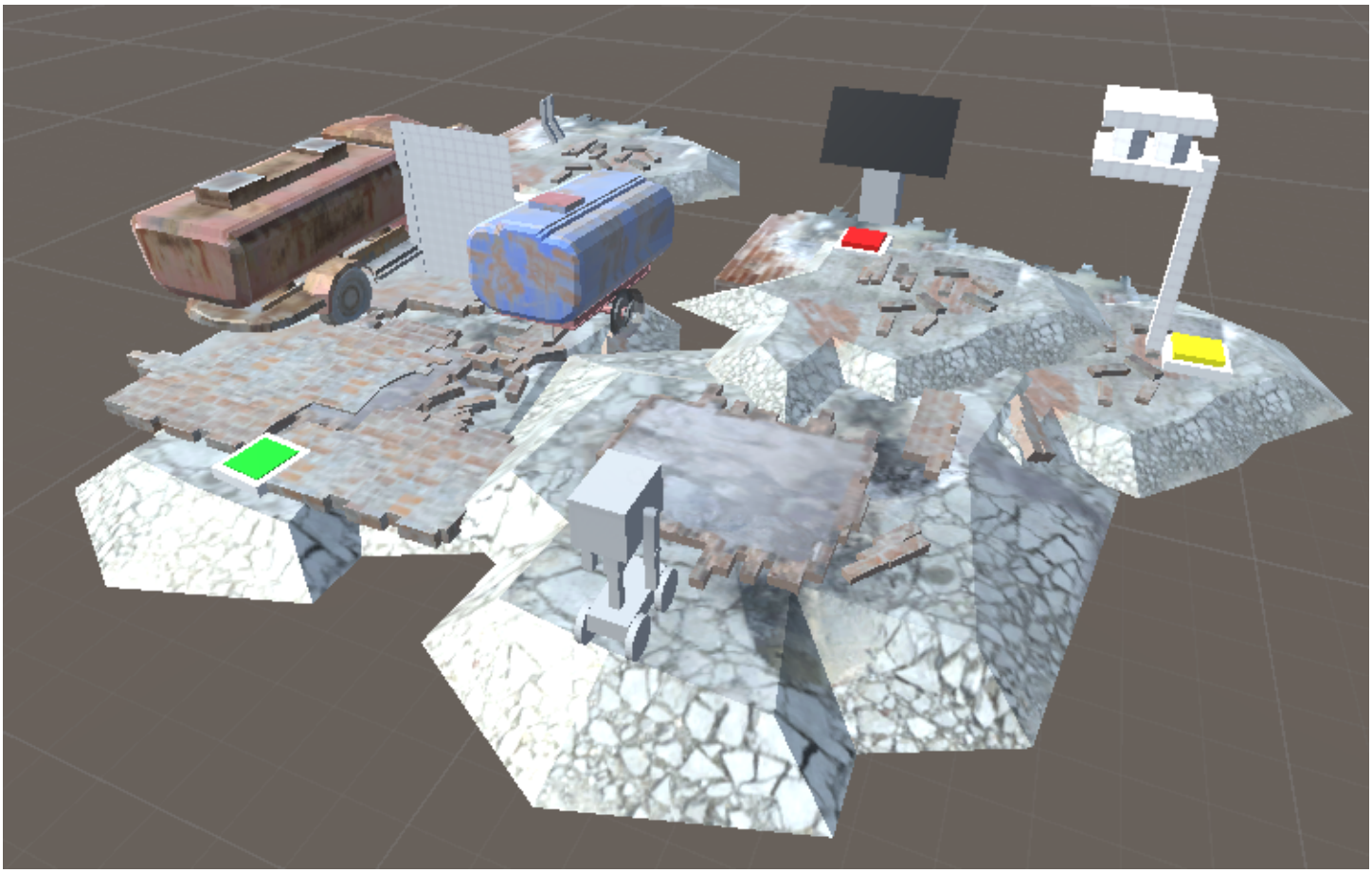
Written on March 21, 2019

Rework First Level

A visual rework of the first level.

Map

The map of the first level remains the same it just got a visual overhault.



The level is enhanced with different textures and models that we found on a distribution website for 3D models. the license of the models allows use to use them even if we publicly distibute the game.

There are currently objects that do not fit the theme of a junkyard. These objects will get an overhault at a later date.

The level could be integrated in a larger open world and would need no changes.

The ground was replaced by 3 different ground models that were used to build the path of the level. in order to bring more atmosphere to the level, there are now two trailers in the level. These also block the entrance to the upgrade.

Time invested

Tim: 3h

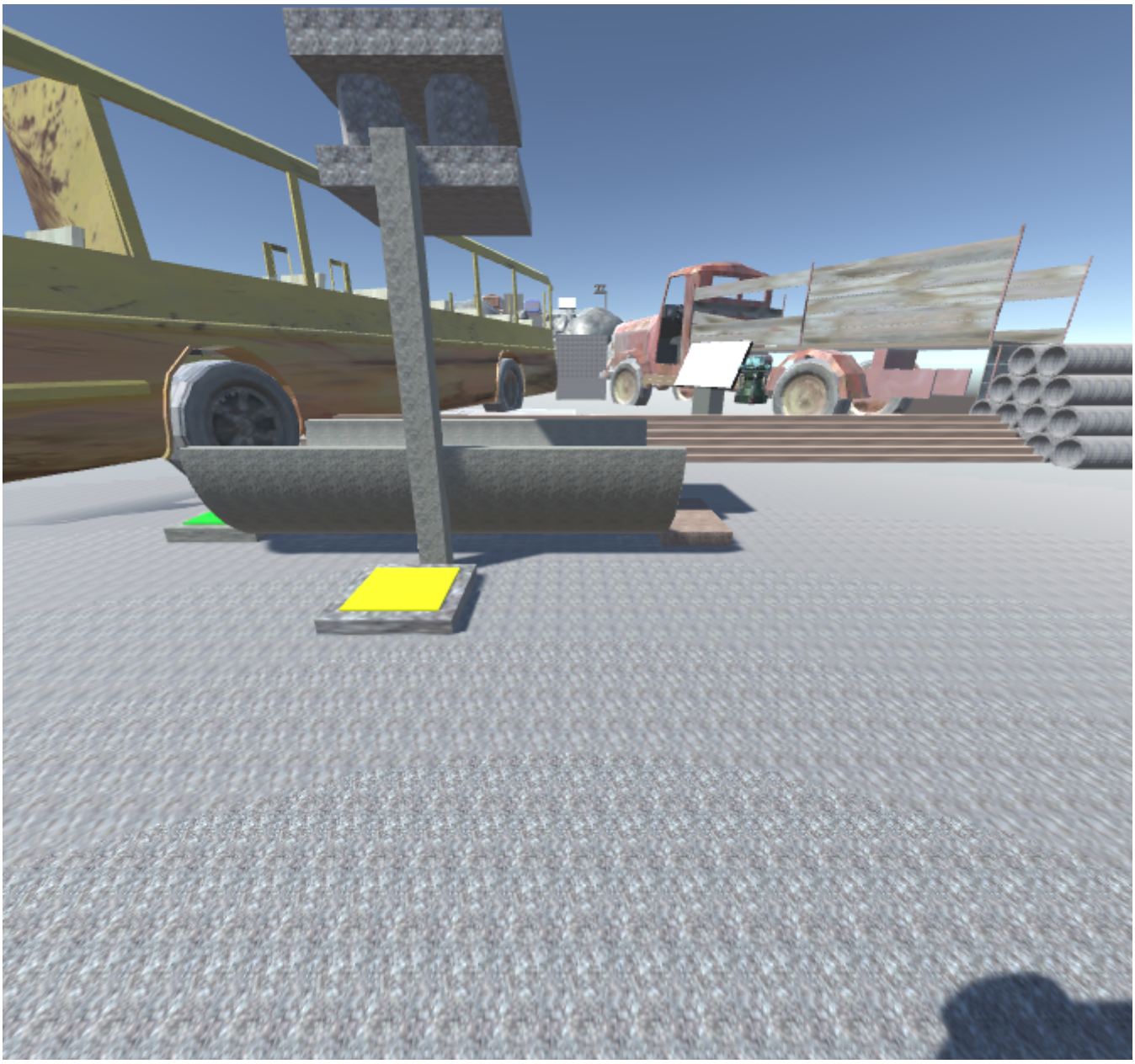
Written on March 21, 2019

Tutorial

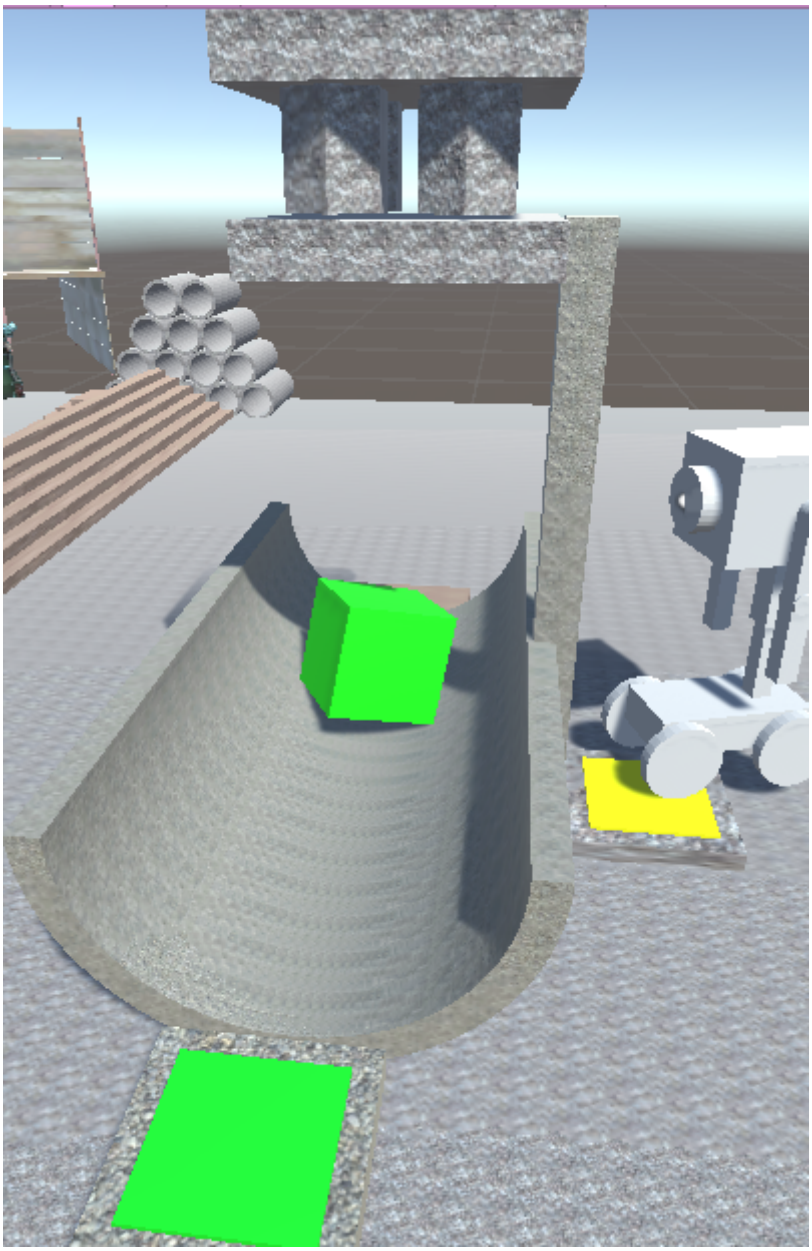
Turorial of the game that teaches you basic game mechanics.

Map

The map is designed to be simple and show the player different options of movement.

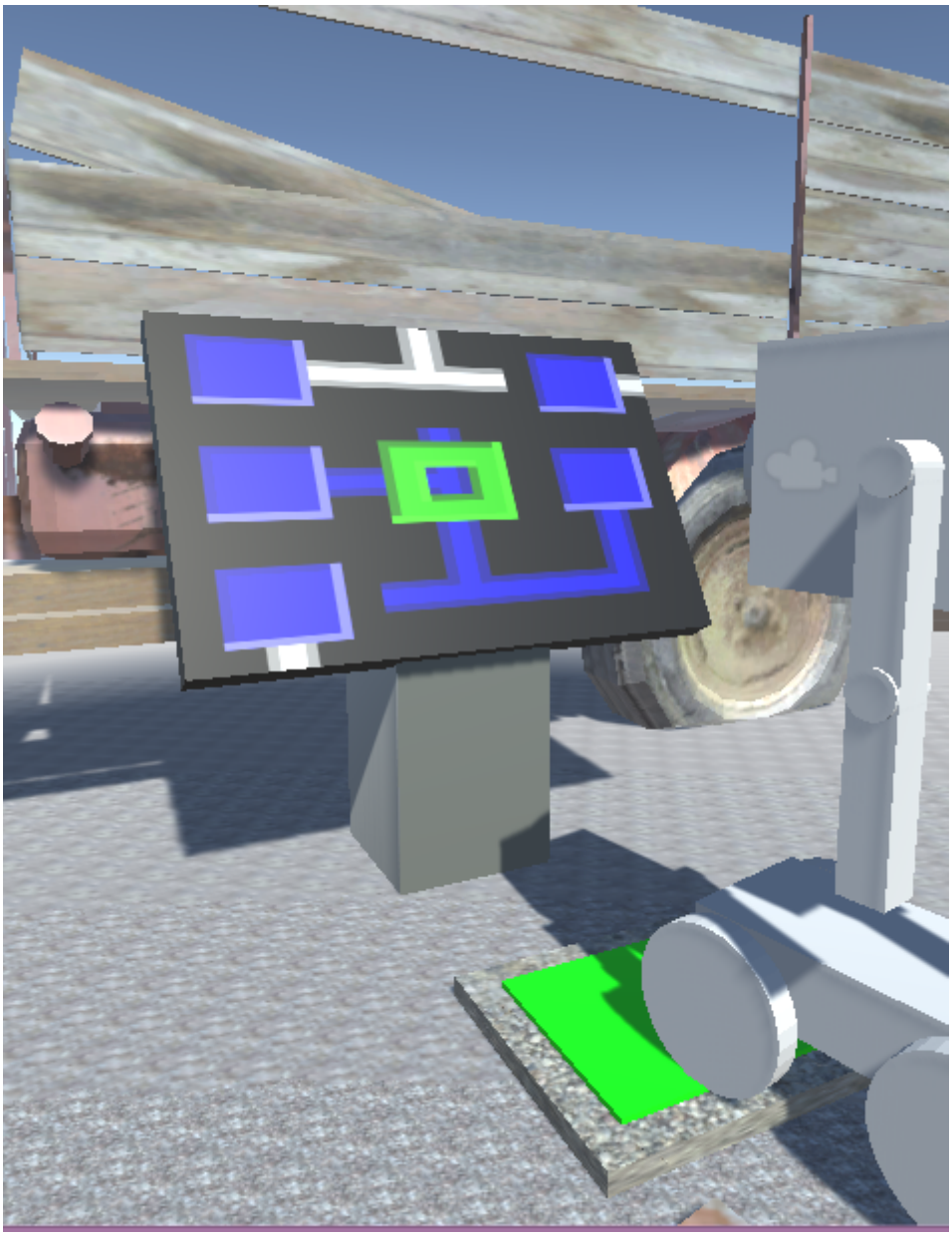


He can directly see two buttons, one yellow right in front of him. He will most likely move on the yellow button and learn that this will spawn a new cube. Afterwards he might try to move the cube on the green button since the halfpipe directs his movement towards it. The player will notice that he cannot move the cube and will loose interest in it for now.

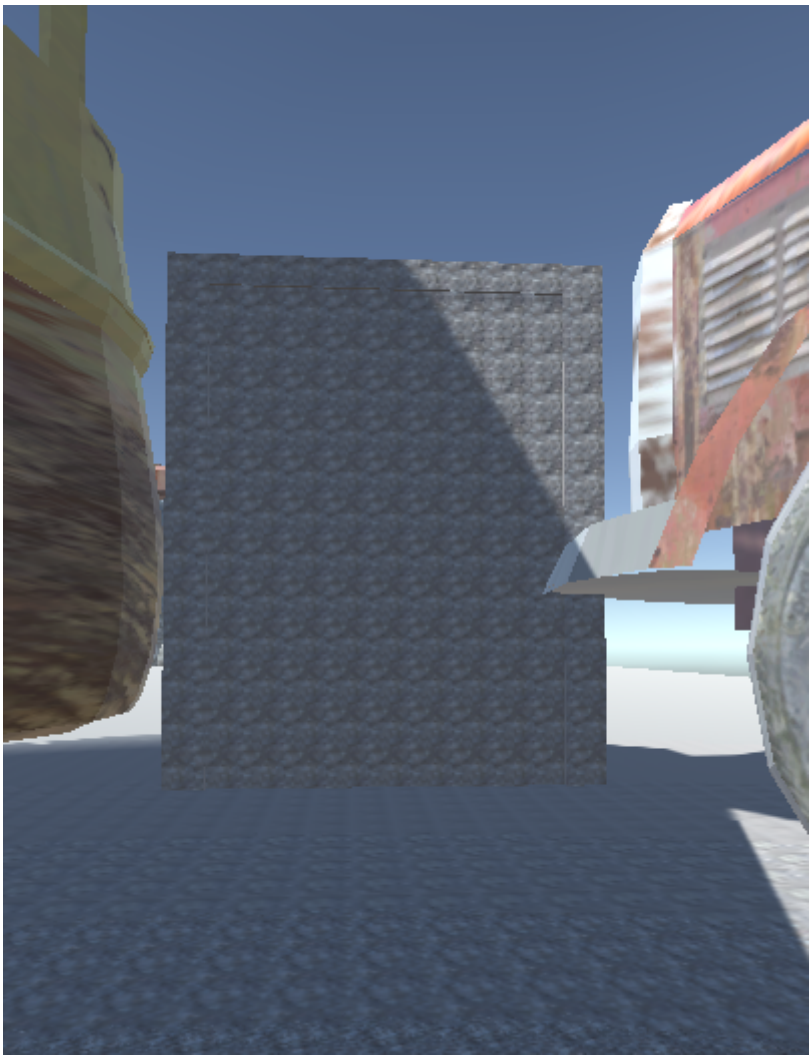


When he turns around, the player sees a moving object which looks like something important since it is moving in a otherways static world.

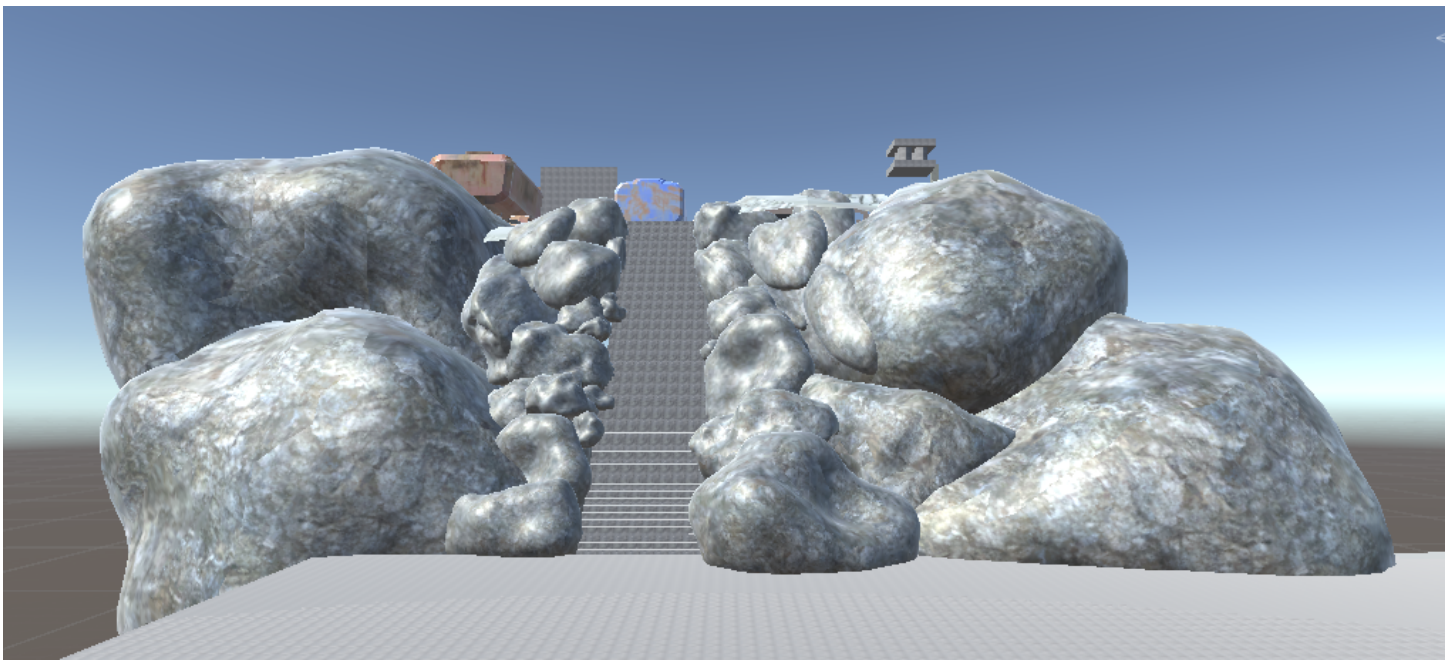
After he picked up the object, he'll notice that the red button turned green, which indicated that he can now do something with it.



After he played the NetGame, the door will not open. So he is left with trying out to push the cubve again to see if something changed. The player is now able to push the cube on the button and open the door to the world.



The player willfind himselfe in a open world where he can go whereever he wants. But directly infrom of him there are stairs which lead to the first level. The stairs will catch his interest and the player will do the firsdt level.



Further Activities

Currently there is no notification that the player has picked up something other than the object disappearing. There is also no notification that he learned a new skill. This will change in the futur with a notification of some sort.

Time invested

Tim: 5h

Written on March 28, 2019

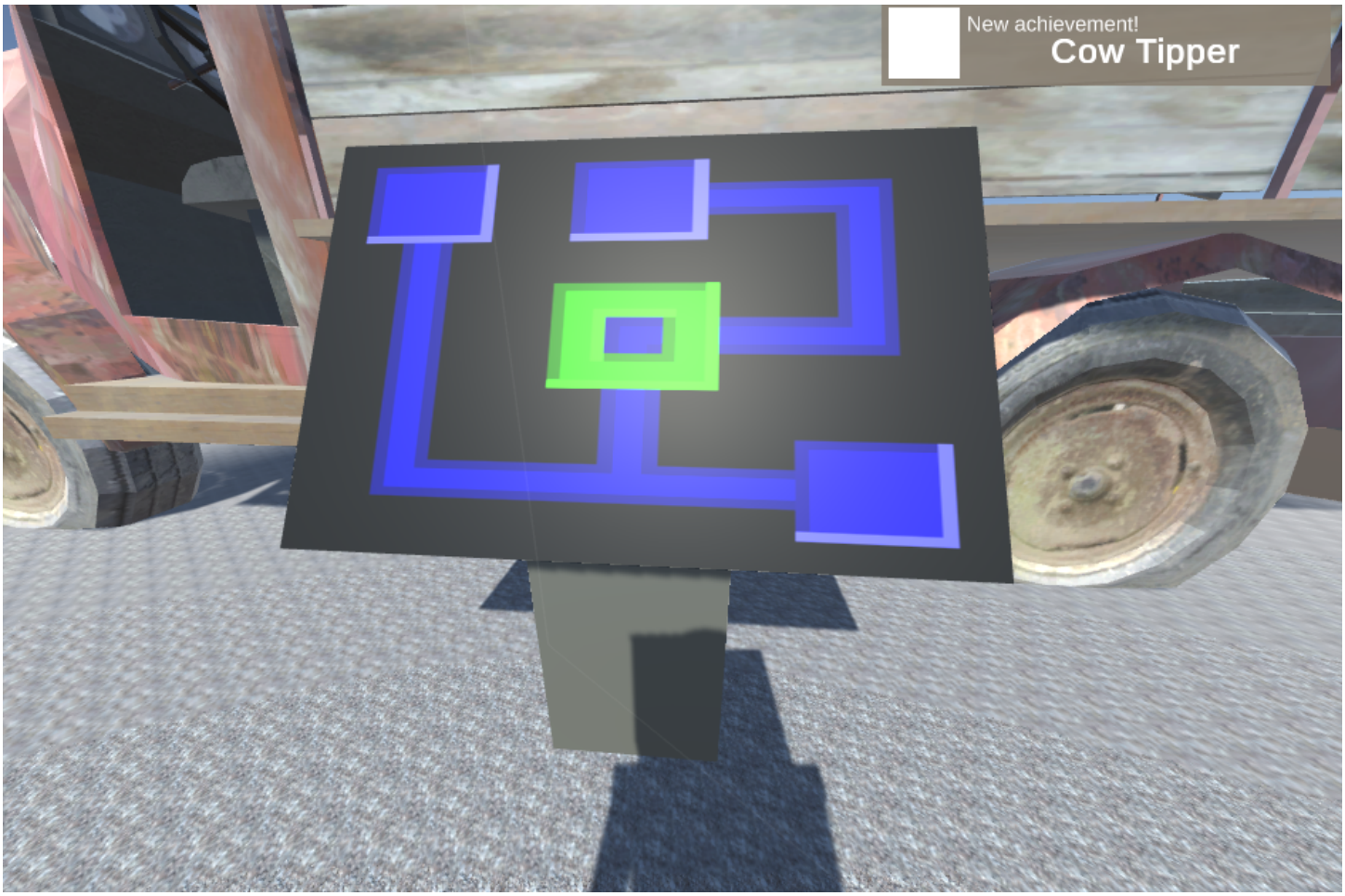
Achievements

You know you want them!

What is that again?

An achievement you ask? It is a token of appreciation, to honor your progress or a particular awesome feat you pulled. No there is not more to it, but the great thing is even if you don't think achievements are worthwhile you still want to collect them.

And who are we to not let you get them? Look at them!



Yes cute icons will be part of achievements.

Unity and their ScriptableObjects

So quick looksy on how they are implemented: They are declared as ScriptableObject we can configure in the Unity Editor. Then every script in the game can get the instance of the `AchievementManager` singleton and trigger its achievement. Also because ScriptableObjects are assets we can add the specific achievement to trigger as reference - also in the Unity Editor. Small fun fact to close: Achievements are the first piece of UI in the game

Time invested

Hermann: 4h

Written on March 28, 2019

How to get addicted

I want to finish the game!

Patterns

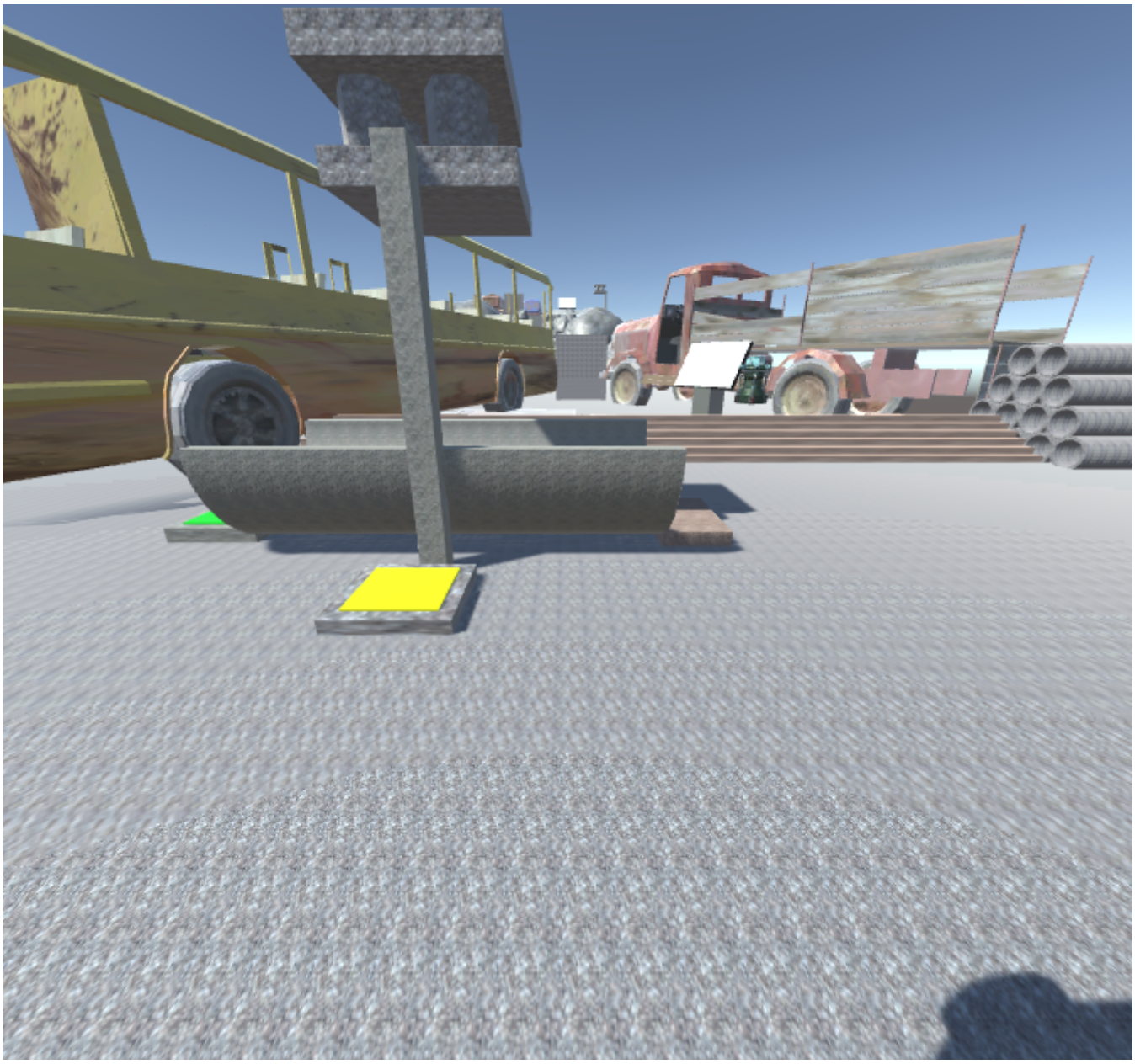
There are different patterns that games use to keep the player playing. These patterns can be grouped in four categories:

- Come and play
- Bring your friends
- Come back soon
- Stay together

We have implemented different patterns that fall into these categories.

Come and Play

To help the player get to know the game and learn him the basic mechanics, there is a tutorial in the game. During this you will learn how you can upgrade you character, move around the world and learn the restictions of you abilities. F.e. in the beginning you are too weak to push an object. Later on you will be stronger and can accomplish such tasks.



The tutorial is designed to present the player an open world but the player will most likely follow a (more or less) given path through the tutorial.

Bring your Friends

Since this game is a singleplayer game there is no direct call to action to bring in a friend. To get new player we are hoping for mouth to mouth propaganda and you tell your friend that this game is awesome!

Come Back Soon

A difficult task is to bring the player back to the game. We challenge this task to with our puzzles and our open world. Since we the puzzles are challenging the player will want to

solve them. If he cannot do it the completionist in the player will rbing him back to finish the puzzle.

The open world helps with this effect since the player can look at level he cannot do yet. This shows the player that he needs to come back to solve a different level if he wants to try the level he is looking at.

Stay Forever

The patterns for coming back also aim to keep you playing in the first place. The puzzles are brain teasers that give a good feeling when you solve them. This feeling is multiplied by the achievements that the player can get. He will be rewardes with these when he does certain actions.

There could also be a screen that displays the achievements you dont have yet. A completionist will want to get all of them.

Time invested

Tim: 2h

Hermann: 2h

Written on March 29, 2019

Level Variety

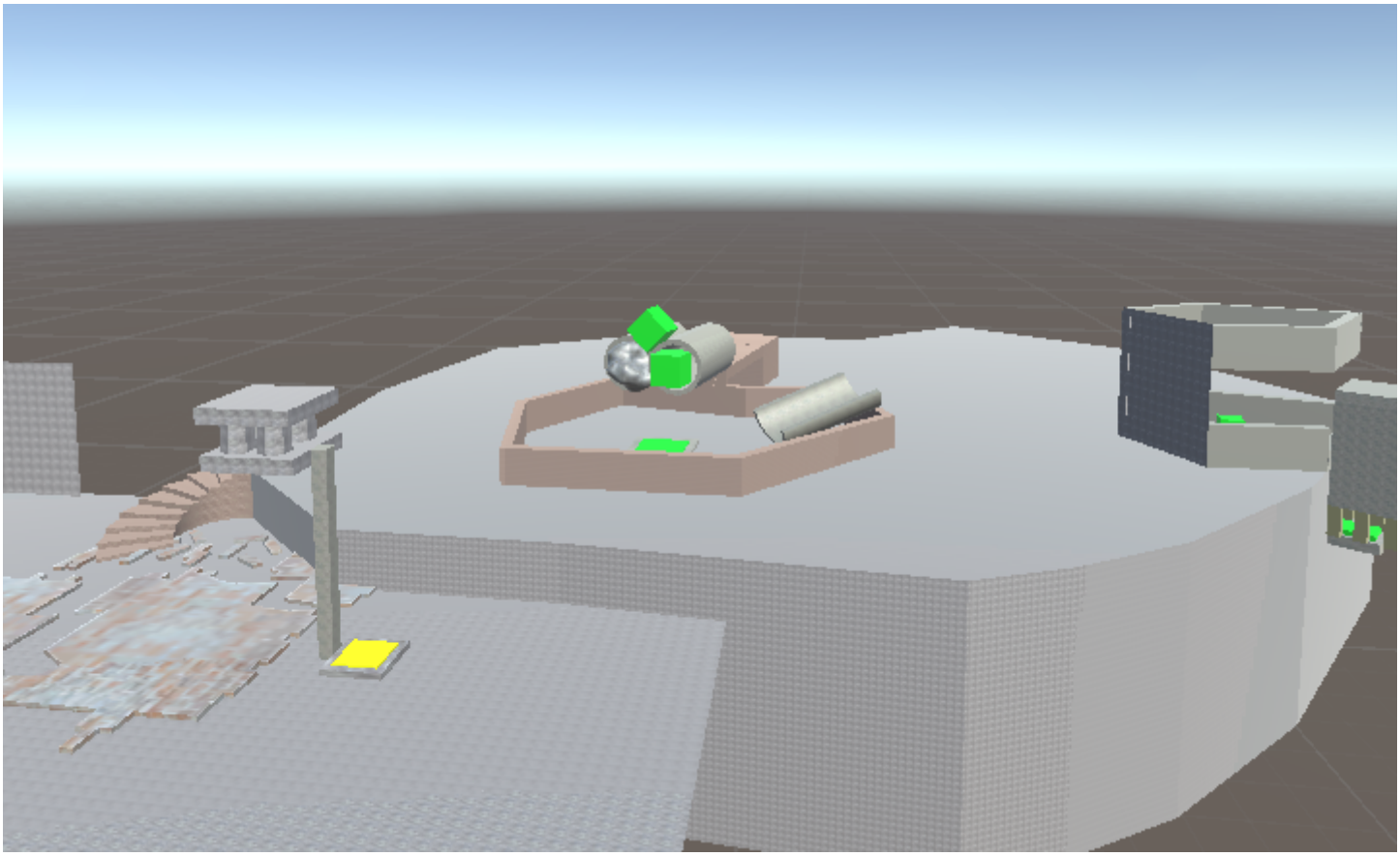
We need more level!

Different level

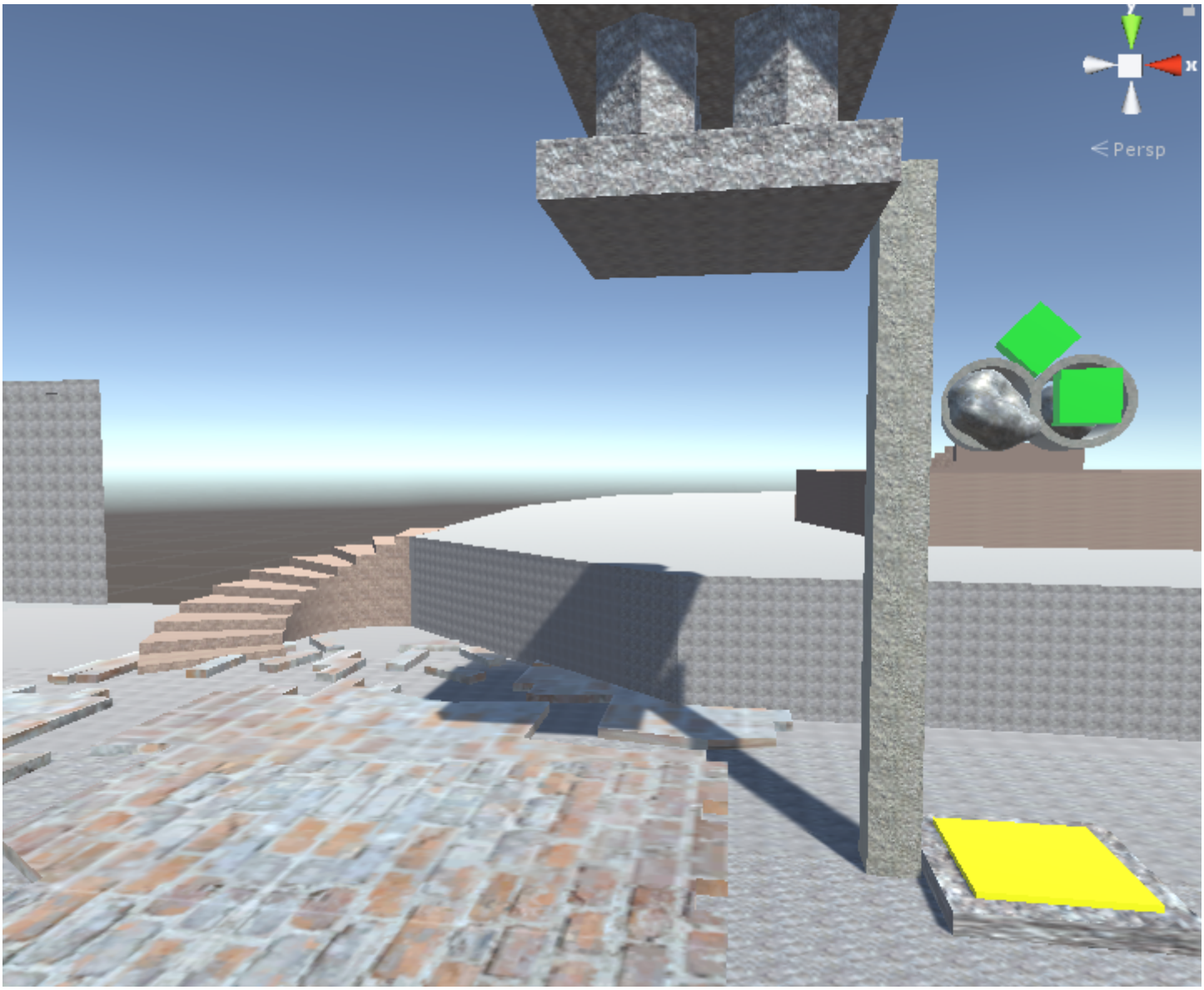
There have only been one levelformat that have been discussed in the blog before. There was the first and second level, in which you have to upgrade yourself in order to gain new skills. And the tutorial has a similar format with the difference that the purpose of it is to show the player the basic mechanics.

The game cannot only consists of these level, since there are only so much upgrades you can do. To get more level, there will be level which unlock new areas that you couldn't do before. These level will all follow one design principle. You have to use the abilities you learned in order to pass the level. Without these abilities the level will be impossible to complete.

We have created a level that follows this and opens up a new area (which is not existing yet) where you will need you skills ofmthe first area.

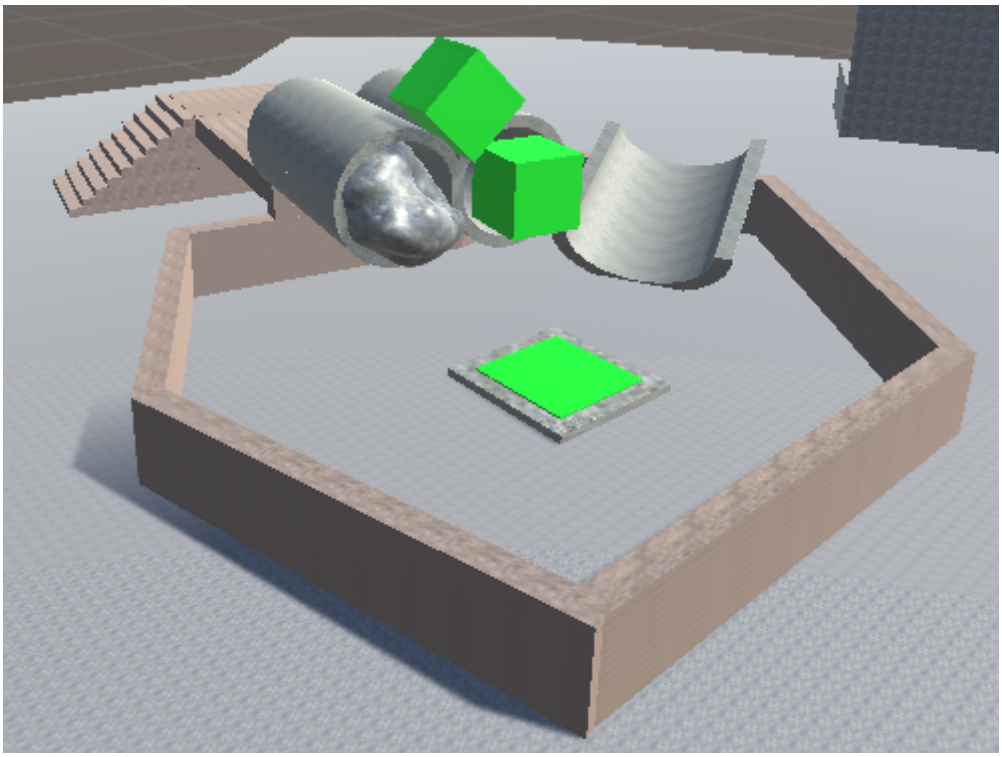


On the left you see the door that blocks the new area and in order to open it you have to prss the button on the far right. This level can only be completed with the “pickup” skill since you need a cube on order to open the door to access the other button. The entry challenge are stairs where you have to lift up a cube. The cube will be spawned at the start of the level.



On order to unlock the area door button you have to solve a simple puzzle. The puzzle is designed to not show the solution right off the bat but in reality the solution is extremly simple.

The puzzle looks like this:



It look like you have to somehow push the cube from the pipe or the cube on top of the pipe. An other option could be to use the ramp on order to slide the cube on the button. In reality both won't work! You will simply have to drive up the stairs with a cube in hand (that you picked up earlier) and drive over the wall into the area with the cube. The ramp is your escape from the area so you can continue the game and yre not trapped.

Time invested

Tim: 3h

Written on April 4, 2019

What do you think?

0 Responses

Polishing

Or “how to do much and achieve less”

Polishing

The act of polishing a game is to look for small improvements (mostly not gameplay-related) to make the game feel more “complete”. Some particles here, some animation there and a prototyp is starting to look more like some actual product. I have taken some time to do a polishing pass on *Purpose and Despair* and here are the little bits I have added:

Upgrade indicator

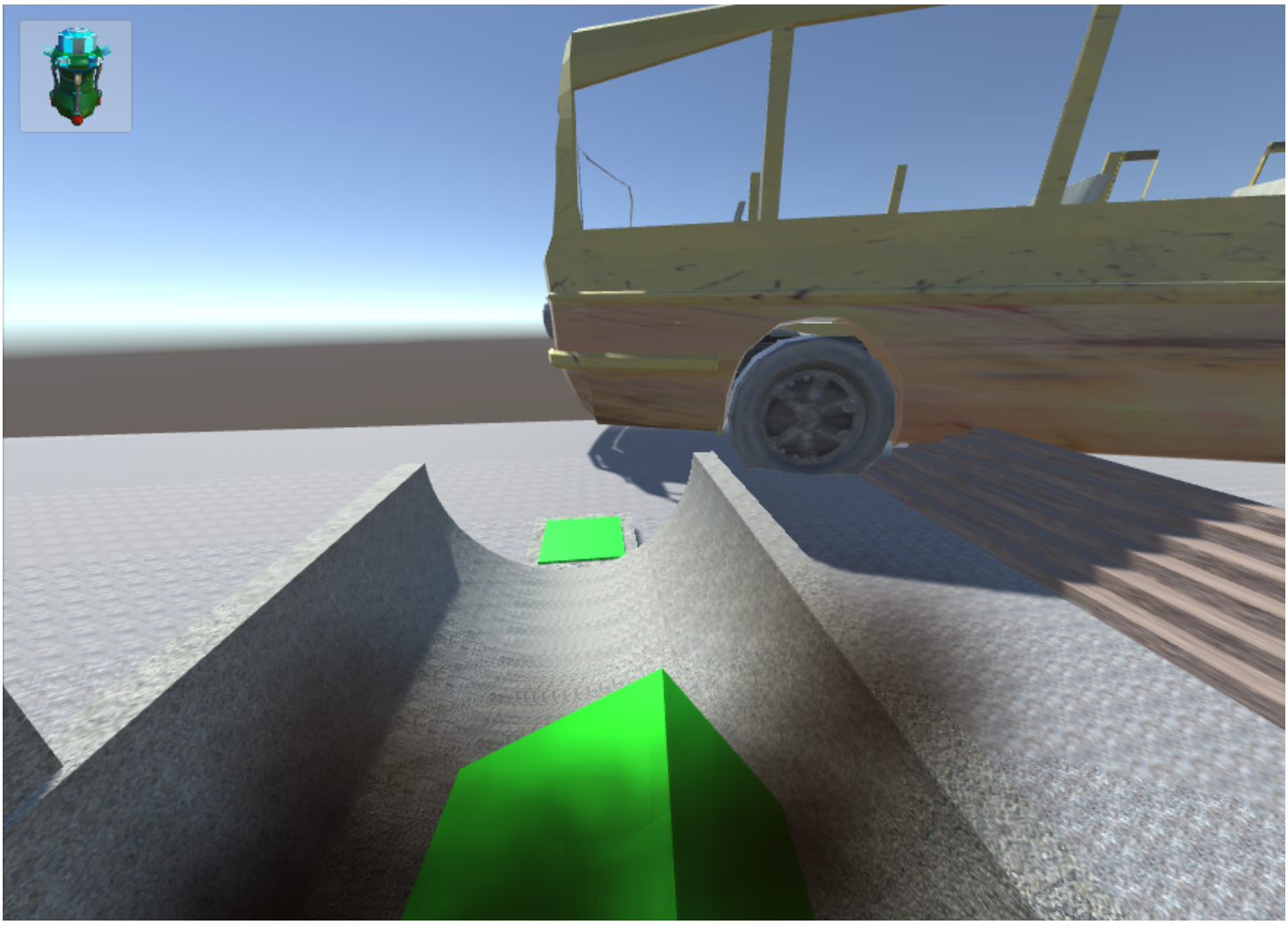
How do you know you picked up an upgrade? Easy right, you drove into it and it vanished. Basic game logic. But nevertheless it would be nice to get some kind of confirmation that would you think happened - actually happened.

So how about a little icon showing the thing you currently have picked up?

Motor stalling

You might remember that in the game at first you cannot even push a cube around - your motor is way to weak to do that. *That is* why you have to get that upgrade in the first place.

So what happens if you drive against the cube? Nothing. Any kind of explanation at the current state? No! Thats bad!
What actually happens is that your motor is stalling and thus heats up quite a lot, enough to make your wheels smoke. So let us see that in action:



Trust me it looks even cooler when those particles are animated.

Achievement icons

Achievement icons are just as important as the text accompanying it, so why do you just get some white square? I don’t know but I fixed it. Play the game to see it for yourself!

Time Invested

Hermann - 3h

Written on April 4, 2019

Analytics

Give me all of you data!

Unity Analytics

You can get a better insight into your game and if the mechanics work as intended with some user data. To getter these data and to visualise we use Unity Analytics. You can enable Unity Analytics with the simple click of a button and it will automaticly collect certain data, like the daily users. This s a nice aspect of it but to unleash the entire power of analytics you will need more than just the standard events. Lukily you can create custom events that will give you a detailed insight into your game, to that extends that you programmed it to be.

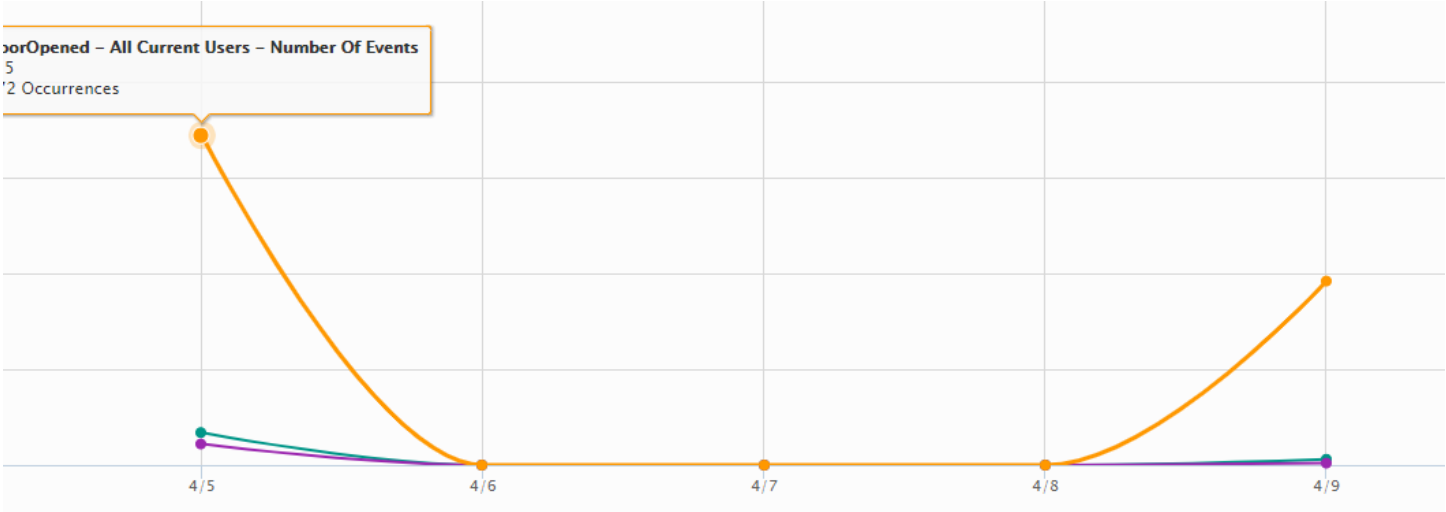
Custom Events

We currently have 7 custom events:

- Achievement Gained
- Buttons Pressed
- Cube Spawned
- Door Opened
- Netgame Completed
- Netgame Created
- Stepped Up

With this events we can monitor more precisely what the user is doing.

These events are also automaticly collected and send to the Analytics dashboard. The dashboard displays the data and allows you to show them in a graph.



In the graph you can see some test events we generated to test the events and dashboard.

Problems

In theorie we have done everything we need to get some good analytics going, unfortunately we ran into some problems. The events are not sended consistently, this means we dont have all the events that are been send to the dashboard. Unity provides an event validator thta displays the events that are send. In the validator we can see the events been send but they never appear in the dashboard. We have search for solutions for it but we weren't able to fix the problem yet. For this reason we only have some event data in the dashboard and not as many as we would liked.

The reason for this is that Unity has a hourly analytics request cap and it won't let you send more than 100 events per hour per user. Because of this we can't send as many data as we would liked. F.e. you step up way to often to track it. The way we fixed this is by sending only a few events.

Another problem with Unity Analytics is the time it takes for events to be processed and displayed in the dashboard. We have encountered waiting times inbetween event creation and the dashboard showing them of multiple days. Maybe this is also the reason why we don't have consistent event data in the dashboard. Nevertheless these waiting time would be acceptable if we get all the events that are been send. It looks like this is not the case though.

Time invested

Tim: 6h

Written on April 25, 2019

Making a demo

What needs to be done to be playable by somebody not us?

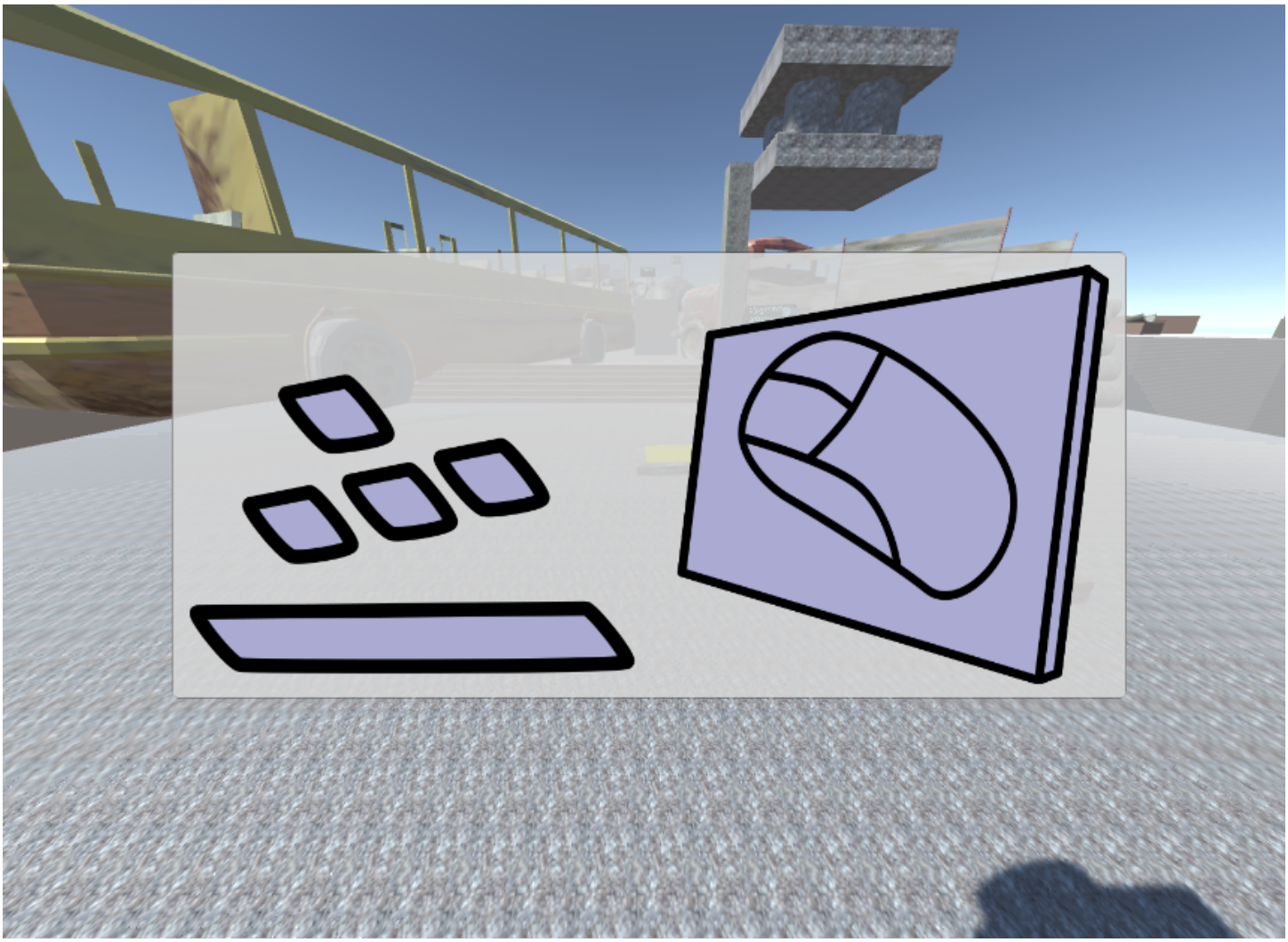
The gist of it

For a technical prototype that we had to be considered as being an actual demo of a game, the main thing that should be complete is the possibility for others to pick it up and play it. If each new player needs personally teached instructions (by us the developers) it is no good.

So this is what got done in the last period of time.

Start/End-Panels

When starting the game right now, you will get this screen:



As you might figured out, the buttons to press and the input method for the netgame puzzles were unspoken pieces of knowledge. And yes, we said (probably multiple times by now) that we don't want an explicit tutorial - but we don't reaally tell you the keys to press (not unambiguously at least).

How *do* you get upgrades anyway?

Also there is a sequence kept quite secret for the player on how to get an upgrade for themself. For the development that looks like this:

1. Collect the part
2. Solve the puzzle

But for the player there is more going on:

1. What is that big monitor-like thing?
2. What is that floating reactor thingy and why do I collect it?
3. Why should I solve the netgame puzzle?

4. What happened after it?

And to make that a bit more explicit we used a bit of black-screen time the netgame monitor had previously. At first you will see the following on the screen. After you progress on getting the upgrade, the screen will change accordingly, giving the player a bit more information on what the heck is actually going on.

How do the players start the game?

So right now, if you don't have the correct versions of multiple programs (Unity and Blender) installed, you won't be able to actually start the game. That is bad… we know… But builds are coming! … Soon… As soon as I have access to a bit more uplink speed

Time invested

Hermann: 5h

Written on April 25, 2019

What do you think?

0 Responses

 Upvote  Funny  Love Surprised Angry  Sad

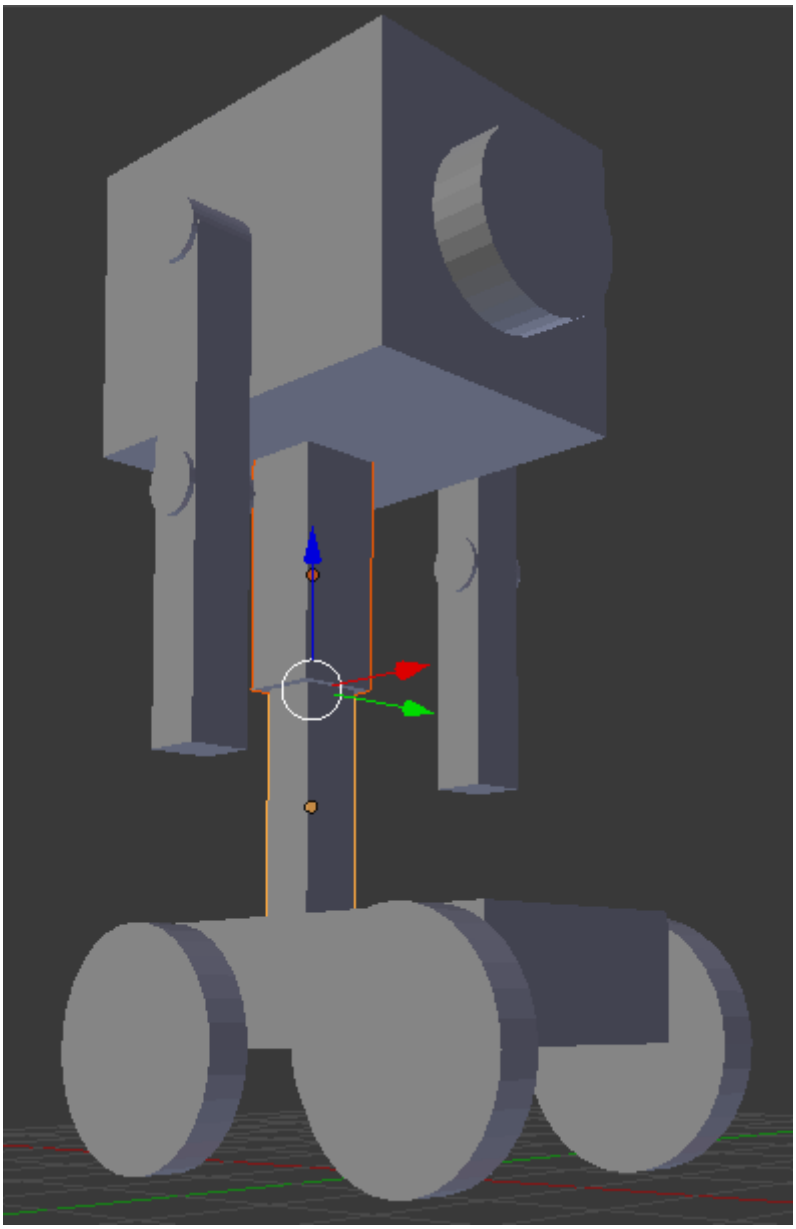
Final Game Prototyp

Let’s finish it for now!

Purpose and Despair

Purpose and Despaitr is a first person puzzle game. It aims to challenge the the player by providing interesting levels that need out of the box thinking.

You are a small roboter the is stranded on a lonely planet. The task you have to tackle is to leave the planet.



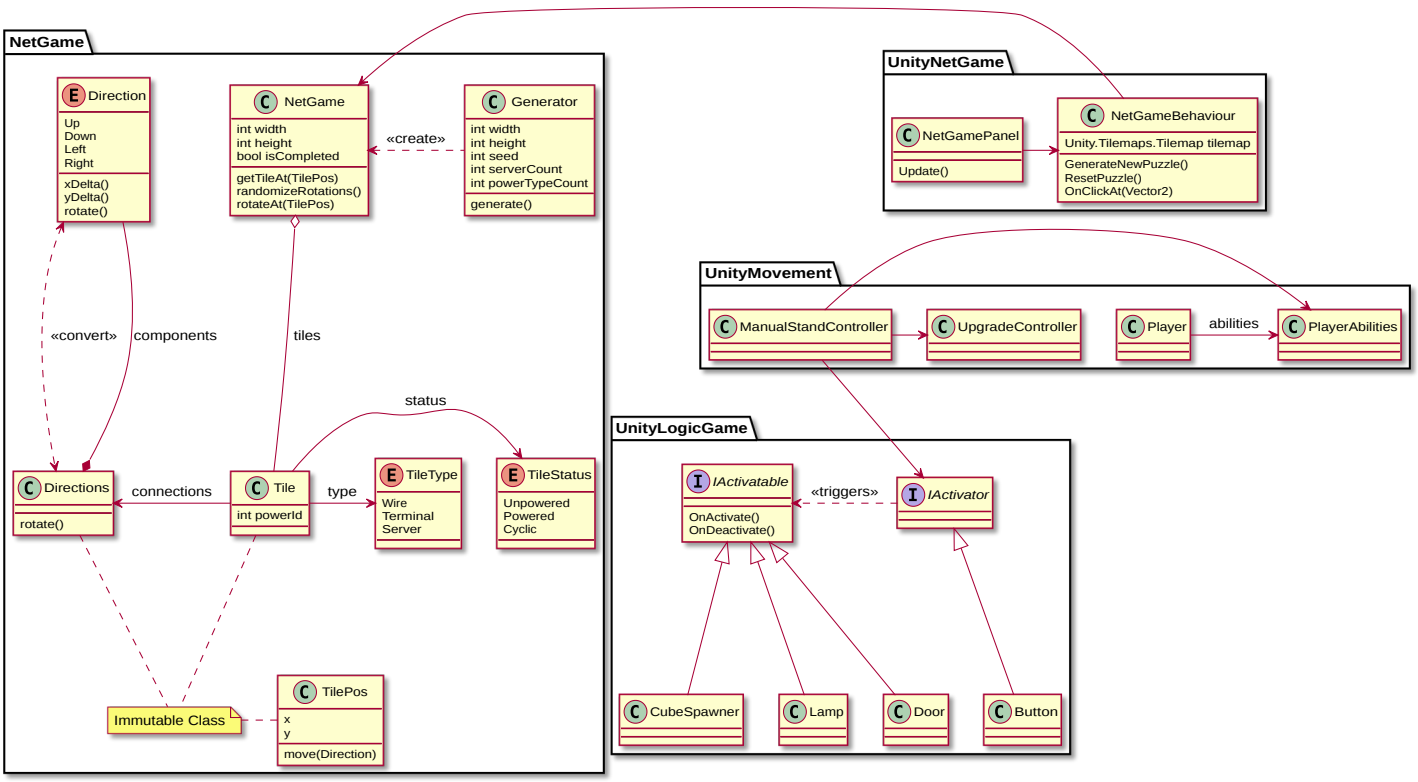
To achieve you goal you will have to learn new skills on order to complete more difficult tasks. And if you complete your journey you might have learned your purpose or are in despair.

Demo

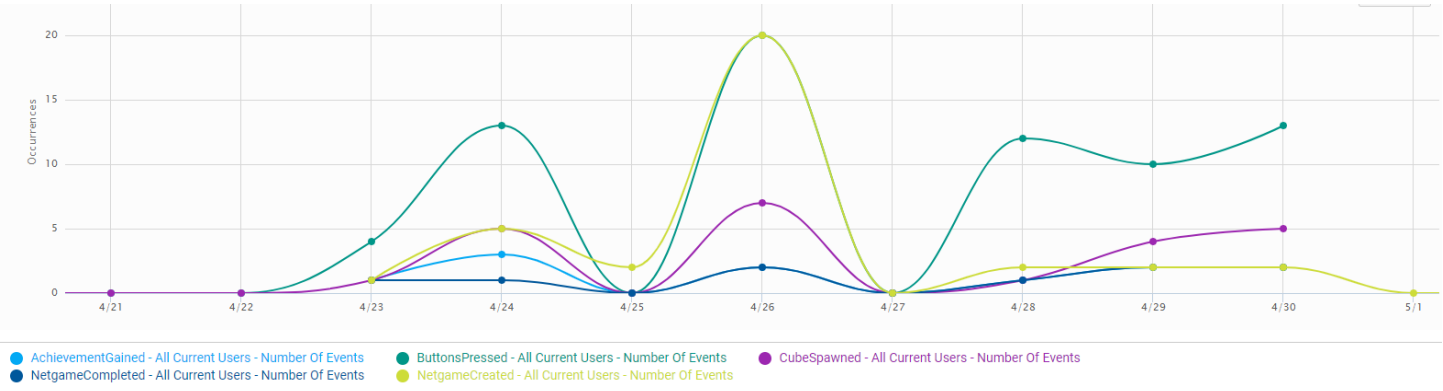
Architecture

During the development of the prototype we focused mainly on getting a great game put together, but there would have been many problems without having a disciplined architecture. Our key features on that regard are based on isolation of different components and responsibilites. For example the puzzle game mechanics could be completly cut out of the code base and it would still work perfectly. Also we leveraged

concepts from domain driven design (e.g. immutable value types) to further enhance our code quality. You can see a summary of the architecture in this diagram:



Analytics



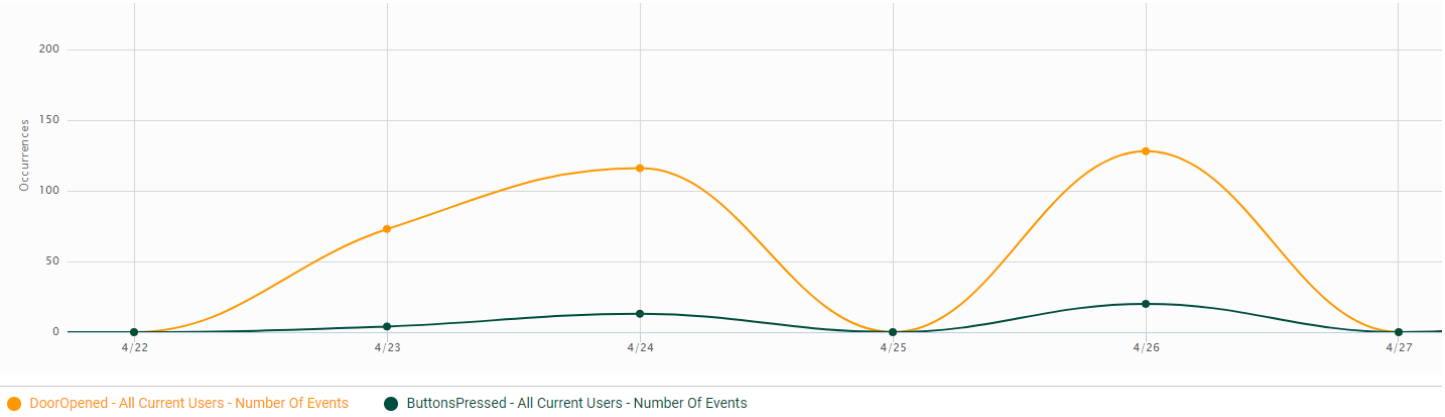
In the picture you can see most of the information we collected during our test period. We collected the following activities with these events:

Event	Activity
AchievementGained	An achievemnt is unlocked by the player
ButtonsPressed	The player uses a button
CubeSpawned	The player spawns a cube
DoorOpened	A door is opened
NetgameCompleted	A NetGame is completed

Event	Activity
NetgameCreated	A new NetGameis created
PickedUp	The player picked up a cube
SteppedUp	The player stepped upa stair

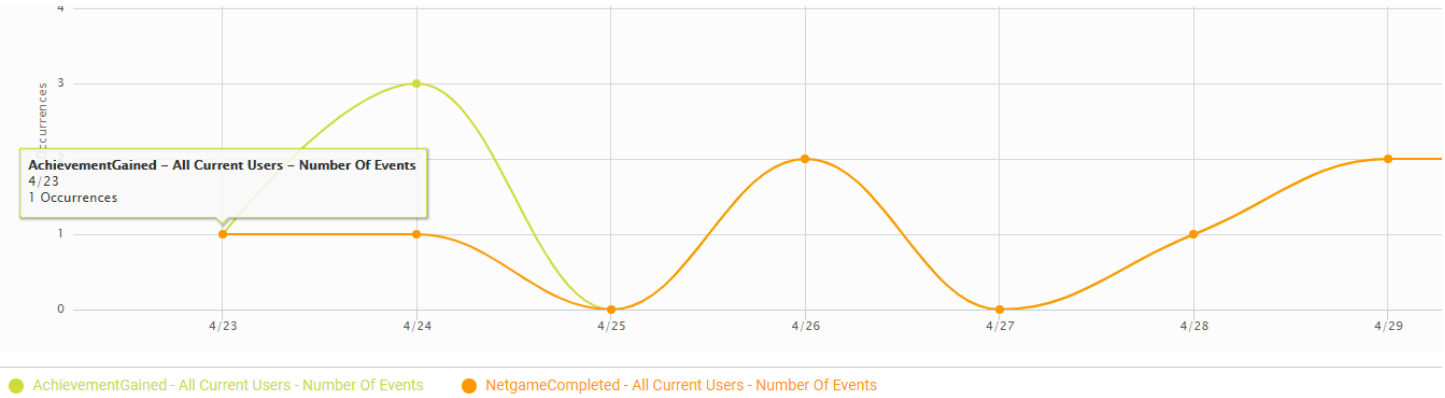
We have quickly realised that the SteppedUp event is rather useless, so we removed this event.

There have been a couple of things we learned from the analytics. The analytics help us find problems in our code and we were able to solve them. In the following graph you can see the amount a door is opened (orange) and the times a button is pressed (black).



What you can see is that there are way more DoorOpend events than ButtonPressed events. Currently the only way to open a door is by pressing a button. This mean, that there is a problem in the door mechanic. A door is opened multiple times per button press. We were able to resolve this problem that we found because of the analytics.

In the following graph you can see the amount an achievements was gained (yellow) and the amount a NetGame is created.

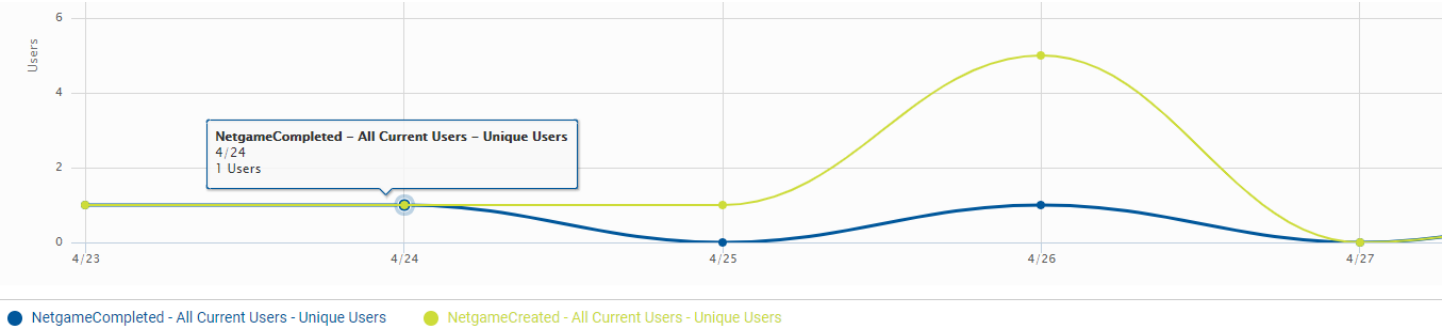


We can see that most of the time both graphs are the same. The reason for this is, because if you complete the first NetGame in the game, you will unlock an achievement. this analytics gets interesting at the points where the graphs are no the same. Currently

you can unlock two different achiements. But as you can see at the 24th of april there were three achievements and only one NetGame completion. There are two different NteGames in the gameworl and at the end of each, there is a different achievement. So this is rather unusual. There should either be 3 NetGames completed or only 1 achievement gained. Interesting to note is also that a single player gained all the achievement. This can be seen if you display the number of unique users for each event (this graph is not shown here since it doesn't give any additional information).

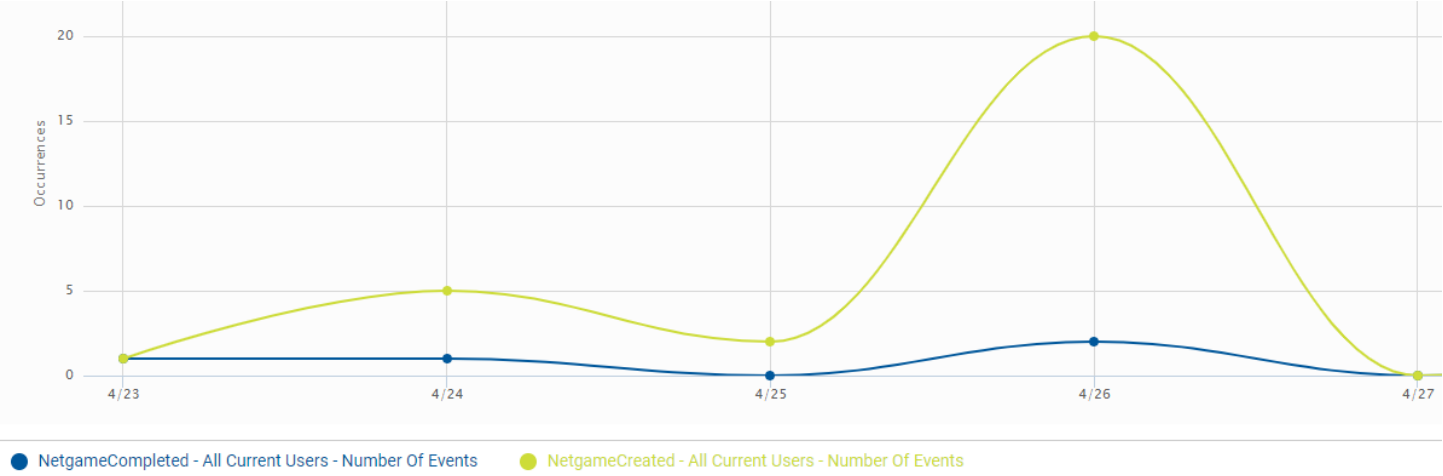
There are two things that could have happend here. The player could have cheated and got more achievemnts that are currently in the game. We consider this to be very unlikly. Because why would you spend the time and effort to cheat more achievements that are in to game. So we discarded this thought and went on to the probable reason. There is another bug in the game that lets you gain an achievemnt multiple times. This Bug was fixed as well.

What the analytics showed us so far is that the game is not perfect (yet). They helped us improved the game and this in itself is a good gain from the analytics. But there were a couple things we learned from the analytics game design wise.

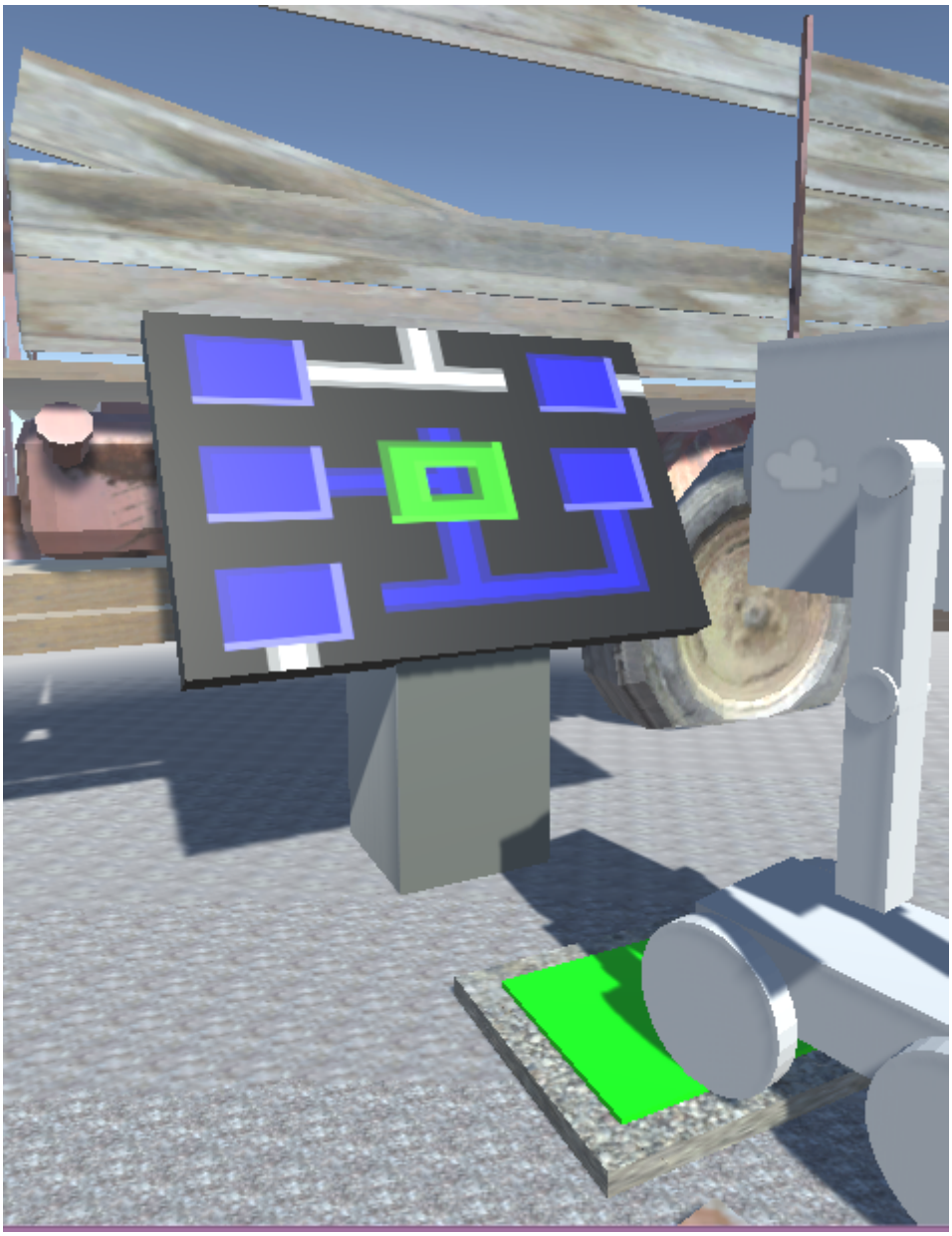


In the above graph you can see unique player count that created a NetGame (yellow) and the unique player count that completed a Netgame.

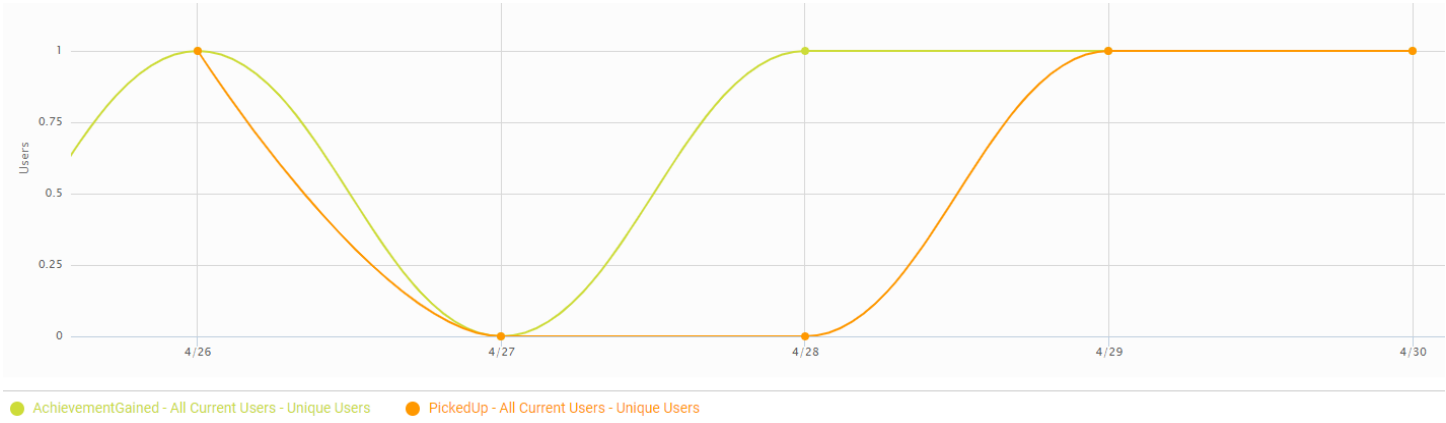
We can see there is a difference inbetween both. This means that the Netgame was not finished by some of the players at all. If we now take a look at the graph below which has the same categories but shows the number of events rather than the unique player count. We cann see, that the player have created a lot more NetGames than they solved.



The question is why do the player not solve the first NetGame they created. This could have a couple of reasons. First of the NetGame, that was randomly created, was too hard for them. Or the player simply didn't know how to solve it. The NetGame of the tutorial is almost the soimplest NetGame you can create (as seen below). The only things you could do to make it easier is to make it a 2x2 NetGame. Since we would argue that the 3x3 NetGame of the tutorial is quite easy. The probable solution for this problem is to give the player more help with the NetGame and the interaction with it. Currently you only get see all the keys you can use in the game befor you started the game. We might have to consider changing this in some way. More regarding this change in the ovreall evaluation of the analytics.



The last categories we thought are interesting is achievements gained (yellow) and cubes picked up (orange) by the unique user count as well.



What we can see is that there are a couple of players, that learned the new pickup skill (the second skill) and managed to use it. But there are a some that learned the new skill but did not use it. This most likely means that they didn't knowhow to use it. This could be used with some more help that will be explained in the overall evaluation.

Overall Evaluation of the analytics

The analytics have showed us some problems we have to tackle. The tutorial has only partially achieved the goal of teaching the player how to play. It has to be redesigned to be easier to understand. It didn't help the player enough to get them into the game, so this design pattern was used inefficiently.

The other design pattern we used were achievements. We have seen that the users earned some achievements and even unlocked multiple. Since part of the playerbase went though the entire game we would say that the achevements were successful, but in order for them to fully show there potential we need more of them and a bigger game where more of them can be earned.

After we analysied our current implementation we have the following ideas to improve the game:

We initialy didn't want any explanations in the game at all but created a displayed keymap to help the player out in the beginning. As we now know, this is not enought and we need more help for the player. We thought about a help feature that highlights certain thing on the screen. For example if you created a NetGame but do not start to solve it in a given time, a computer mouse will be shown. This might help the player know that he has to use the mouse to solve the NetGame. This will also be ablt to help with new skills you learned. So if you unlocked a new skill. A small schemantic of the new skill could be shown. F.e. if you learned how to pick up a cube, you will see a schemantic of a root that drives to a cube, presses "space" and lifts the cube. This schemantic will be shown as solng as the player didn't pick up a cube for the first time.

It would make sense to provide the player with these helping features even after they should lready know them. Because if you return to the gae after a breal you might now remember the keys you need. So we thought about giving the player the option to show the helps on some sort of menu or at a certain place in the game (like a traing ground for the abilities).

Design Patterns and Achievements

As we already stated, we have implemented two design patterns. A tutorial and achievements. And as shown in the overall analytics evaluation especially the tutorial needs some work. We discussed the way we would improve the turorial and achievements already, so we thought about other design pattern that are not yet implemented but would fit the game as well.

We would like to expand on the illusion of control our game gives the player. The player needs more “choices”. So we will need more variety in level and environment in general. But the player will only be able to play the game with more skills he can unlock. For this we need more skills as well and more level entry obstacles. This concept could be improved with a skillpoint system as well. Lets say the player unlockes an upgrade, rather than getter the upgrade the player will get an amount of skillpoints. The player can invest these in the skills he likes. This results in more control for the player. And we could also make more expensive skills where you need to finish mulitple NetGames in order to unlock a single skill.

This illusion of control is next to the tutorial and achievements the main design pattern we want to fo us on.

Future Work

There are a couple different thing that would be could be implemented in the future. We have menntioned alot in the upper chapters and only want to summerize here.

First of there is the player help as proposed in the overall evaluation of the analytics. This includes a redesign of the tutorial area to try and make it more self explanatory.

Another thing is the expansion of the game world itself. Currently the gameworld is pretty small with only a couple of levels. We would like to expand the world so that there are more level, more achievements and a greater challenge in the more difficult level.

In order to add more depth for the player we are thinking about adding a skill tree where the user can choose what skill he wants to unlock.

Something we haven’t considered yet is the final launching plattform. Currently it looks like this will be a PC only game, but we can see it find some players in the mobil market as well. So we might want to consider making a mobile build and testing it out on our phone to see how it works. If this test is satisfying this might be the market to go.

Things we learned

There were a lot of things we were able to learn during the course. There are a couple of obvious things we learn. For example we created a unity game, which improved our C# skills since unity uses C#. While we created new code, we had to refactor our own old code and improve it. We were also able to learn about modeling of characters and a game world in general. There have been multiple occasions were we learned how hard it can be to create the simpleset of a level. You have to consider alot of different thing, especially the “user that wants to break stuff”. We had to change our levels numerous times in order for them to provide the gameplay we are aiming at. Another thing we never really thought about was the time it takes to create a 3D environment. It take way more effort than we thought.

Since we created the game with unity we were able to learn about the ins and outs of unity. Unity provides a lot of useful features, but if you are not aware of how unity does certain thing, you can ran into problems.

Since we were working in a team, we were able improve our team work skills. This started with the planning of the project. We had to communicate a lot in the group. The use of the version control system git allowed us the efficently work together and we had the ability to tests out a couple different things using Git. We tested out GitHub pages to create this blog, and we were impressed how easy it is. Another thing we were not expecting is the fast you can make your game publicly available. We used the WebGL build feature unity provides to create a WebGL build. We were than able to upload this build GitHub pages and provide a way to play the game in a browser. If you click here or on the “Play it Now!” button you can test it out yourself!

Next to the game itself we had to think about the user that is supposed to play the game. We weren’t ablt the just code a game and call it done, we had to think about our users behavior and adapt accordingly. To help us understand our users we created some analytics. We learned alot about creating them and how you could use them to improve the game. Unity provides a free, easy to use analytics service that we could enable ad use to our own liking.

Contributions of team members

Contribution for this post and fixes

Hermann: 5h

Tim: 5h

Overall time investement

Hermann: 36h

Tim: 36h

Written on May 9, 2019