*Presentation title:* Node based world engine development inside the Unity3D game engine using multi-threading, compute shaders and instanced shaders - the story of, what went wrong, what went well, the inbetween and future + tips and tricks for creating your own project that extends the standard Unity3D functionality

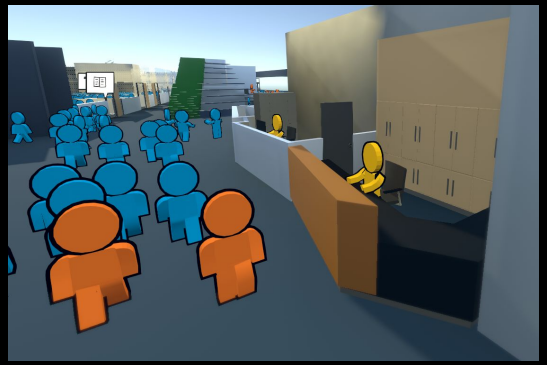
*Presenter name:* Kristo Männa

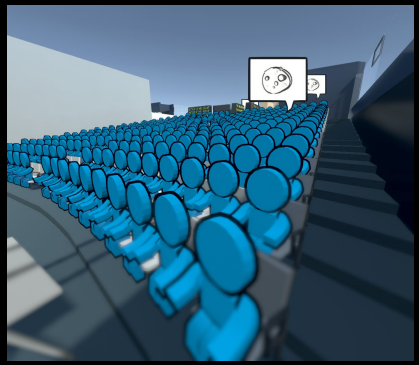
*Date:* 2020 May 15

# About me

Kristo Männa

- Unity3D generalist
- Solo game developer
  - Made 2 games, third in the making
- Also worked on DBV





# Meta

Focus:

1. World loading solution for open world games
2. Creating a project with Unity3D

NB! Presentation notes:

- Questions *(you'd better have some...)* -> ask immediately without hesitation
- Topics from beginner to advanced so if I start speaking Unity jargon let me know

# Problem & Existing solutions

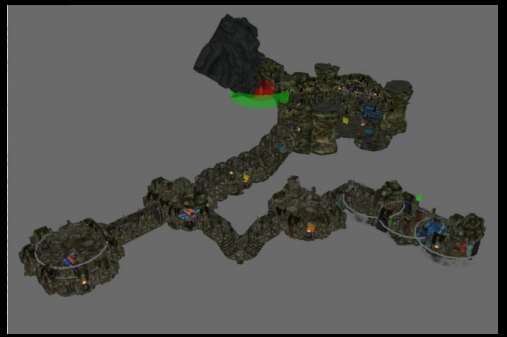
Problem: Can’t have the whole world loaded at once

Existing solutions:

- Levels -> Split world into levels and load 1 level at a time
- 2D chunks -> Split world into 2D grid of chunks, load close to player
- Mix -> Sometimes use chunks (outside) sometimes levels (indoors)

*Game examples in order: Mario, Minecraft, Skyrim*

My solution: Relative chunks (nodes)

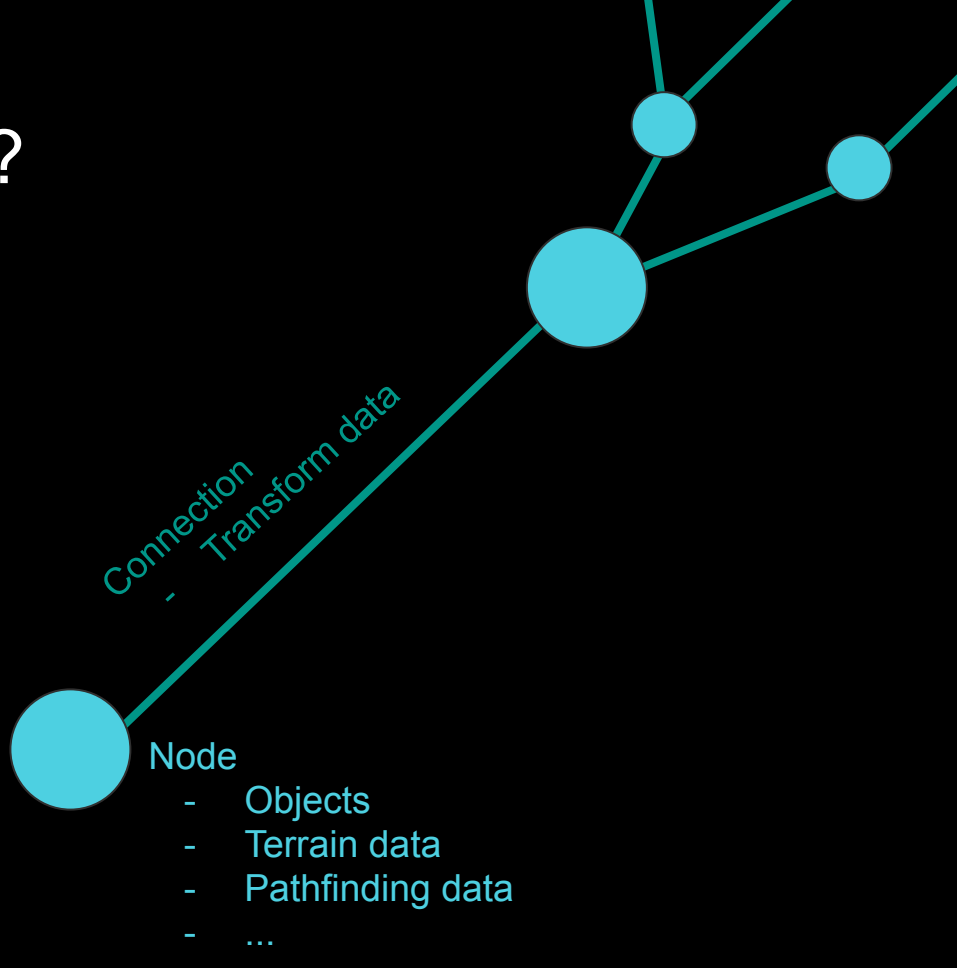
# Node based world loading?

Chunk based loading:

- Chunks have absolute position

Node based loading

- Nodes have connections
- Connections have transforms



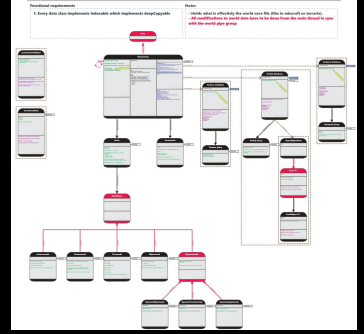
# So lets make it!

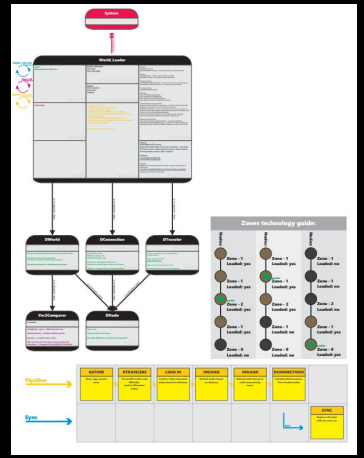
My reasoning:

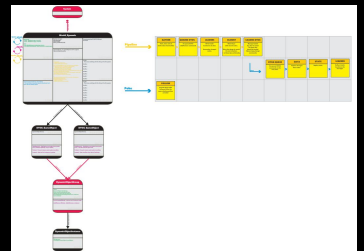
1. Make plan
   a. Start with low res. and work your way up to high res.
   b. Don’t plan too little nor too much
   c. KEEP. IT. SIMPLE.
2. Follow plan
   a. Plans change so keep your wits about you

Driver drives car into lake because GPS

https://abcnews.go.com/WNN/video/gps-leads-driver-lake-39137655





# Planned*(and implemented)* project structure overview

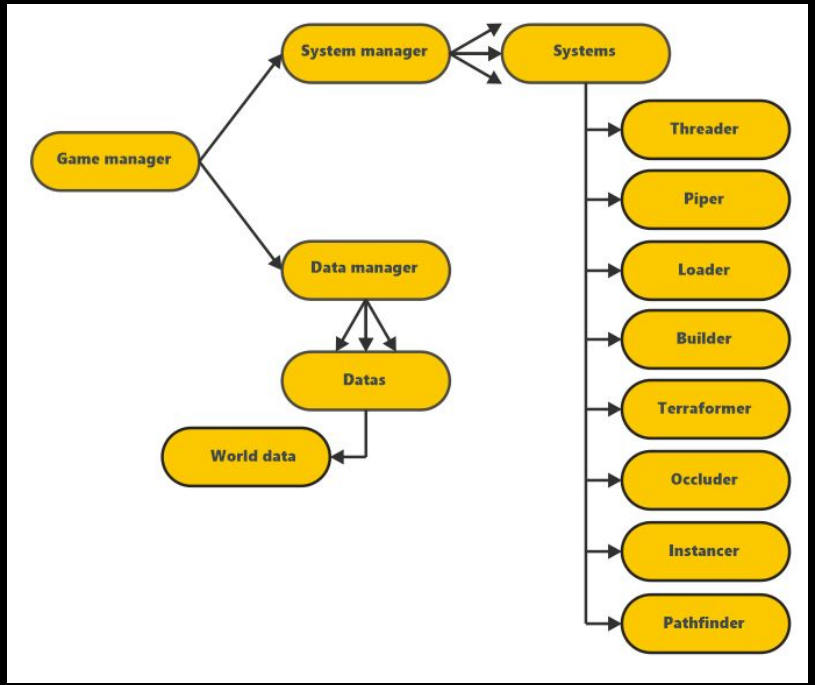
You have your Data:

- Saved to disk
- Runtime state is generated from this

and

you have your Systems:

- Operates on runtime state
- Each does one biggish task
- Make the world appear out of thin air



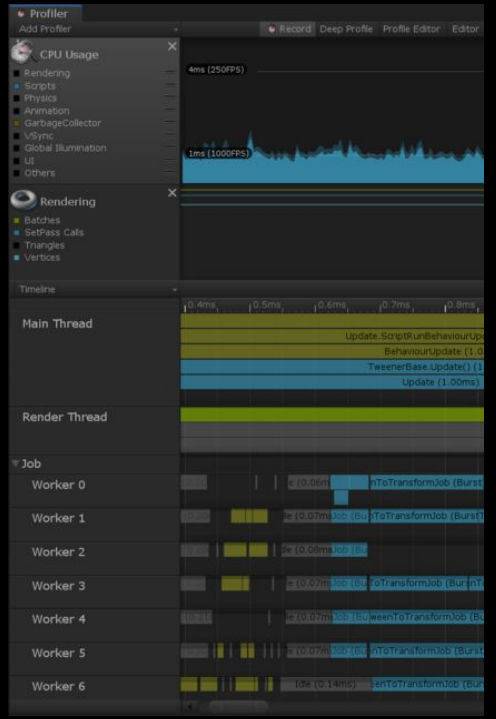
# Oke but I wanna make it fast!

Solution -> Multithreading!

NB!! It is very easy to create multithreaded code that is way slower than its single threaded counterpart

2 kinds of multithreading popular in games:

- Bulky separate threads eg. render, ai and main thread
  - Relatively simple to manage
- Microtasks eg. C# Parallel.For, Unity DOTS -> Jobs
  - Difficult to manage but can be extremely fast



# My multithreading solution

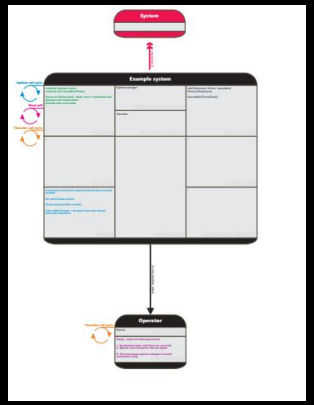
Custom worker thread pool

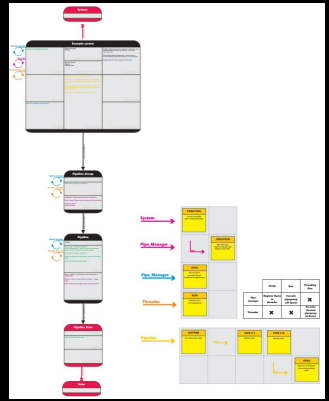
- Does max 1 cycle per frame
- Repeats jobs

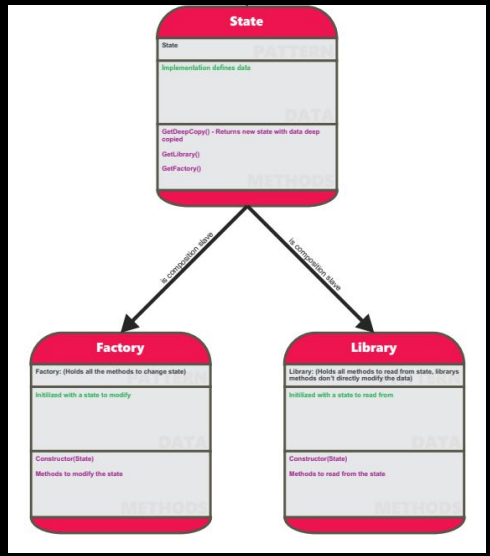
Problem: Need to read and write the same data

Solution: Pipelines and groups of them

- 1 pipeline for each system
- Either Working or Syncing





# Also going to need custom tools...

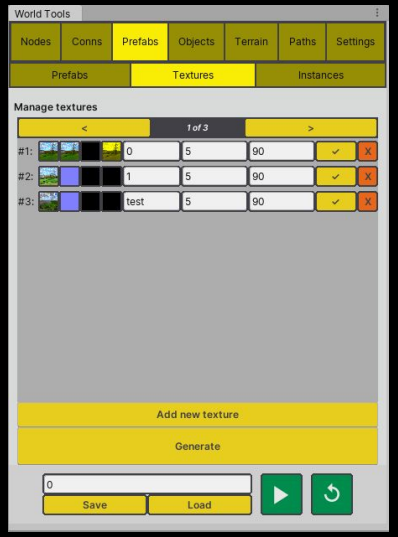
Custom tools in Unity -> Extending the Unity editor

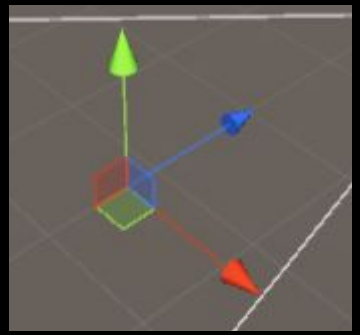
Recommended ways to do this:

- Custom inspector
- Custom window
- Handles

Other ways:

- ImGUI
- UIElements (alpha)



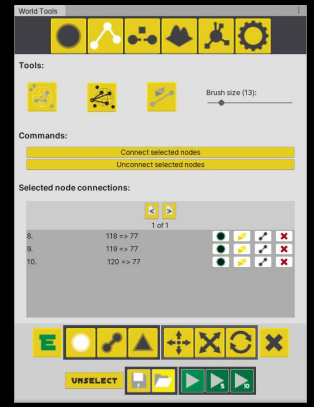



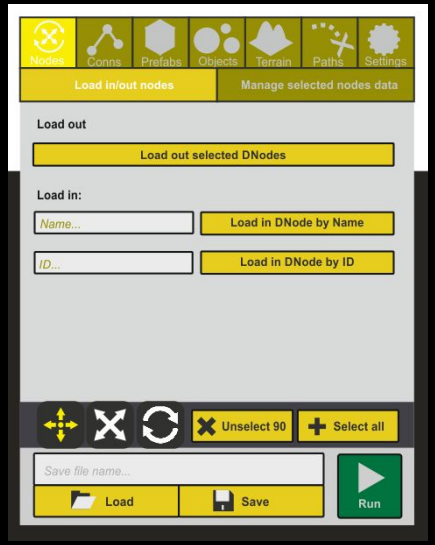
# Node based world editor

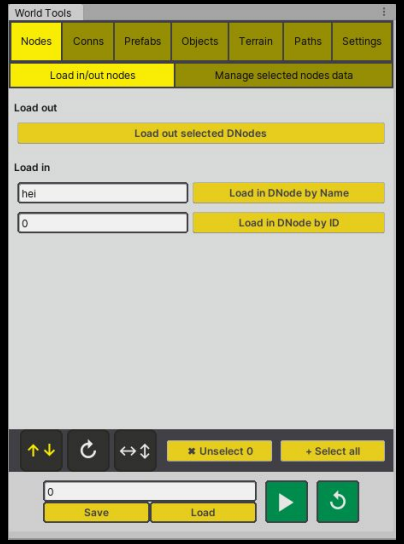
1. UI mockup in CorelDRAW
2. Used UIElements to create layout
3. Created custom editor window to render it

Tips:

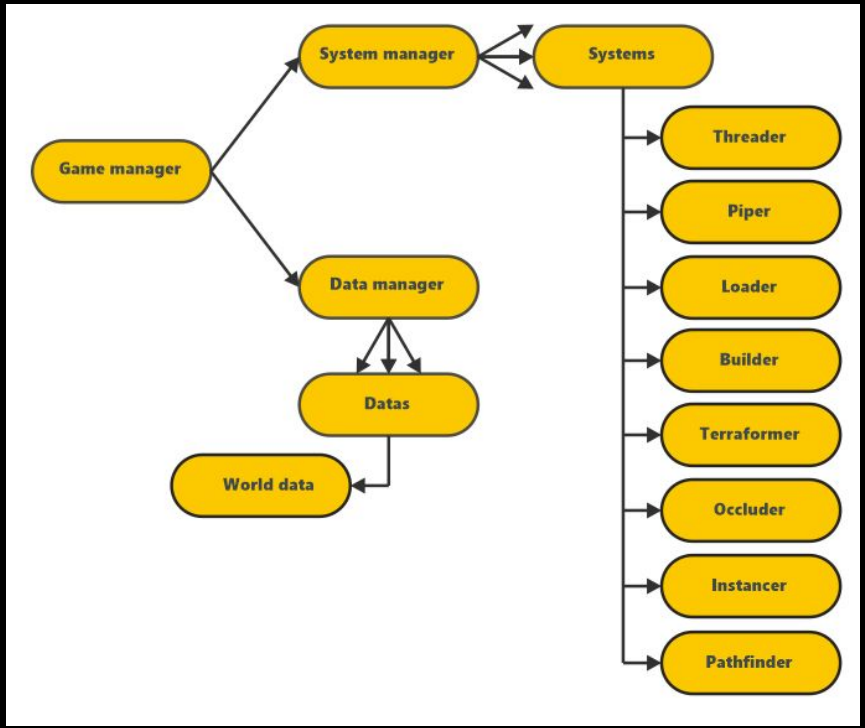
- Tools programming is deceptively simple - make sure to plan resources to it
- Separate layout & functions



# How does a run cycle look like?



# Terrain rendering

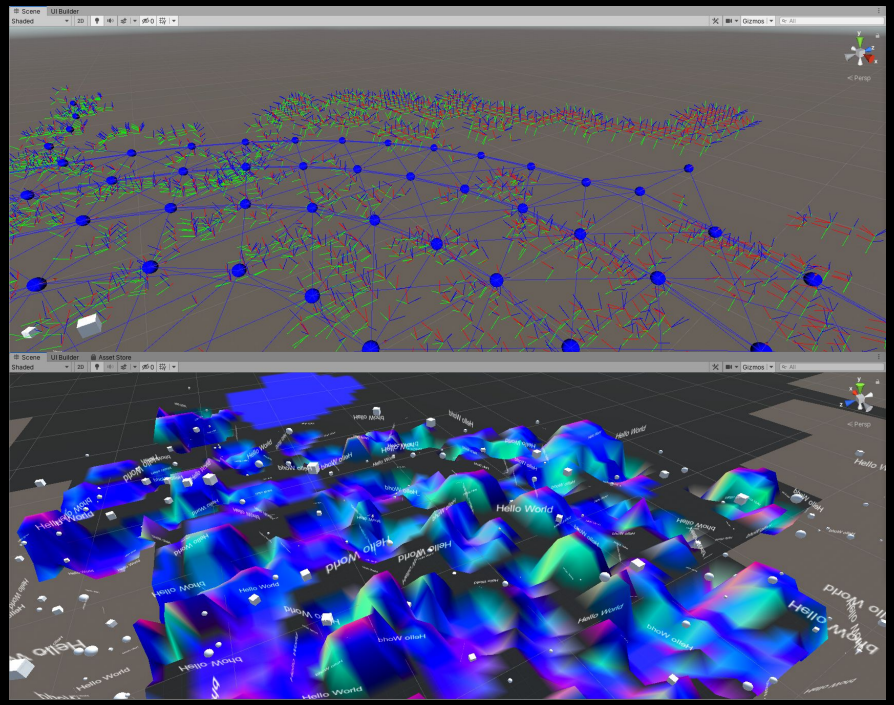
Loaded nodes -> Terrain nodes -> grid of terrain cells -> planes

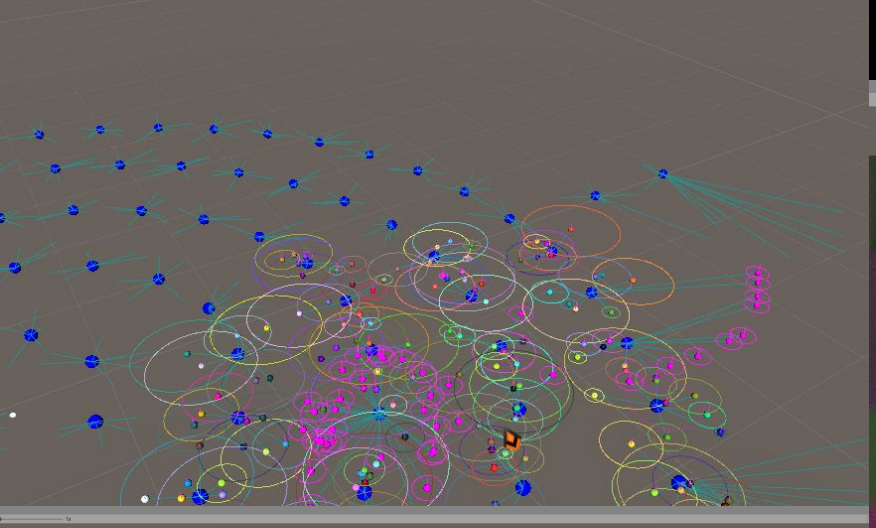
Original goal: Normal terrain with triplanar blending

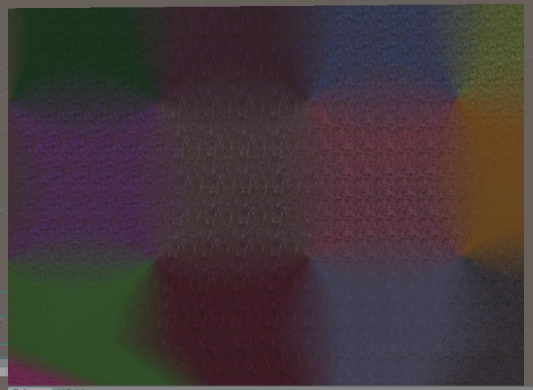
What went wrong?...

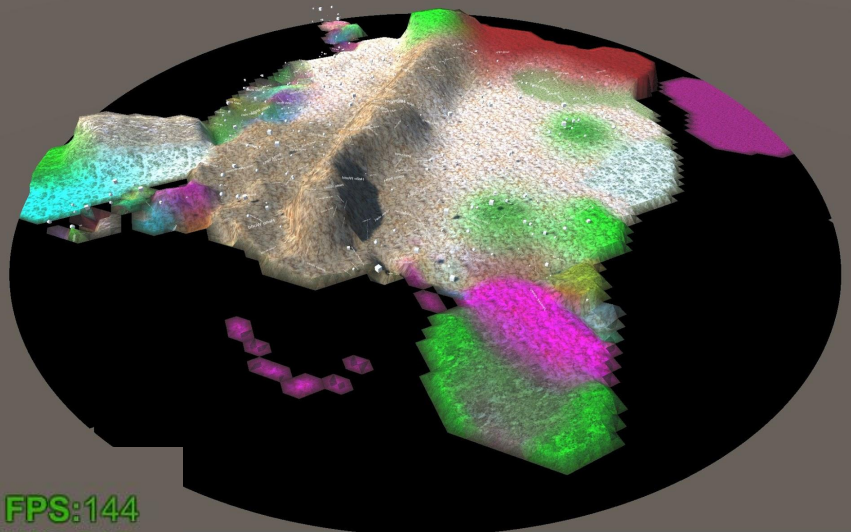
- Too much data
- Too many operations
  *(4 x 3 x 2 x 3 -> 72 texture samples per fragment)*
- Blending
- Careless mistakes

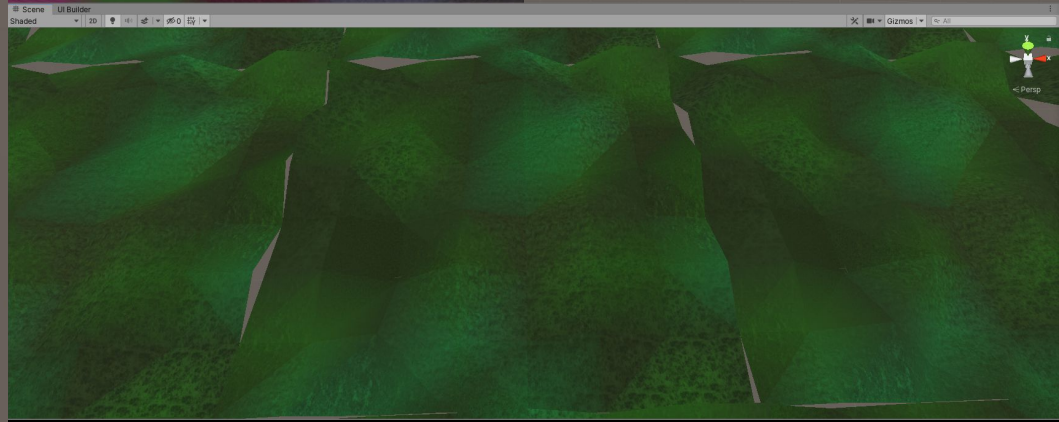
Solution: Pivoted to stylised terrain








FPS:144


Scene
UI Builder
Shaded
2D
Gizmos
Persp

# //Todo finish the project

Playable demo will be done for student project competition next week (24.05)

Sneak peak of what is still to come:

- Per voxel occlusion culling
- Small object instancing (Vegetation)
- Imposter support
- Support for dynamic objects (Physics objects)
- Pathfinding & Navmesh generation (A* or some moving target pathf. algo.)

# Conculsion

Keep it simple

Don't underestimate tools programming

Uber shaders are bad

# Thank you for listening/reading/participating in my talk titled:

Node based world engine development inside the Unity3D game engine using multi-threading, compute shaders and instanced shaders - the story of, what went wrong, what went well, the inbetween and future + tips and tricks for creating your own project that extends the standard Unity3D functionality

# Sources

What I based my Occlusion culler plan on: https://www.youtube.com/watch?v=U20dIA3SLTs

Pictures: www.Google.com