For original slides (with notes) see tinyurl.com/dok-gdc17

GDC

The Great Divide

Unique Visuals and Deterministic Gameplay in *Homeworld: Deserts of Kharak*

BLACKBIRD INTERACTIVE

Yossarian King
CTO, Blackbird Interactive

gearbox software

"How To Make a Multiplayer RTS Game With Unity and C#"

unity

C#

video

The Great Divide - Simulation vs. Presentation

Unique Visuals - Terrain & Aesthetic Physics

Deterministic Gameplay - Architecture, no `float`

Performance - LOD, C#

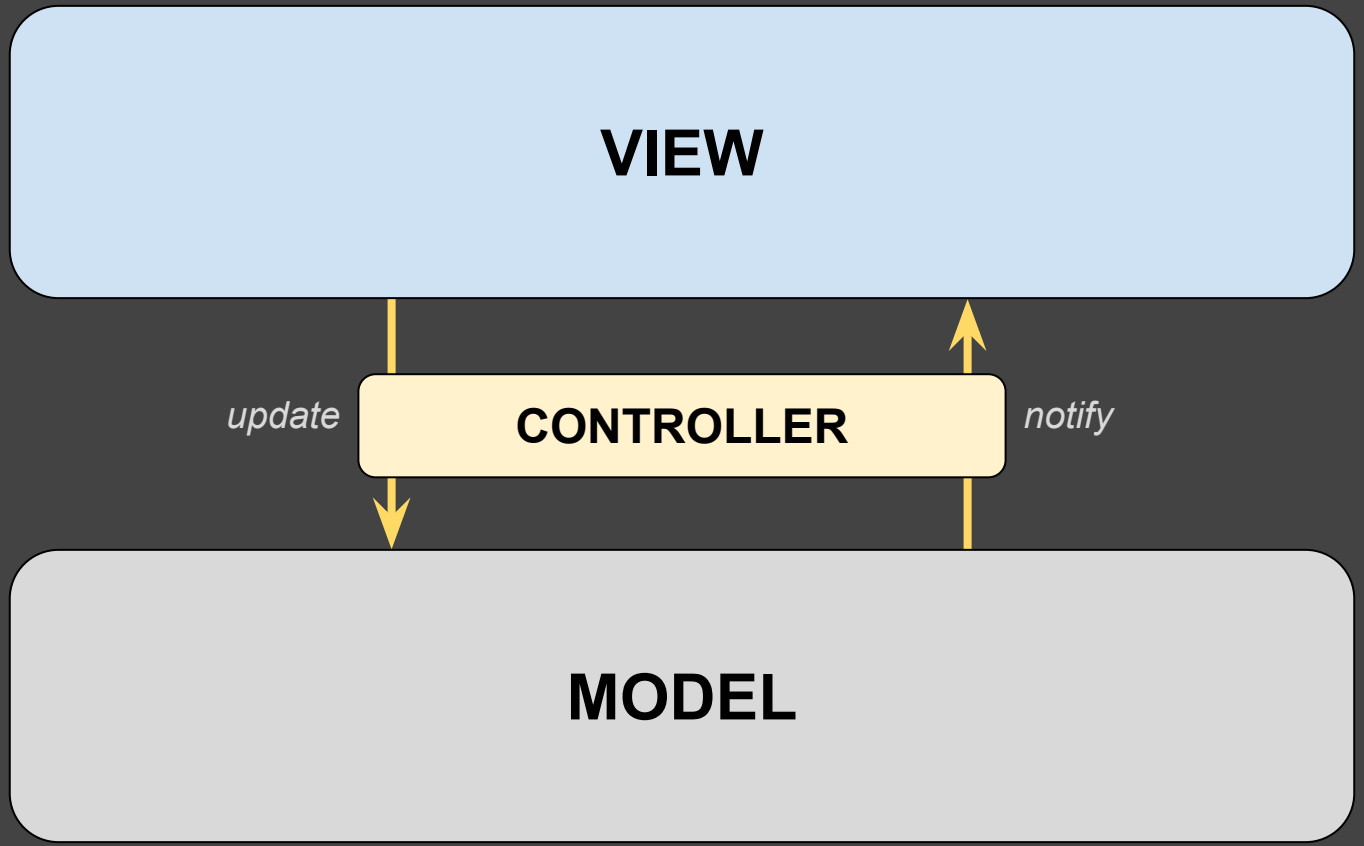
"The Great Divide"

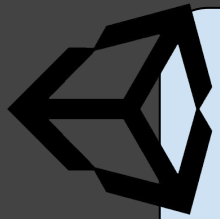
Simulation vs. Presentation

Gameplay vs. Graphics

Core game logic & systems (in pure C#)

Rich audiovisual presentation (in Unity)


VIEW
update
CONTROLLER
notify
MODEL



PRESENTATION

commands

CONTROLLER

state, events

SIMULATION

Unique Visuals - Terrain

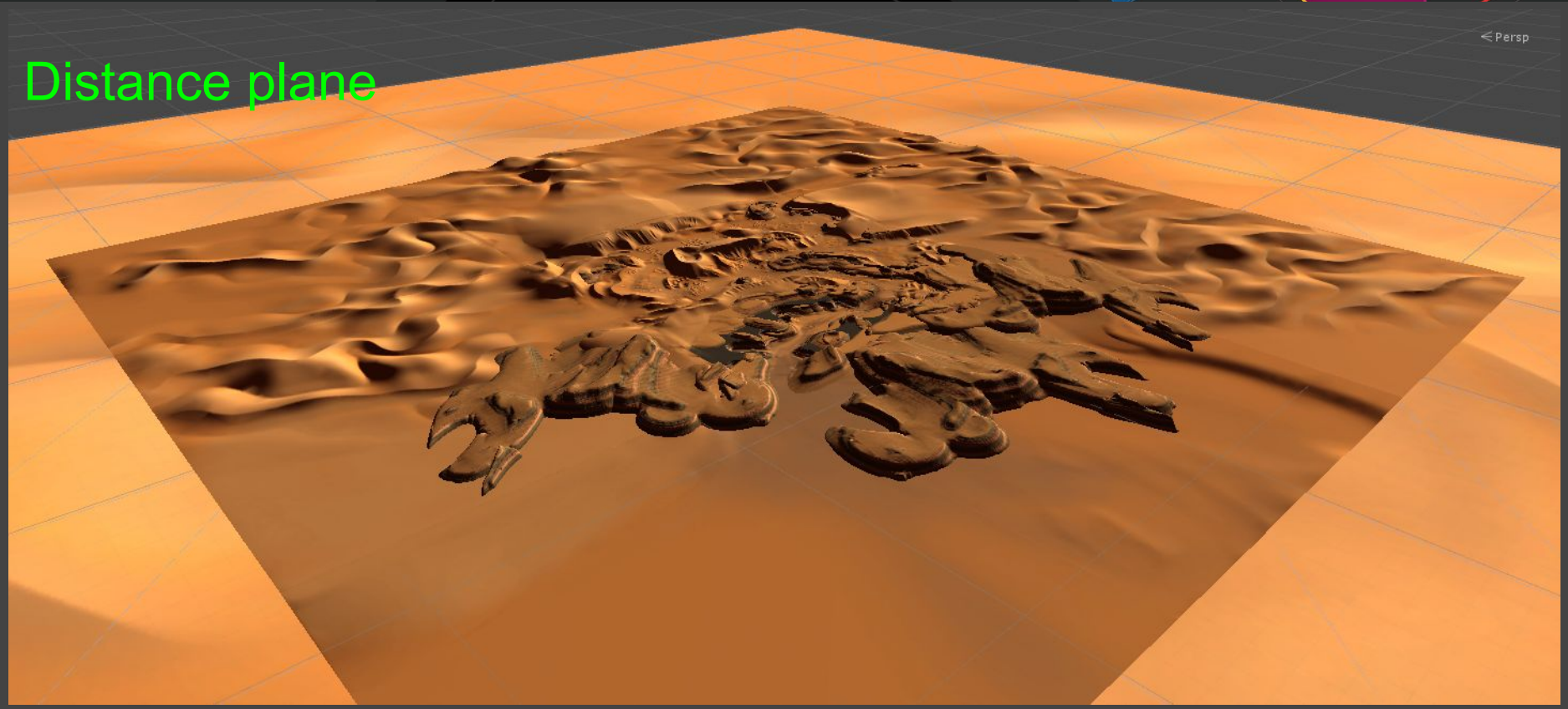
Terrain elements and layers

Deferred decal system

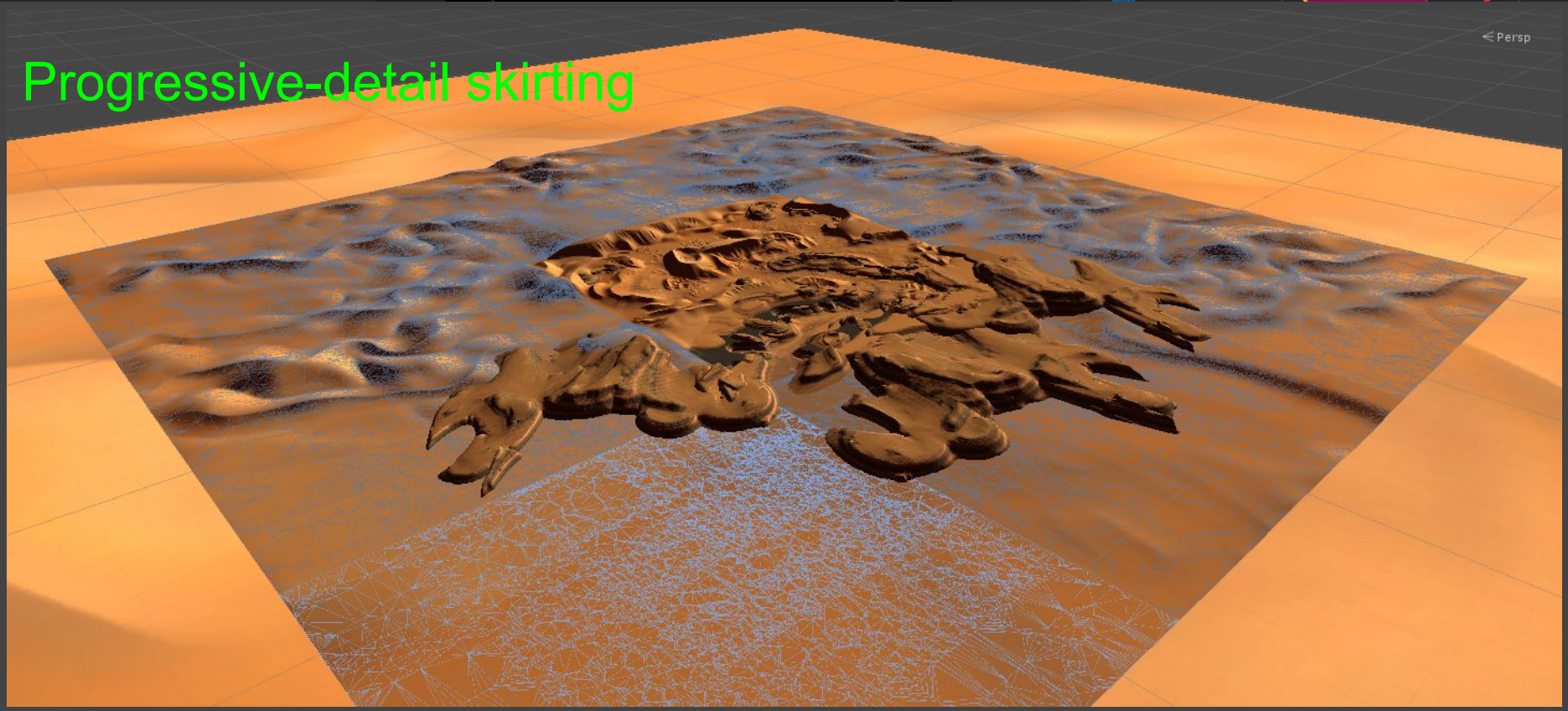
Authoring process

Terrain (barely) in the sim

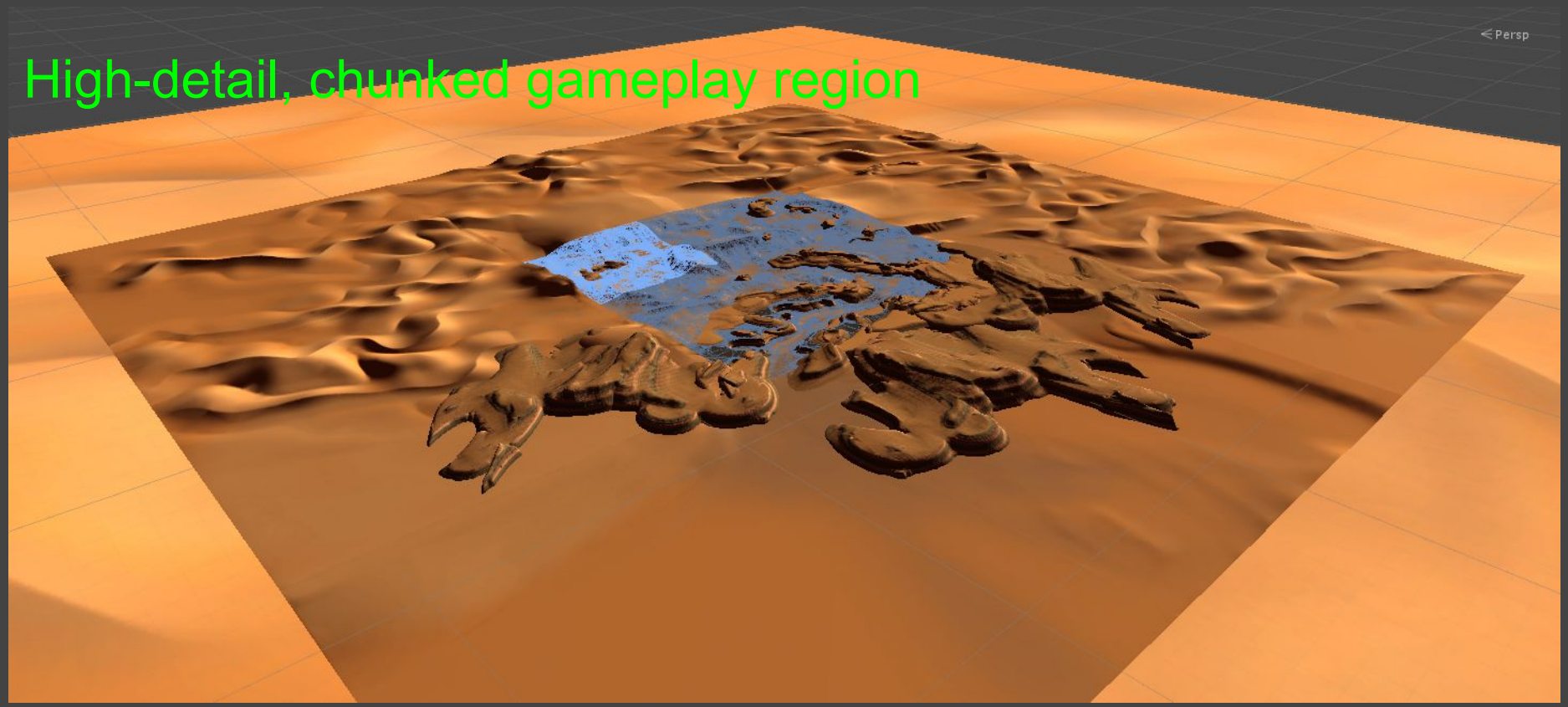

Sand


Persp
Distance plane

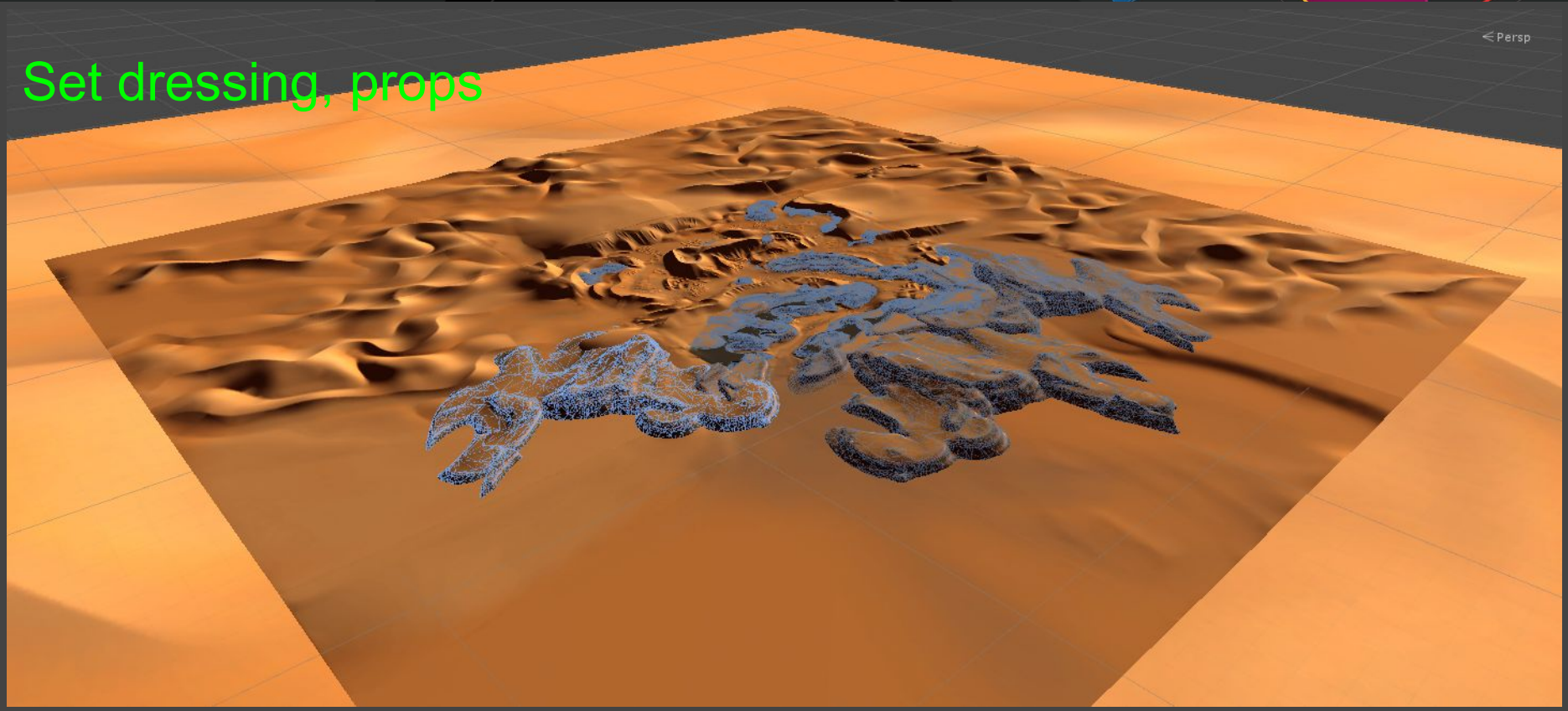
Progressive-detail skirting

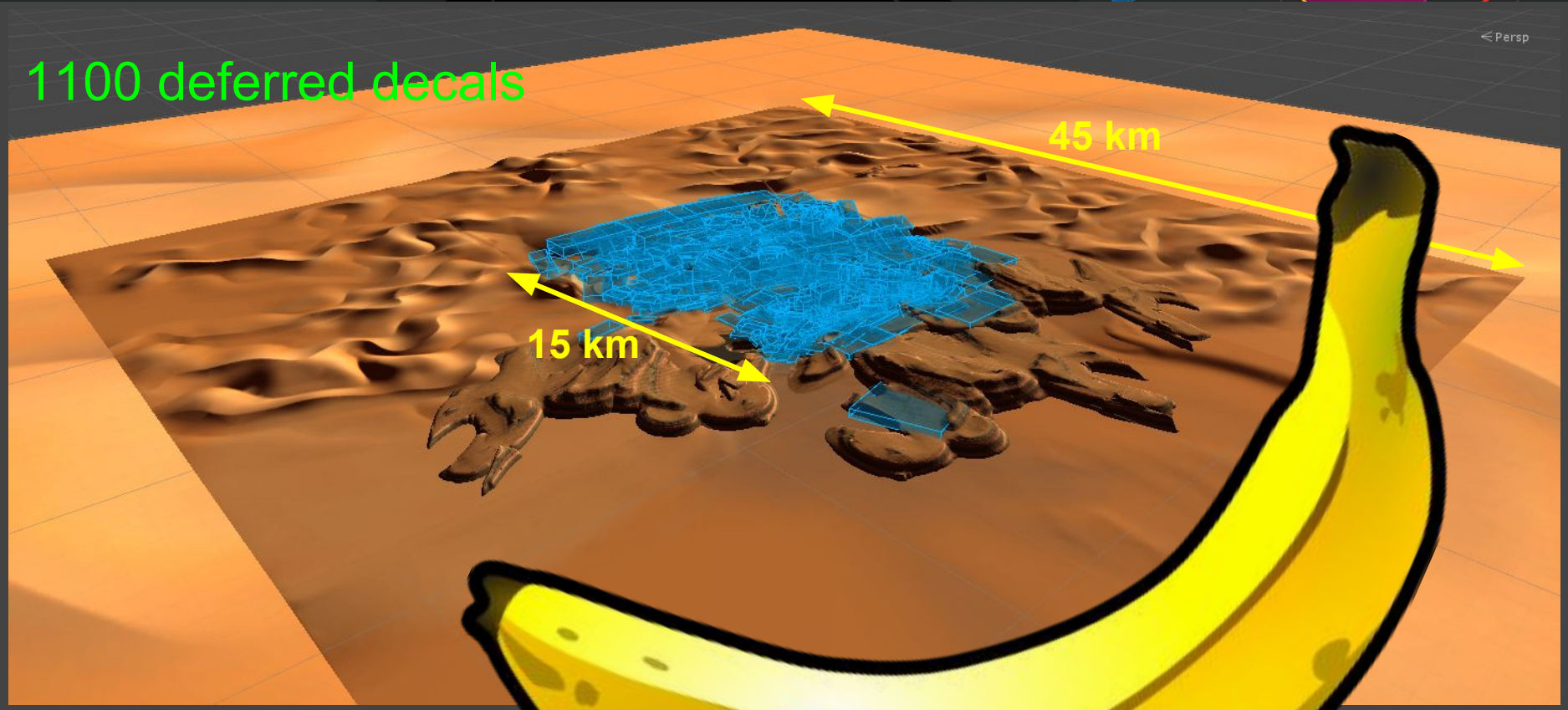


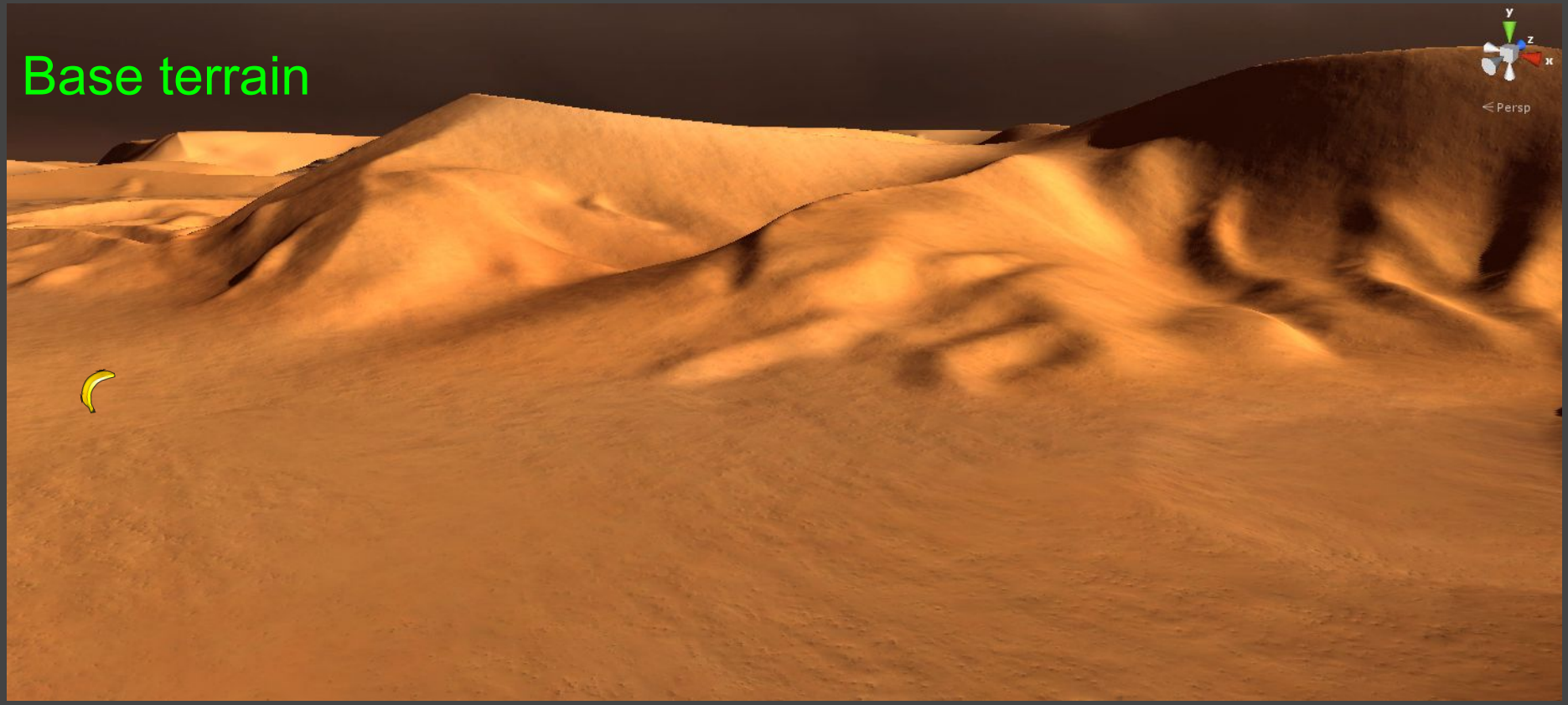
High-detail, chunked gameplay region



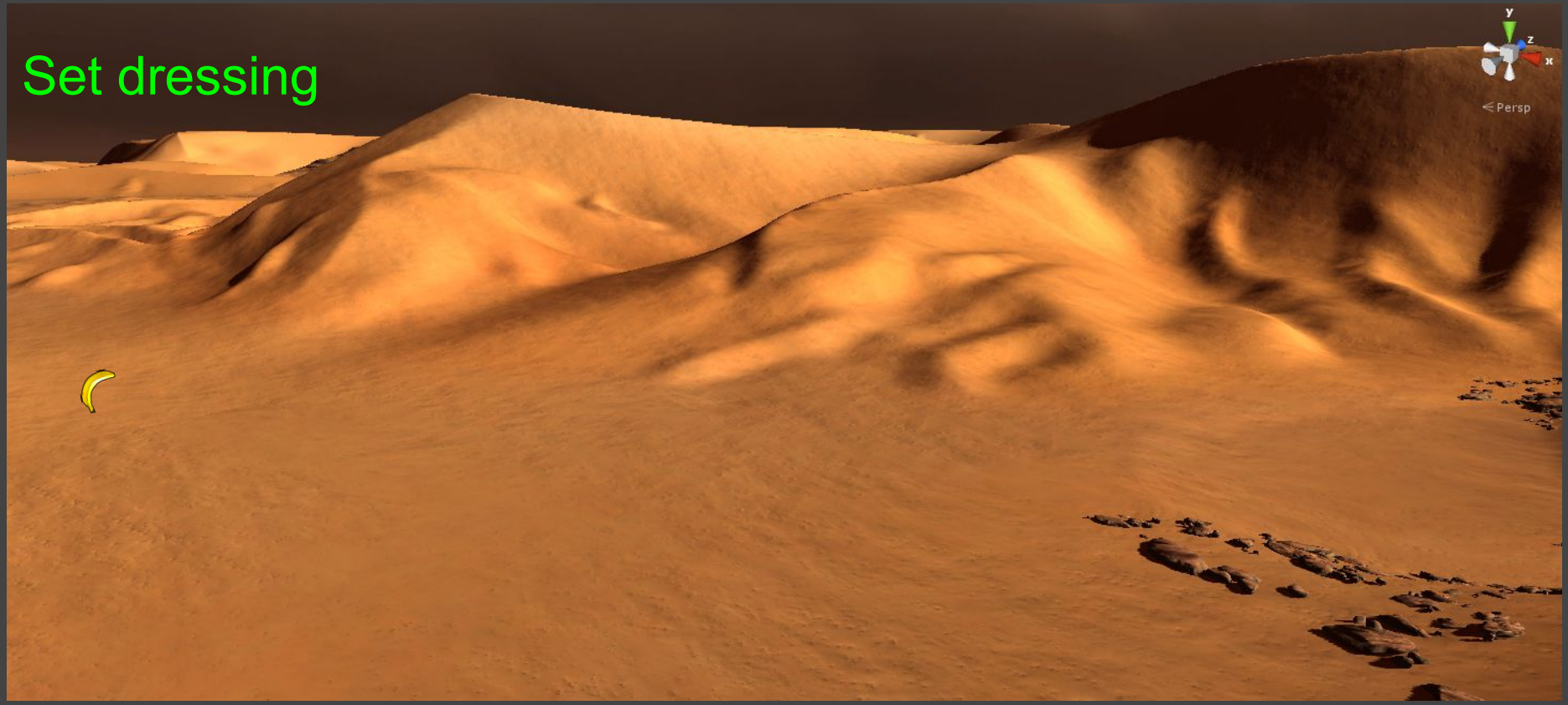
Set dressing, props

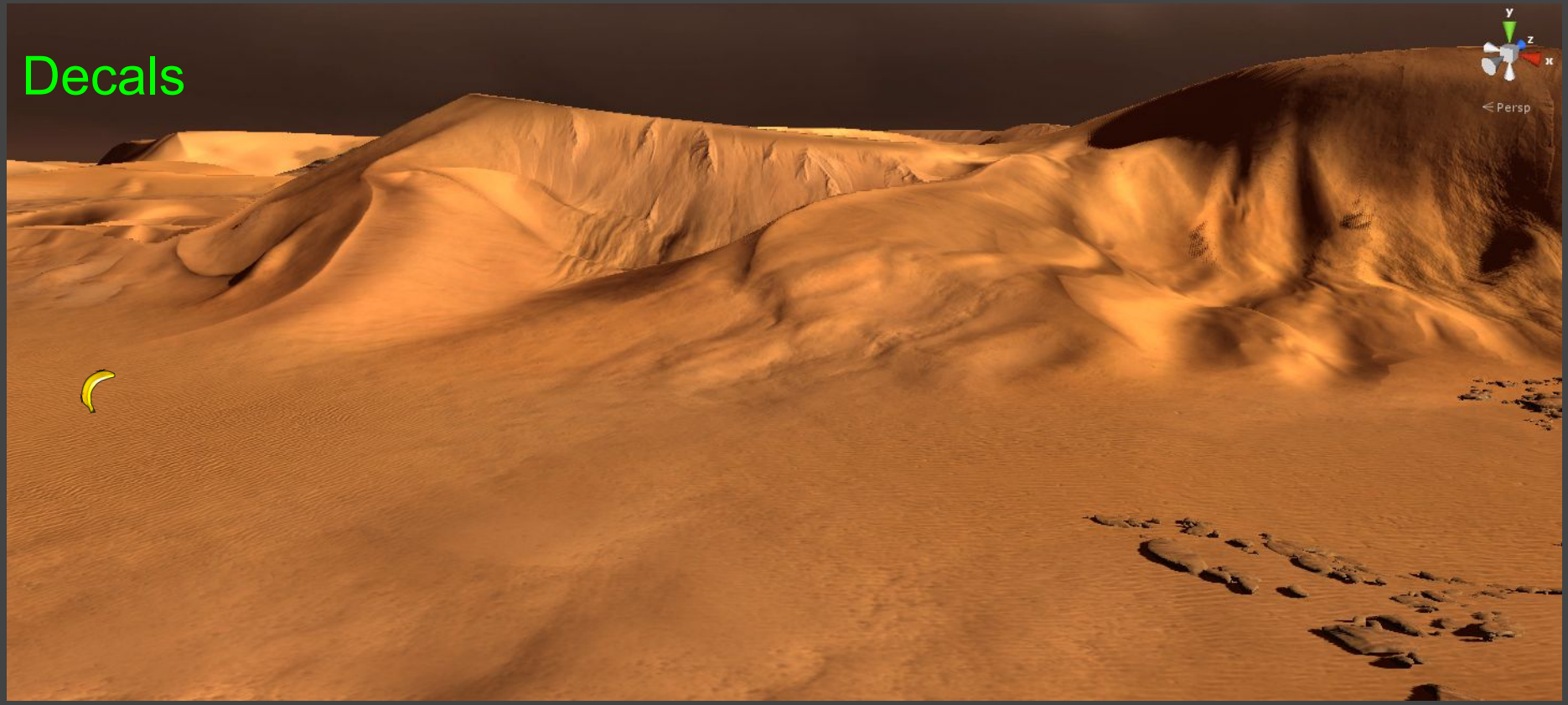


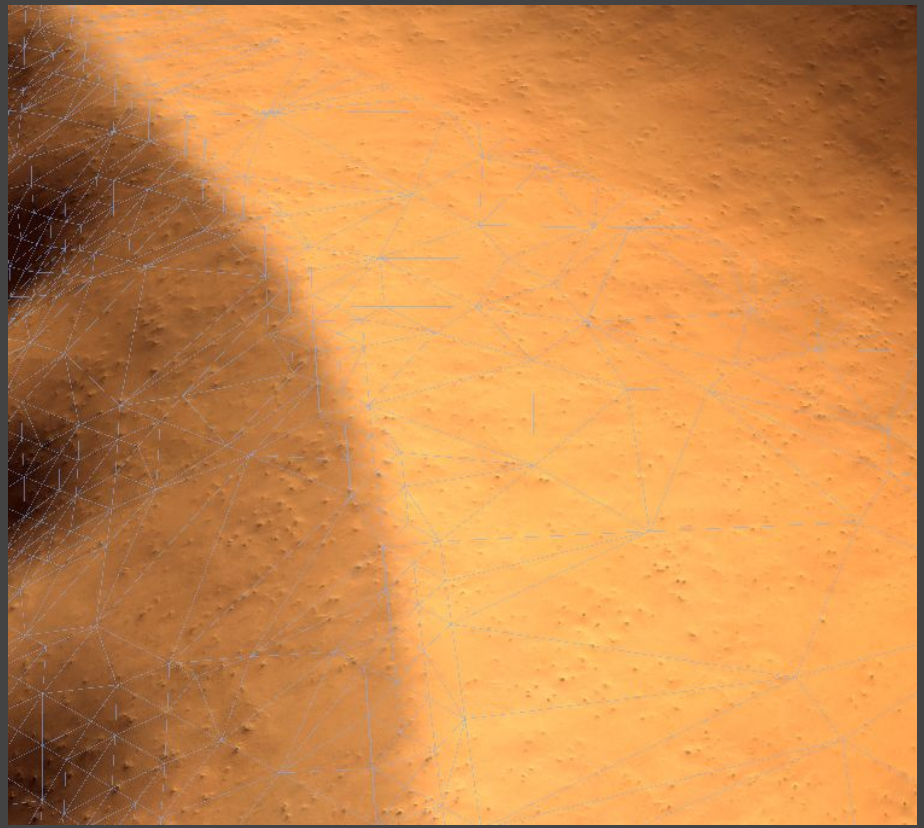

1100 deferred decals
45 km
15 km
Persp


Base terrain
y
z
x
Persp

Set dressing




Decals
y
z
x
Persp



Terrain-chunked

Shader: HWSB/Terrain/MixMap_Terrain_ColourMap

Control (RGBA)
Tiling X 1 Y 1
Offset X 0 Y 0
Select

Control 2 (RGBA)
Tiling X 1 Y 1
Offset X 0 Y 0
None (Texture)
Select

Base Colour Map (RGB)
Tiling X 1 Y 1
Offset X 0 Y 0
Select

Colour Map Blending. X=Fade in start, Y=Fade in length, Z=F
X 5000 Y 2000 Z 1 W 0

World Normal Map
Tiling X 1 Y 1
Offset X 0 Y 0
Select

World Normal Map Blending X=Fade in start, Y=Fade in length
X 5000 Y 2000 Z 5 W 0

Shininess (Specular S 0.106
Specular Color

Tex 1 (RGB) Specular(A)
Tiling X 50 Y 50
Offset X 0 Y 0
Select

Tex 1 Bump (RG)=Bump B=Mix
Tiling X 50 Y 50
Offset X 0 Y 0
Select

Tex 1 Avg Colour

Tex 2 (RGB) Specular(A)
Tiling X 27.27273 Y 27.27273
Offset X 0 Y 0
Select

Tex 2 Bump (RG)=Bump B=Mix
Tiling X 27.27273 Y 27.27273
Offset X 0 Y 0
Select

Tex 2 Avg Colour

Tex 3 (RGB) Specular(A)
Tiling X 25 Y 25
Offset X 0 Y 0
Select

Tex 3 Bump (RG)=Bump B=Mix
Tiling X 25 Y 25
Offset X 0 Y 0
Select

Tex 3 Avg Colour

Tex 4 (RGB) Specular(A)
Tiling X 75 Y 75
Offset X 0 Y 0
Select

Tex 4 Bump (RG)=Bump B=Mix
Tiling X 75 Y 75
Offset X 0 Y 0
Select

Tex 4 Avg Colour

Tex 5 (RGB) Specular(A)
Tiling X 100 Y 100
Offset X 0 Y 0
Select

Tex 5 Bump (RG)=Bump B=Mix
Tiling X 100 Y 100
Offset X 0 Y 0
Select

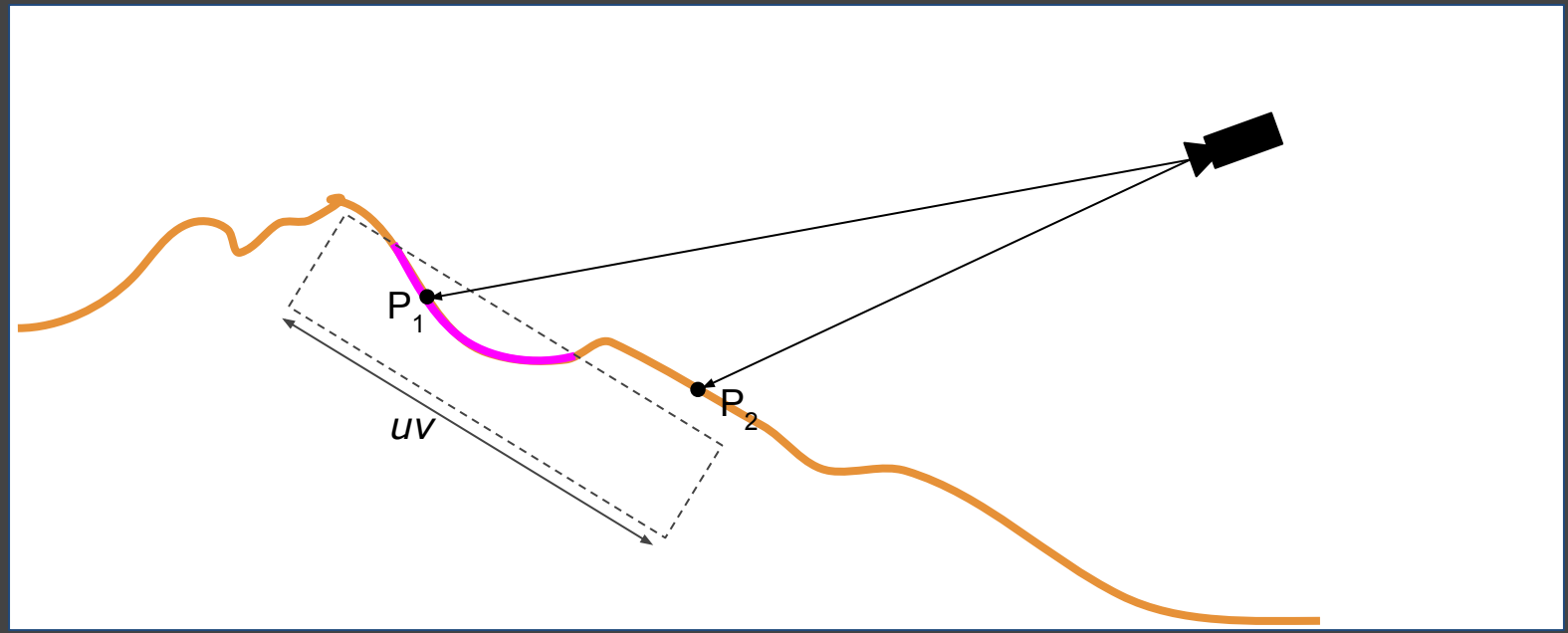
Tex 5 Avg Colour

Tex 6 (RGB) Specular(A). **Uses Tex 5 Bump**
Tiling X 0 Y 0
Offset X 0 Y 0
None (Texture)
Select

Tex 6 Avg Colour

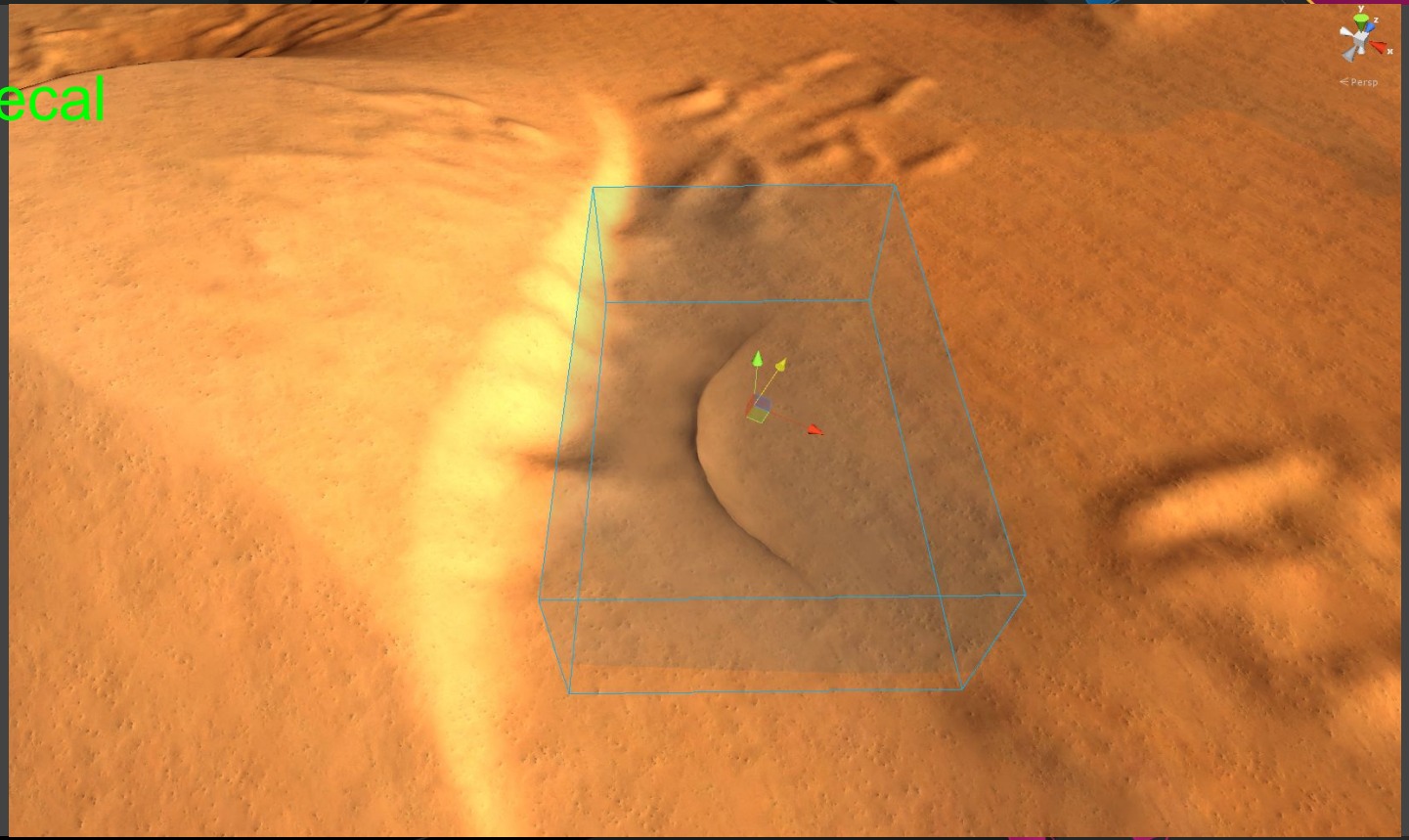
DEBUG Render Mode: OUTPUT_DIFFUSE

deferred decals

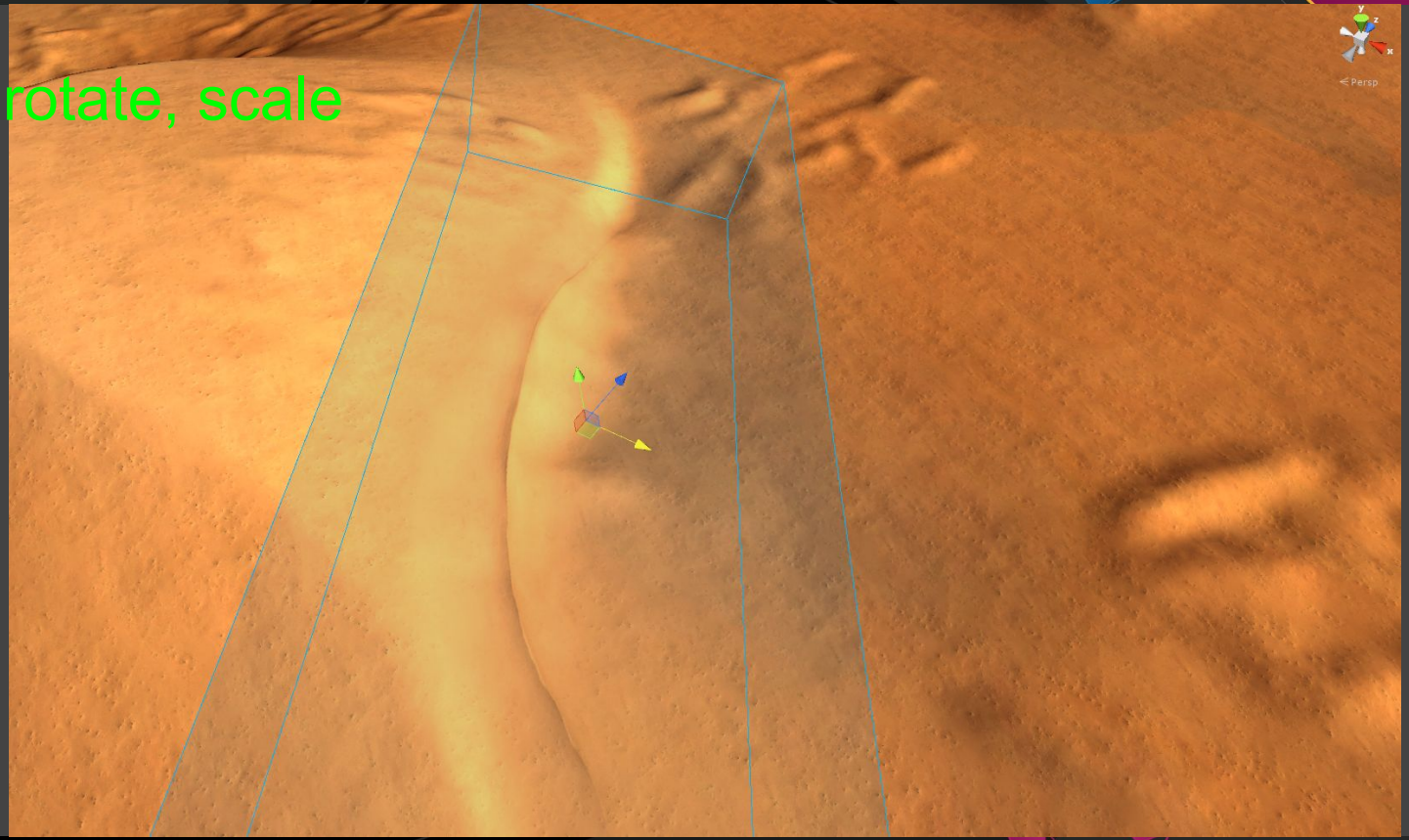

P1
P2
uv

Base terrain

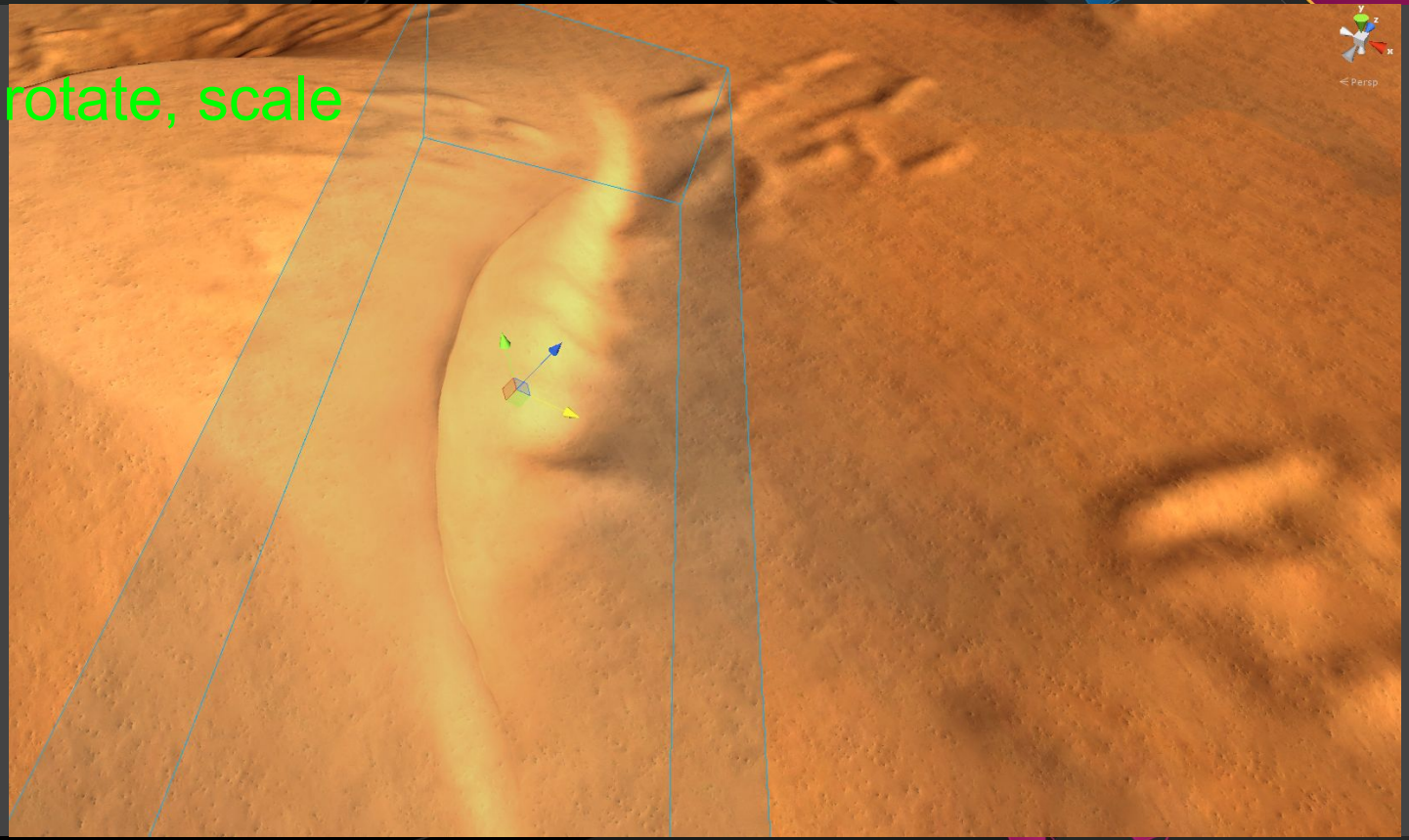
New decal



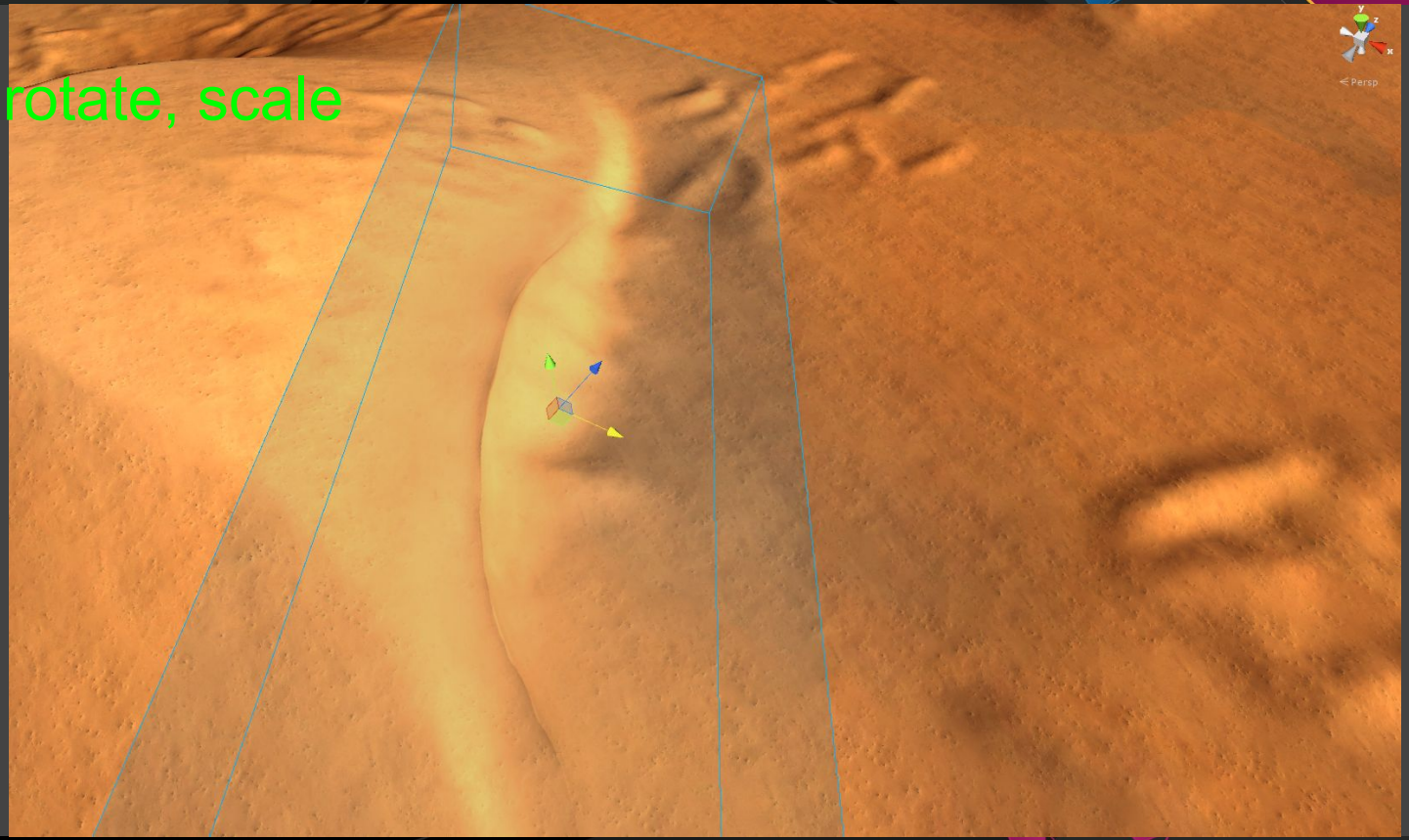
Move, rotate, scale



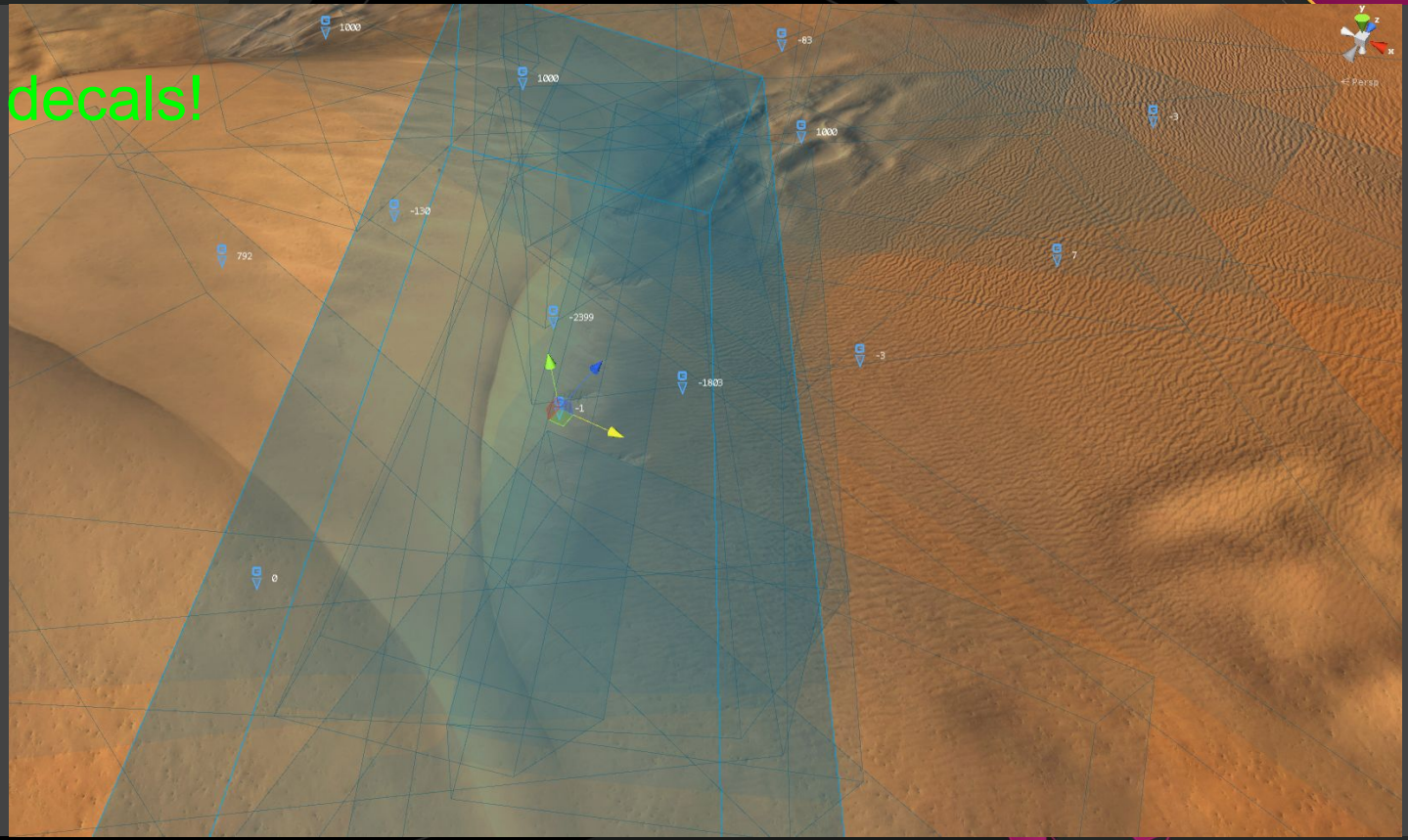
Move, rotate, scale

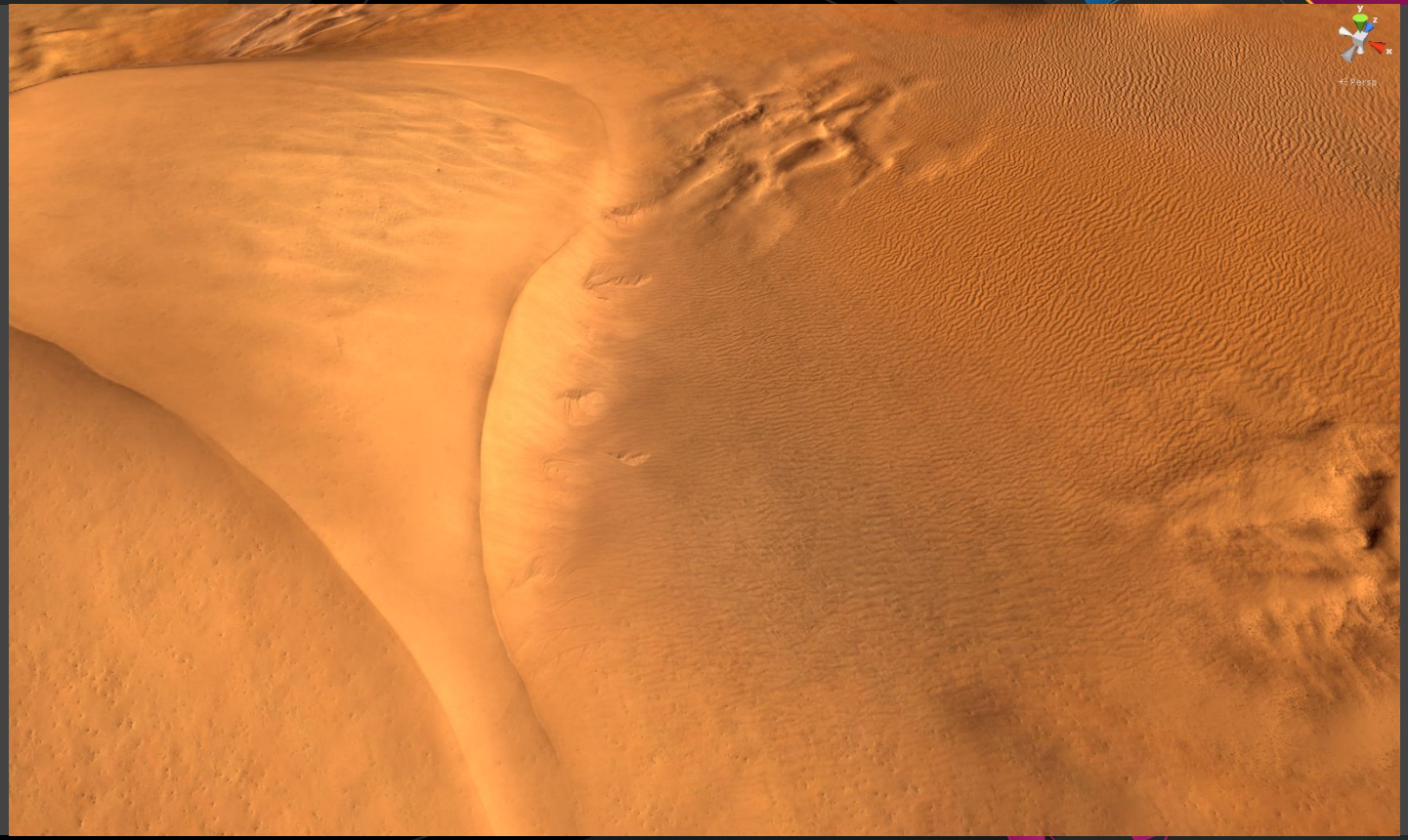


Move, rotate, scale

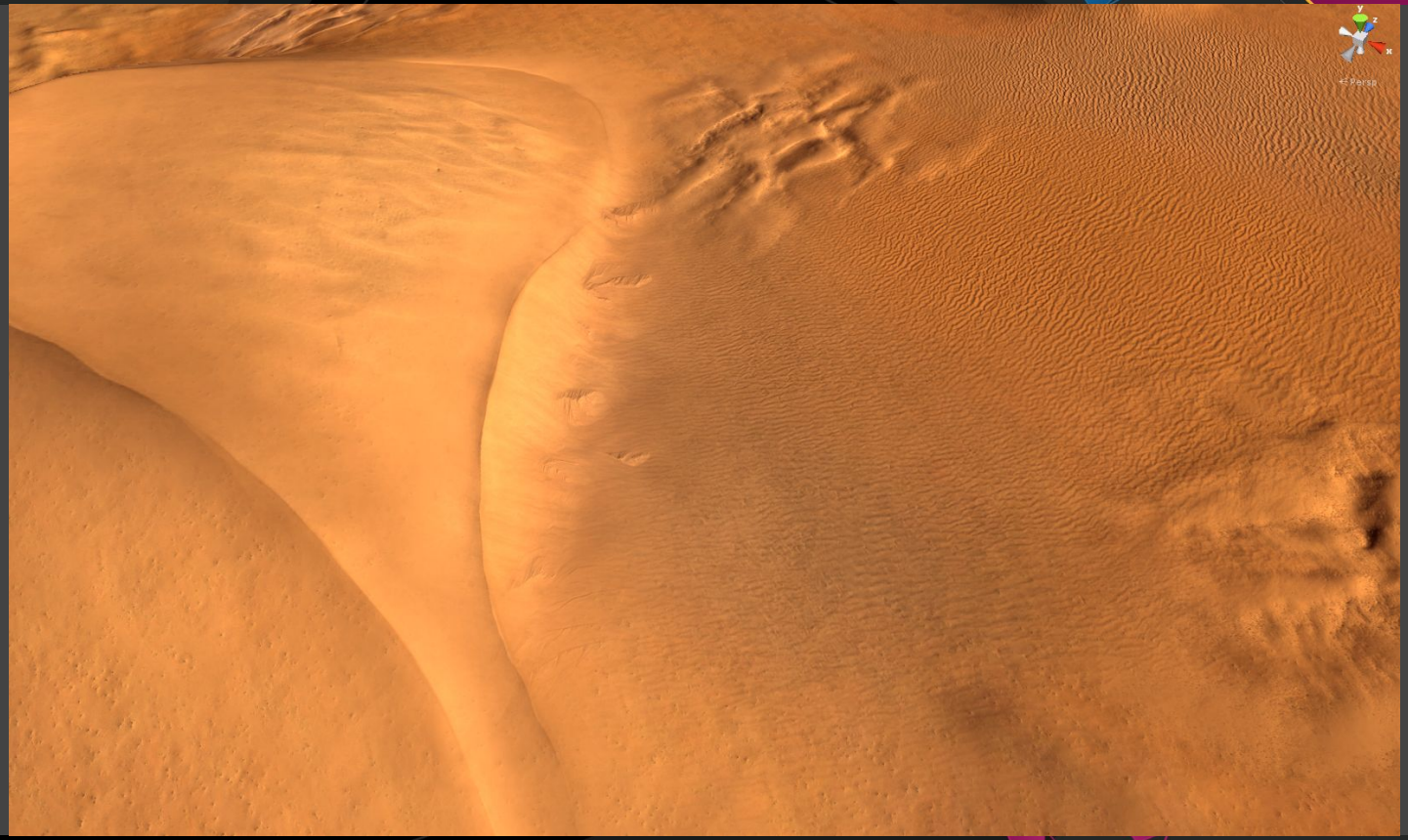


All the decals!





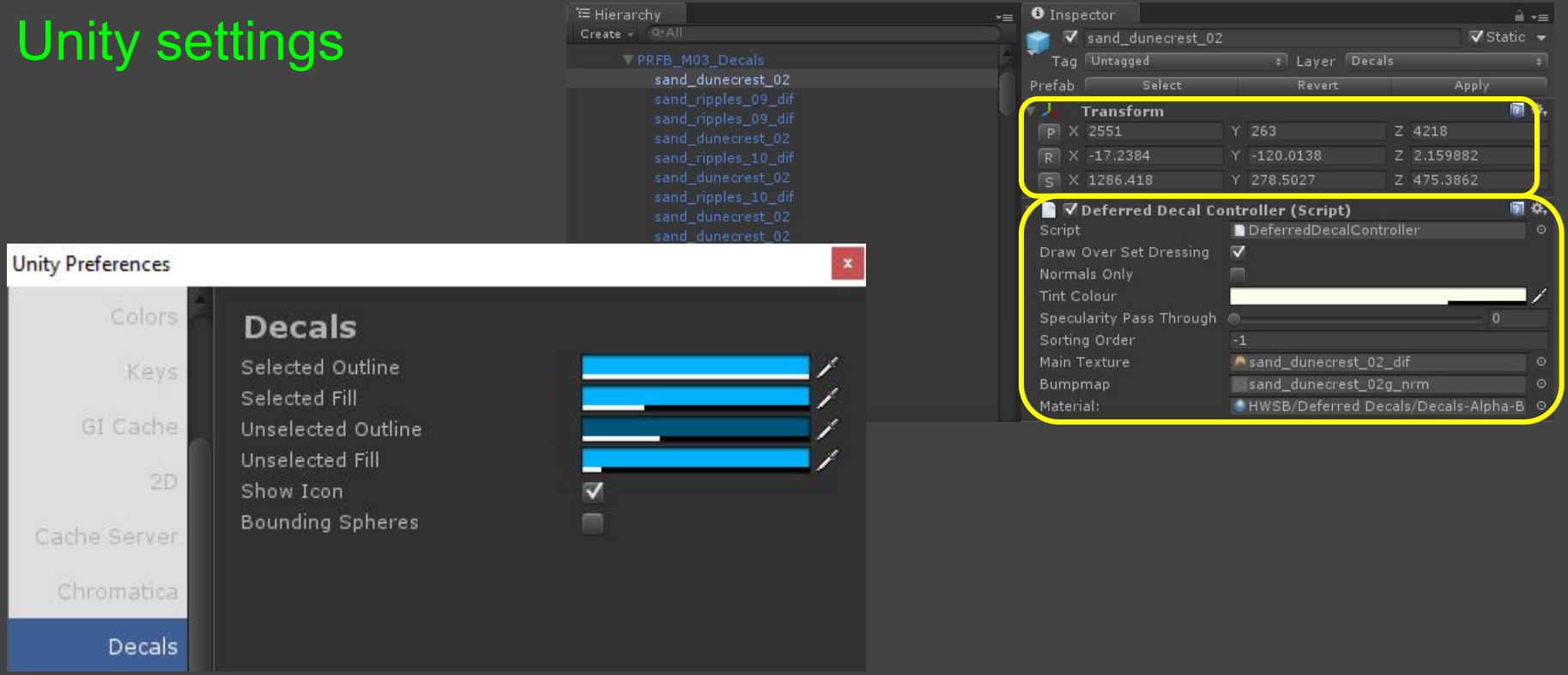

Persp

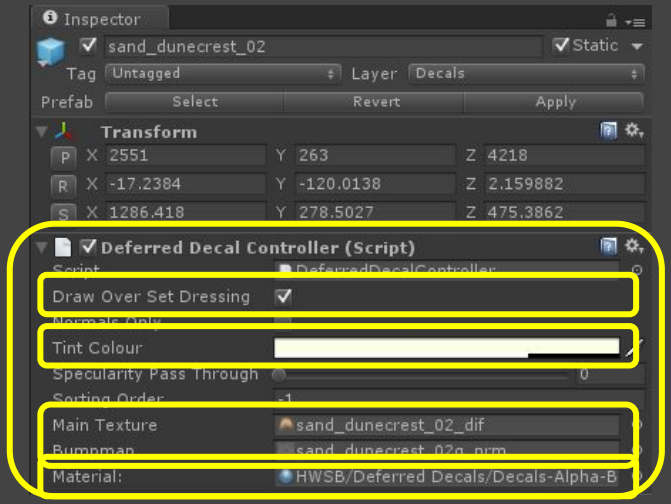



decals


dynamic decals

Unity settings






```
Shader "HWSB/Deferred Decals/Decals-Alpha-BumpedDiffuse" {
  Properties {
    _Color ("Main Color (RGB)", Color) = (1,1,1,1)
    _MainTex ("Diffuse", 2D) = "white" {}
    _BumpMap ("Normals", 2D) = "bump" {}
    _SpecularityPassthrough("Spec Passthrough", RANGE(0,1)) = 0
    _Transparency ("Transparency", RANGE(0,1)) = 1
  }
  SubShader {
    Pass {
      Stencil {
        Ref 2
        Readmask 2
        Writemask 0
        Comp NotEqual
        Pass keep
      }
      Tags { "LightingMode"="Deferred" }
      ZWrite Off
      ZTest GEqual
      Cull Front
      Blend SrcAlpha OneMinusSrcAlpha

      #include "Include/DiffNrm.cginc"
    }
  }
  Fallback Off
}
```

Screen Space Decals in Warhammer 40,000: Space Marine, Pope Kim, Relic Entertainment, SIGGRAPH 2012, Los Angeles

Extending Unity 5 Rendering Pipeline: Command Buffers, Aras Pranckevičius, Unity Technologies, blog post, February 6, 2015

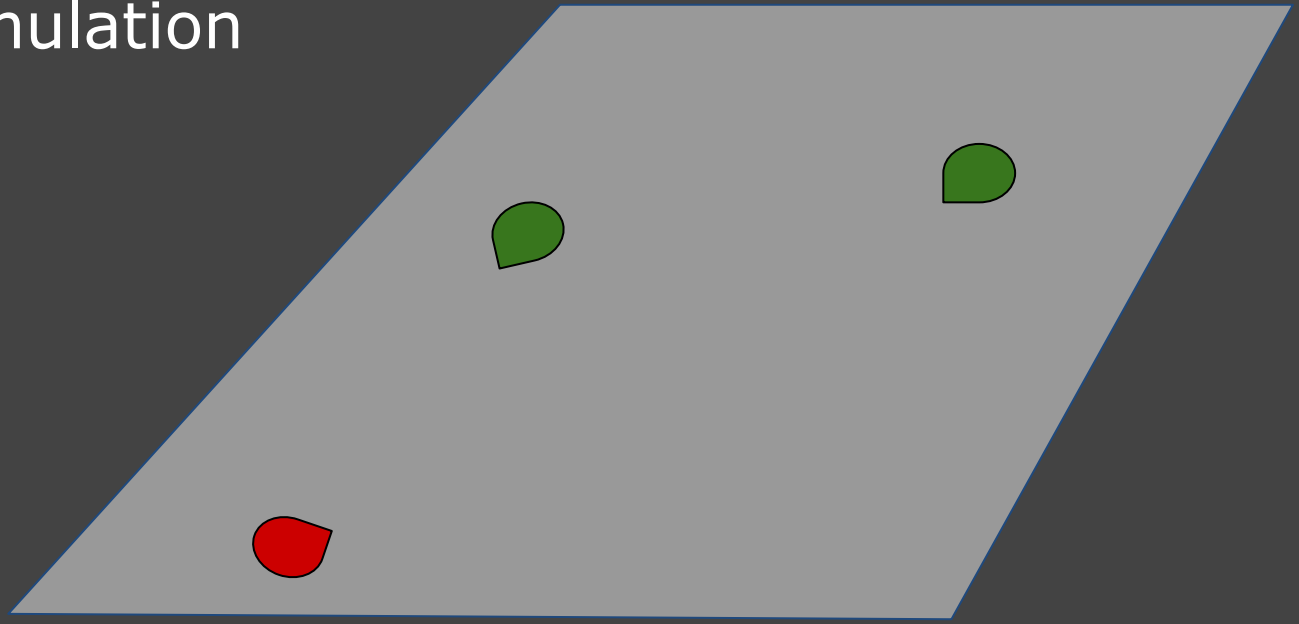
During its examination of Mars, the Viking 1 spacecraft returned images of Valles Marineris, a huge canyon system 5,000 km, or about 3,106 miles, long, whose connected chasma or valleys may have formed from a combination of erosional collapse and structural activity.


A BLACKBIRD INTERACTIVE PRODUCTION
IN COOPERATION WITH THE NASA JET PROPULSION LABORATORY
PROJECT
EAGLE
BBI
BLACKBIRD
INTERACTIVE

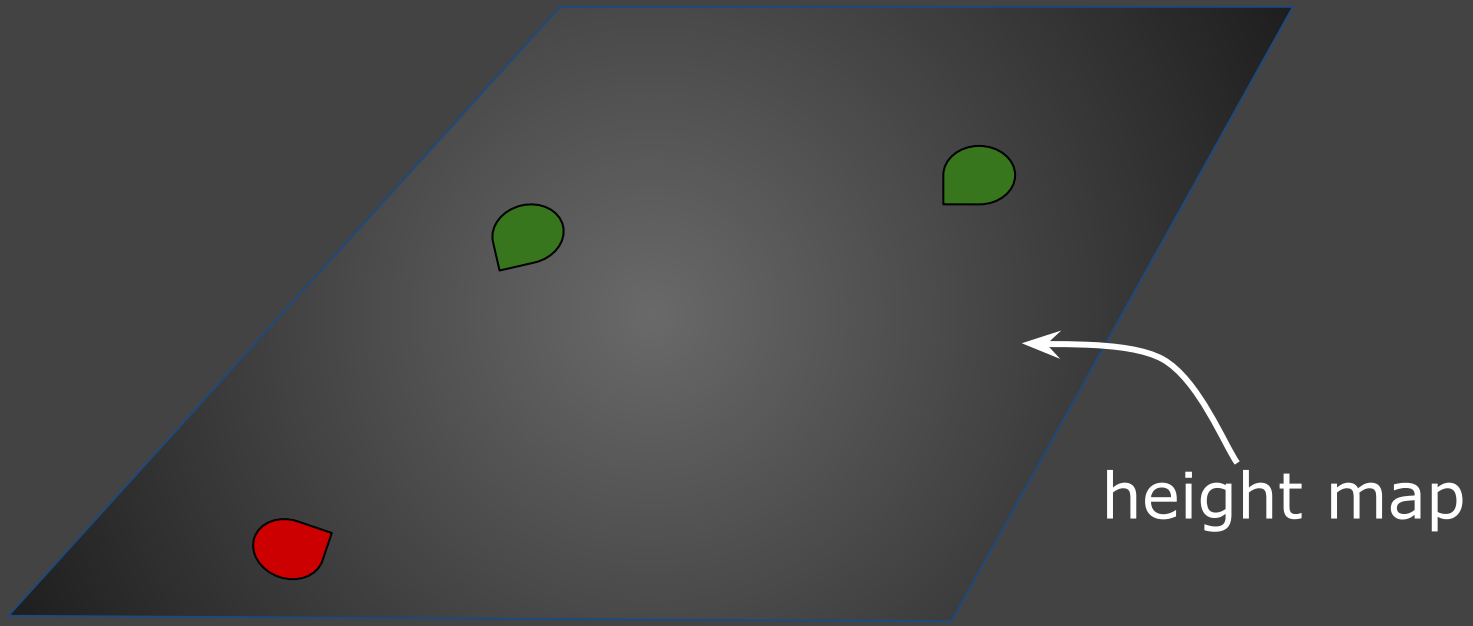

blackbirdinteractive.com/news/project-eagle
BLACKBIRD INTERACTIVE
01
EB-02 COLONY CONSTRUCTION VIS
EAGLE BASE-Exploration Demo
build.v00.1

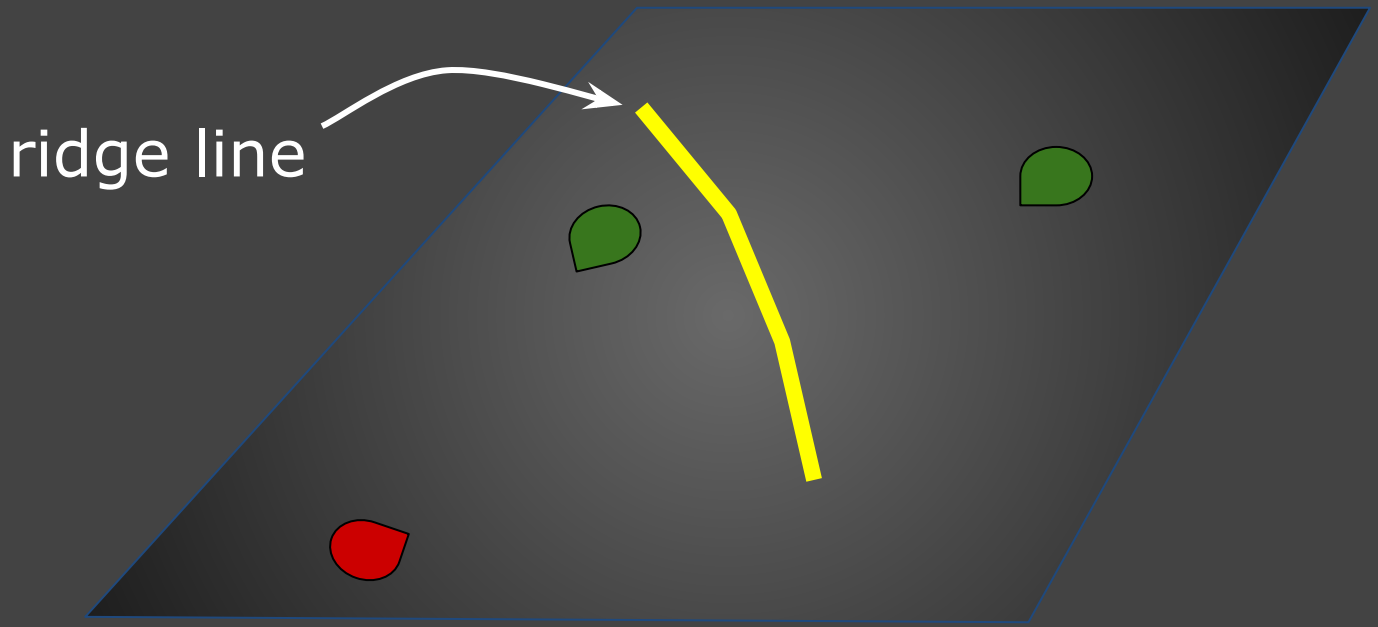
Simulated Terrain?

2D Simulation

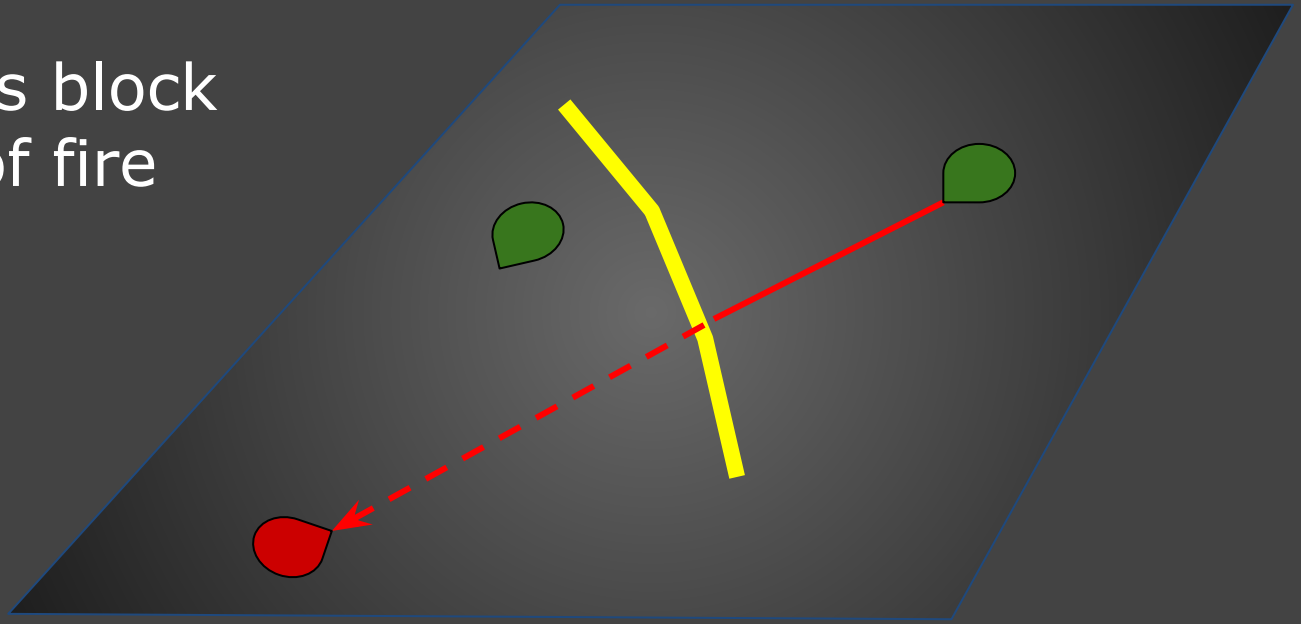


... this is Homeworld... mostly

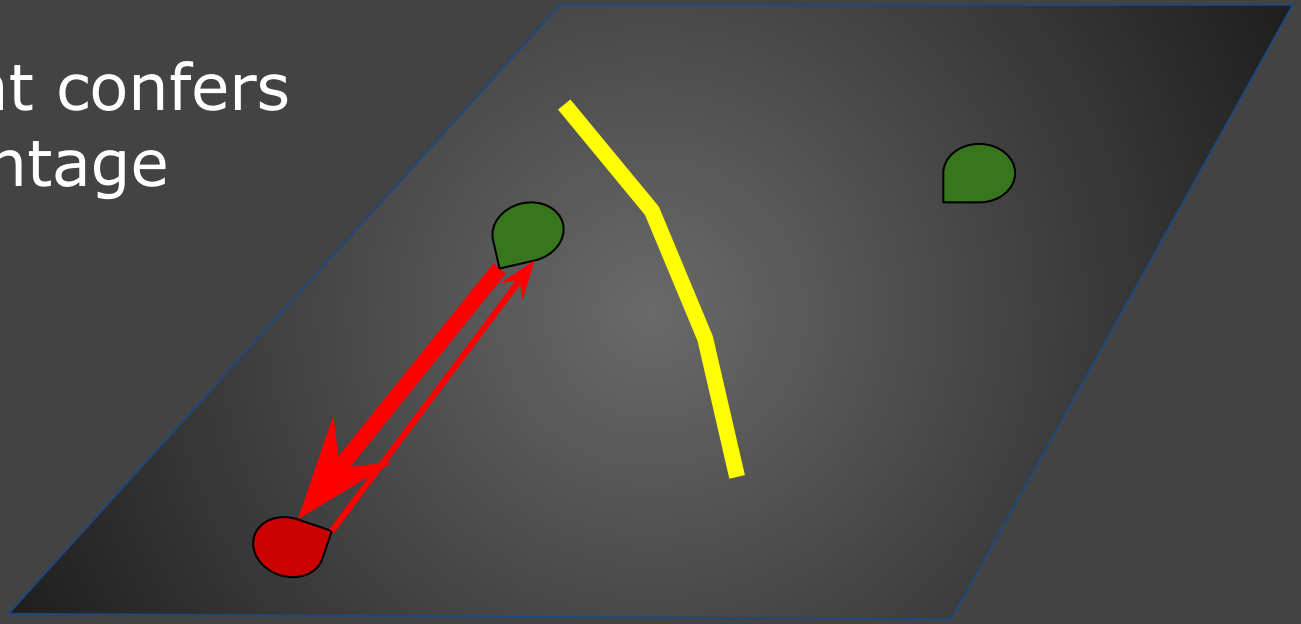

height map


ridge line

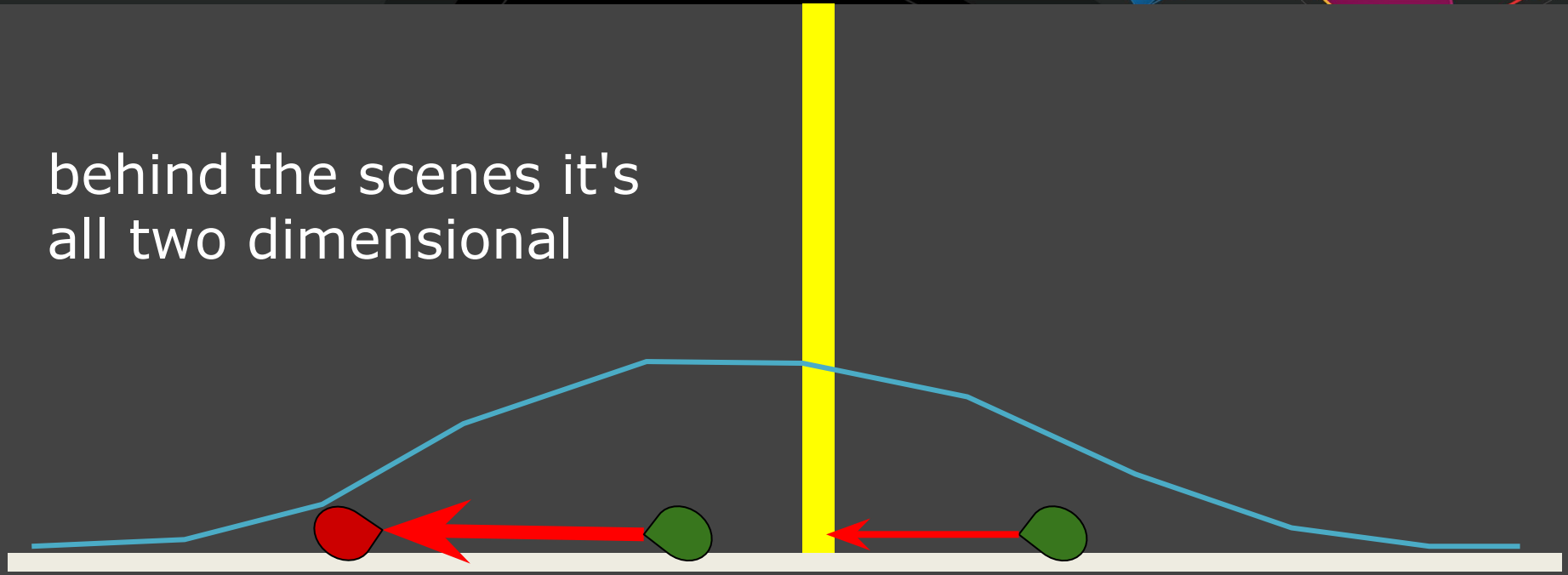
ridges block line of fire



height confers advantage



behind the scenes it's all two dimensional



Unique Visuals - Aesthetic Physics

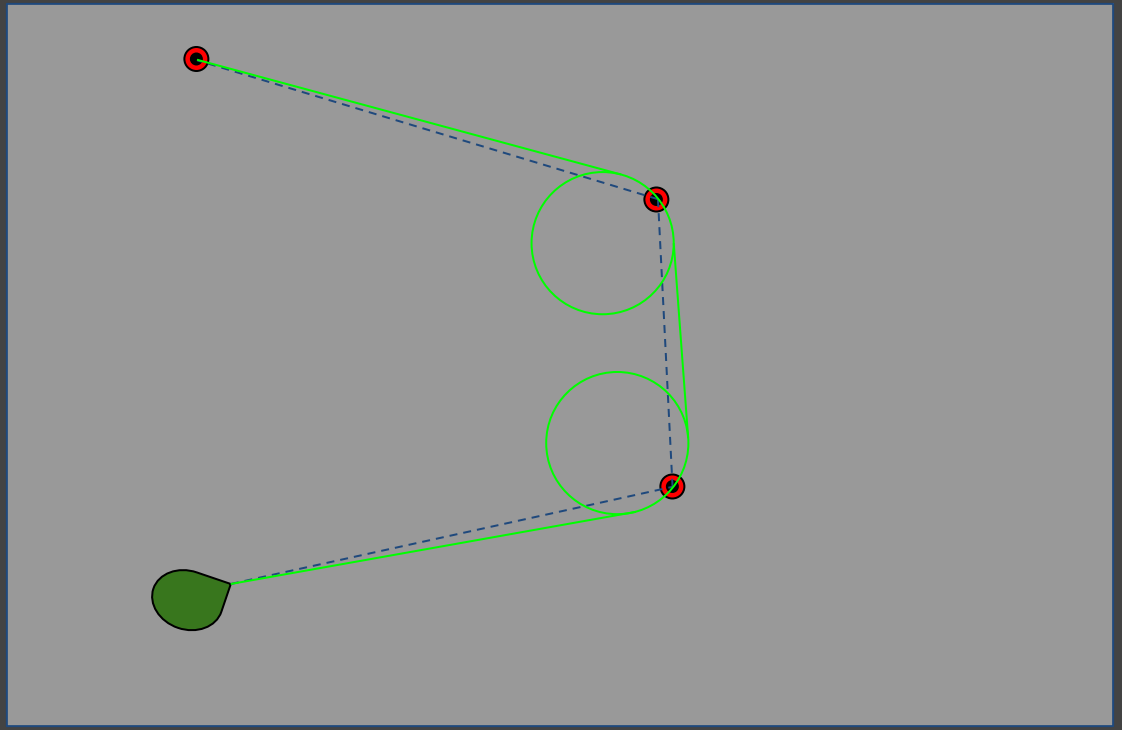
vehicle fantasy

2D sim vs. 3D presentation

- deterministic vehicle model
- suspension, jumps, etc.

implementation, rigging, interpolation

2D Movement

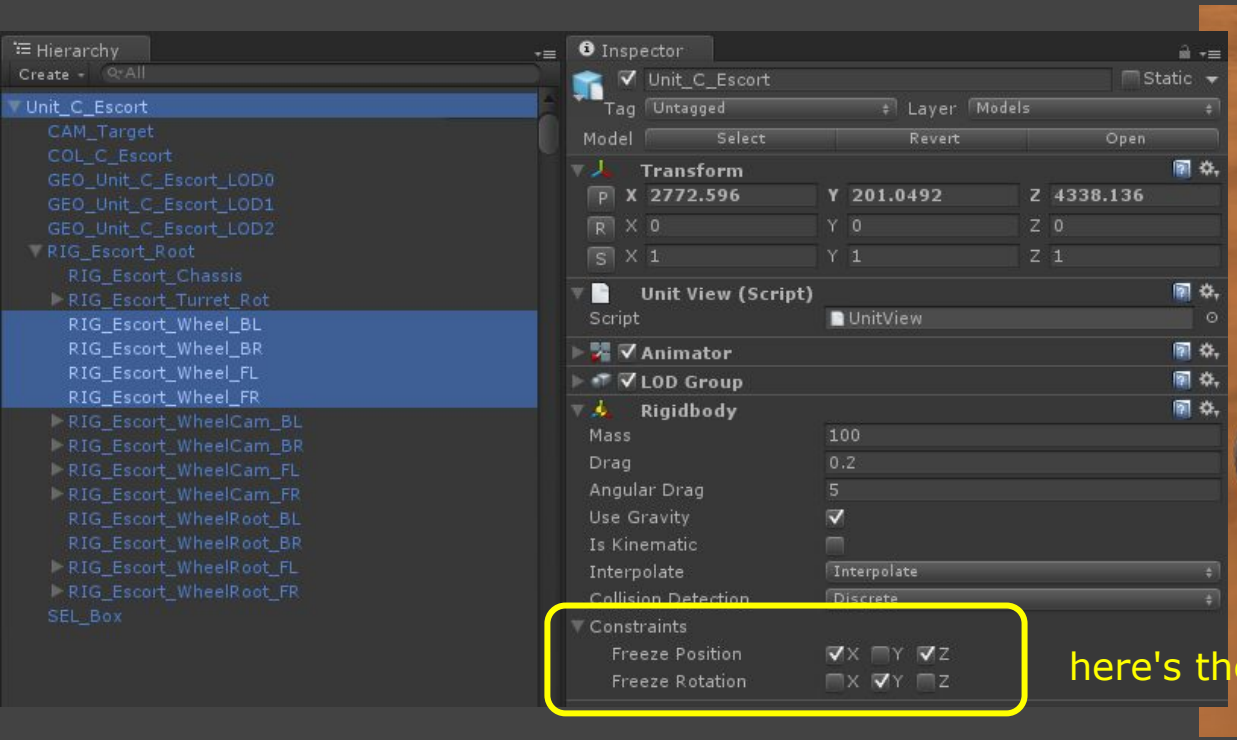


but, but ...


simulation
presentation


simulation
presentation


03
015



here's the pin!

Hierarchy

Create ▾ Q▾All

- Unit_C_Escort
  - CAM_Target
  - COL_C_Escort
  - GEO_Unit_C_Escort_LOD0
  - GEO_Unit_C_Escort_LOD1
  - GEO_Unit_C_Escort_LOD2
  - RIG_Escort_Root
    - RIG_Escort_Chassis
    - RIG_Escort_Turret_Rot
    - RIG_Escort_Wheel_BL
    - RIG_Escort_Wheel_BR
    - RIG_Escort_Wheel_FL
    - RIG_Escort_Wheel_FR
    - RIG_Escort_WheelCam_BL
    - RIG_Escort_WheelCam_BR
    - RIG_Escort_WheelCam_FL
    - RIG_Escort_WheelCam_FR
    - RIG_Escort_WheelRoot_BL
    - RIG_Escort_WheelRoot_BR
    - RIG_Escort_WheelRoot_FL
    - RIG_Escort_WheelRoot_FR
  - SEL_Box

no geometry?

magic!

Inspector

RIG_Escort_Wheel_BL — Static

Tag: Untagged — Layer: Models

Model: Select | Revert | Open

Transform

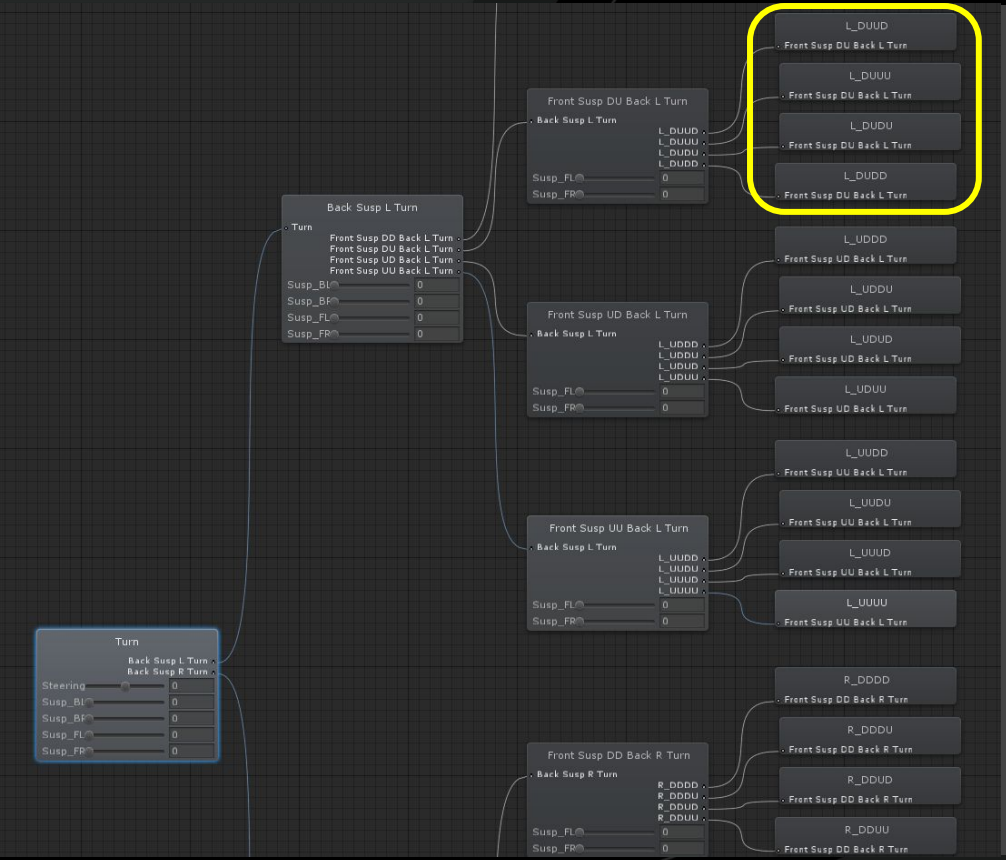
P	X -3.377036	Y 1.465921	Z -4.148181
R	X 0	Y 0	Z 0
S	X 1	Y 1	Z 1

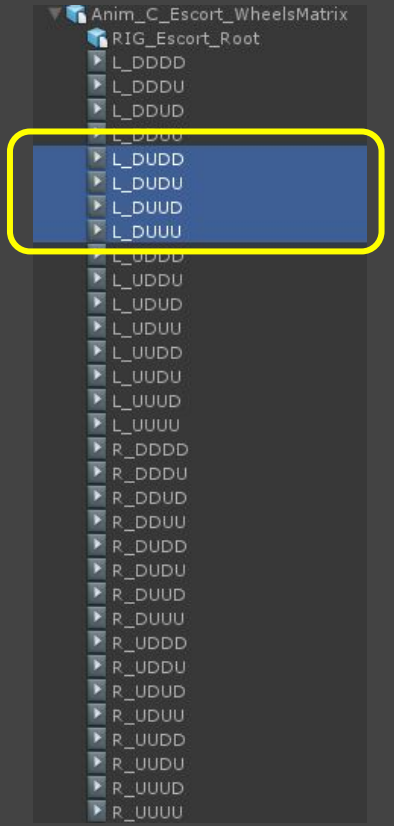
Wheel Collider

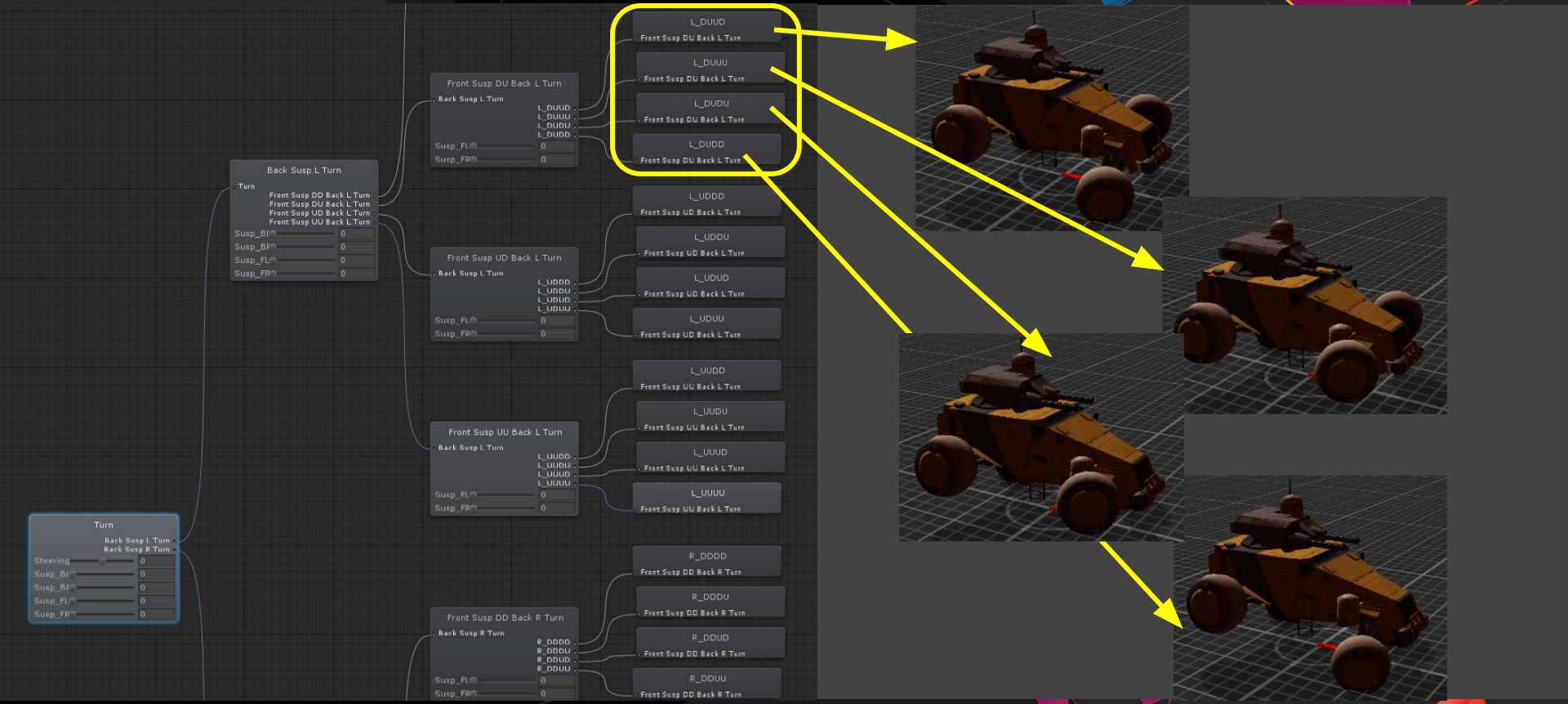
Center	X 0	Y 1.318	Z 0
Radius	1.6		
Suspension Spring			
Spring	2000		
Damper	200		
Target Position	0.3		
Suspension Distance	1.649		
Force App Point Distance	1.1		
Mass	5		
Wheel Damping Rate	0.75		
Forward Friction			
Sideways Friction			

Animated Suspension (Script)

Script	AnimatedSuspension
Height Smooth	0.05
Tilt Smooth	0
Wheel Rot Bone Name	RIG_Escort_WheelRot_BL
Height Param	Susp_BL_Height
Tilt Param	


Turn
Back Susp L Turn
Back Susp R Turn
Steering
Susp_BL
Susp_BR
Susp_FL
Susp_FR
Back Susp L Turn
Front Susp DD Back L Turn
Front Susp DU Back L Turn
Front Susp UD Back L Turn
Front Susp UU Back L Turn
Front Susp DU Back L Turn
Front Susp UD Back L Turn
Front Susp UU Back L Turn
Front Susp DD Back R Turn
L_DUUD
L_DUUU
L_DUDU
L_DUDD
L_UDDD
L_UDDU
L_UDUD
L_UDUU
L_UUDD
L_UUDU
L_UUUD
L_UUUU
R_DDDD
R_DDDU
R_DDUD
R_DDUU


Anim_C_Escort_WheelsMatrix
RIG_Escort_Root
L_DDDD
L_DDDU
L_DDUD
L_DDUU
L_DUDD
L_DUDU
L_DUUD
L_DUUU
L_UDDD
L_UDDU
L_UDUD
L_UDUU
L_UUDD
L_UUDU
L_UUUD
L_UUUU
R_DDDD
R_DDDU
R_DDUD
R_DDUU
R_DUDD
R_DUDU
R_DUUD
R_DUUU
R_UDDD
R_UDDU
R_UDUD
R_UDUU
R_UUDD
R_UUDU
R_UUUD
R_UUUU


Turn
Back Susp L Turn
Back Susp R Turn
Steering
Susp_BL
Susp_BR
Susp_FL
Susp_FR
Back Susp L Turn
Turn
Front Susp DD Back L Turn
Front Susp DU Back L Turn
Front Susp UD Back L Turn
Front Susp UU Back L Turn
Front Susp DU Back L Turn
Back Susp L Turn
L_DUUD
L_DUUU
L_DUDU
L_DUDD
Front Susp UD Back L Turn
Back Susp L Turn
L_UDDD
L_UDDU
L_UDUD
L_UDUU
Front Susp UU Back L Turn
Back Susp L Turn
L_UUDD
L_UUDU
L_UUUD
L_UUUU
Front Susp DD Back R Turn
Back Susp R Turn
R_DDDD
R_DDDU
R_DDUD
R_DDUU
L_DUUD
Front Susp DU Back L Turn
L_DUUU
Front Susp DU Back L Turn
L_DUDU
Front Susp DU Back L Turn
L_DUDD
Front Susp DU Back L Turn
L_UDDD
Front Susp UD Back L Turn
L_UDDU
Front Susp UD Back L Turn
L_UDUD
Front Susp UD Back L Turn
L_UDUU
Front Susp UD Back L Turn
L_UUDD
Front Susp UU Back L Turn
L_UUDU
Front Susp UU Back L Turn
L_UUUD
Front Susp UU Back L Turn
L_UUUU
Front Susp UU Back L Turn
R_DDDD
Front Susp DD Back R Turn
R_DDDU
Front Susp DD Back R Turn
R_DDUD
Front Susp DD Back R Turn
R_DDUU
Front Susp DD Back R Turn

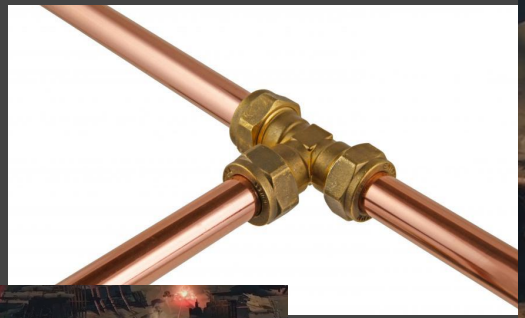
and also ...

not to mention ...

Sastrei, "Retreat! Re-DERP! Re-HERP!", imgur

Deterministic Gameplay - Why?

because multiplayer



BANDWIDTH IS ...

position, speed, health, power ...

turret orientations, ammo, projectiles …

active abilities, cooldowns ...

goals, targets, ...

buffs* .....

… x 800 units

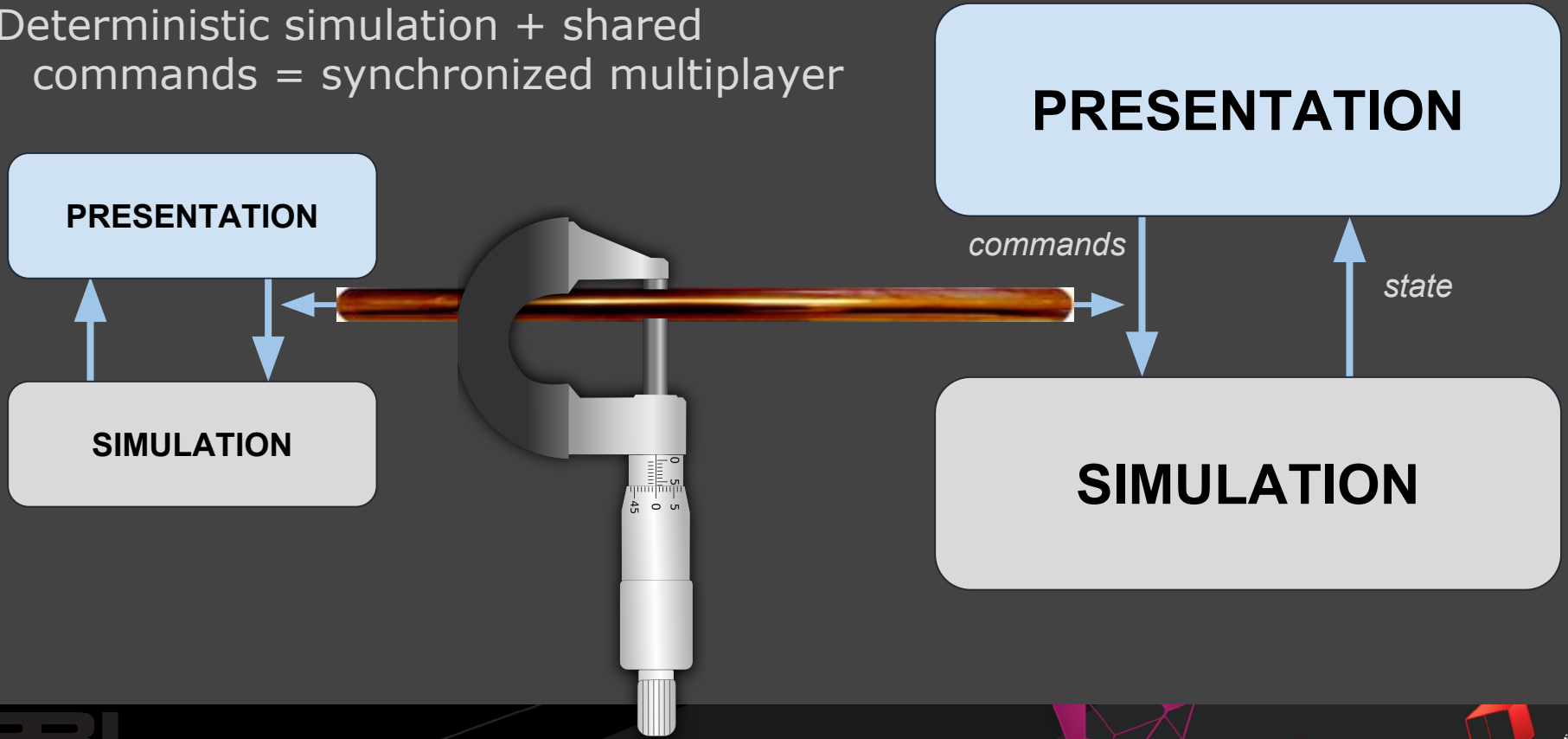
*attribute modifiers

BANDWIDTH IS ...

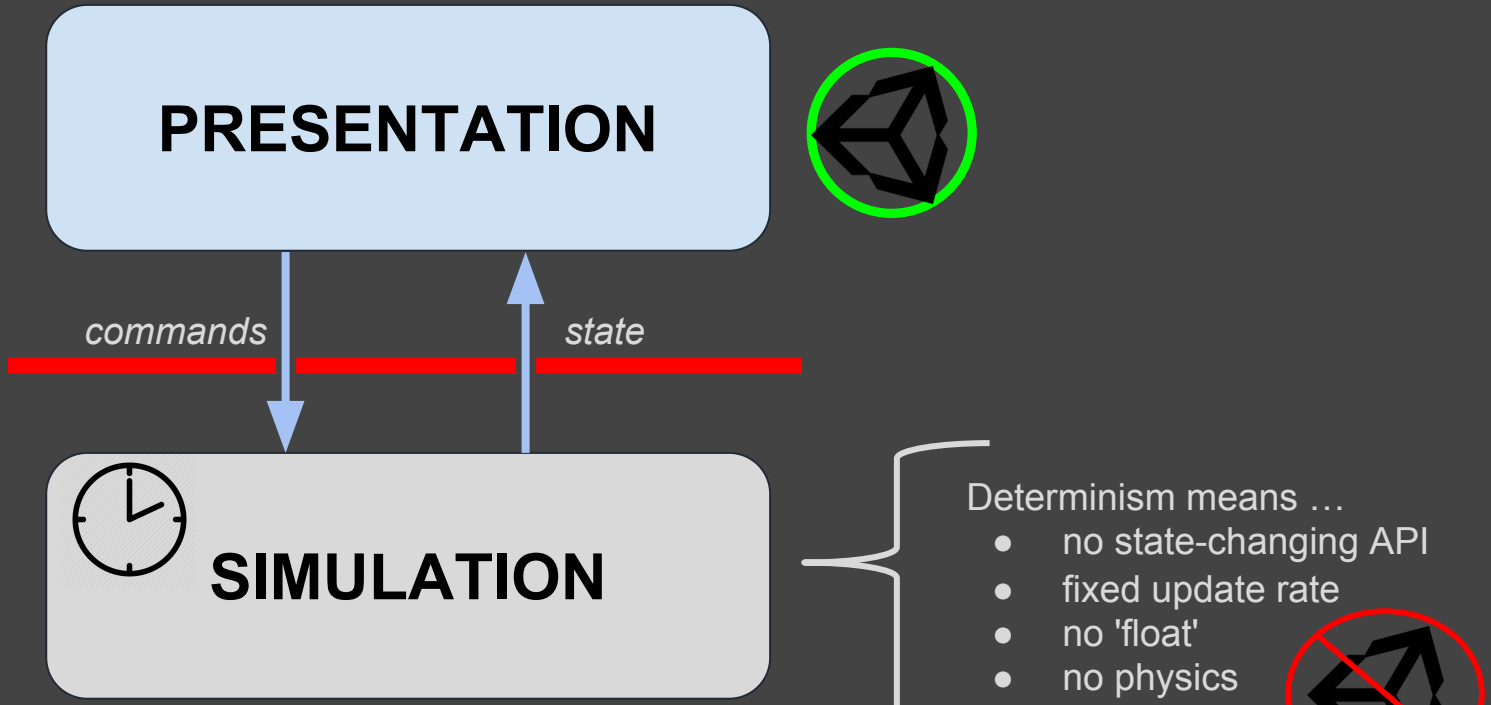
Determinism = repeatability

Given the same inputs, program should pass through the same sequence of states and produce the same outputs.

Deterministic simulation + shared commands = synchronized multiplayer




Sounds reasonable
What's the catch?

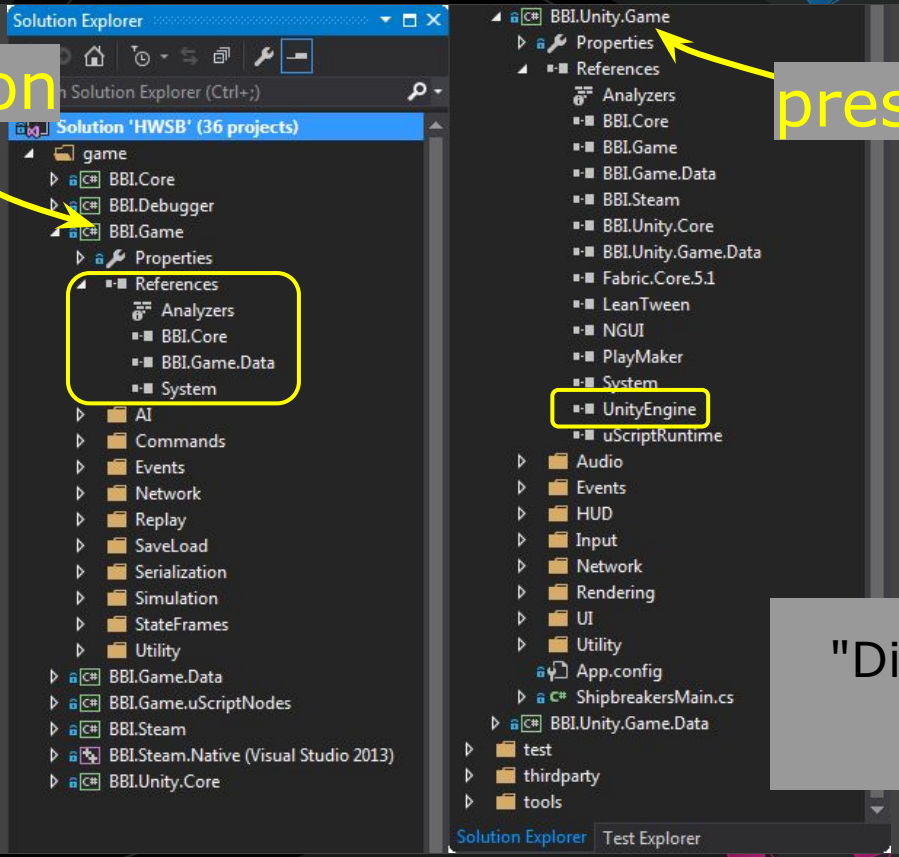

PRESENTATION
commands
state
SIMULATION
Determinism means …
no state-changing API
fixed update rate
no 'float'
no physics
no Unity

Deterministic Gameplay - Architecture

simulation

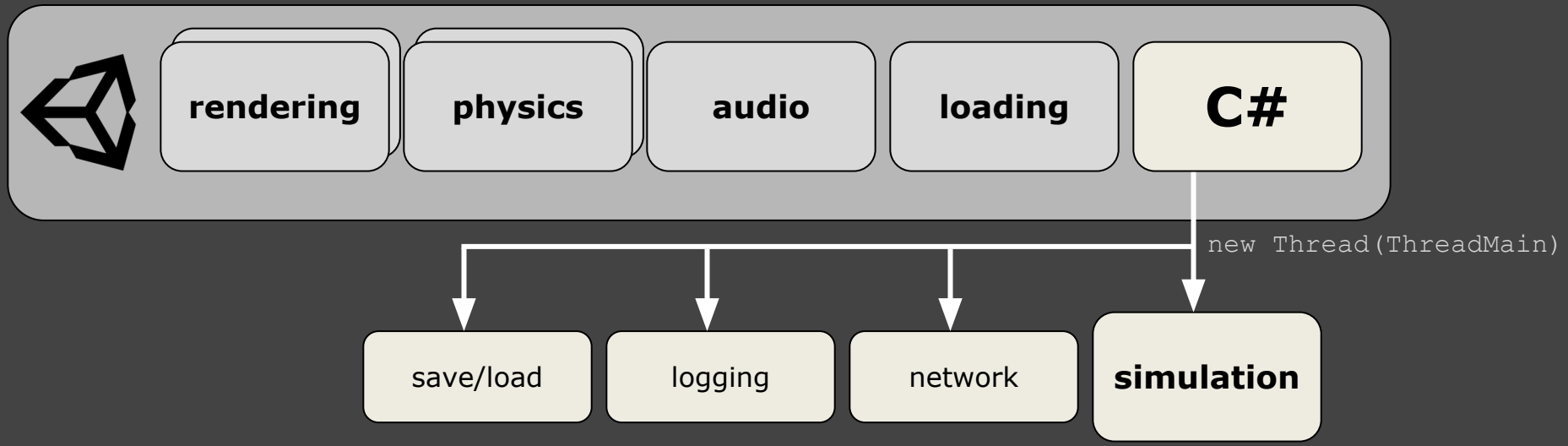
presentation

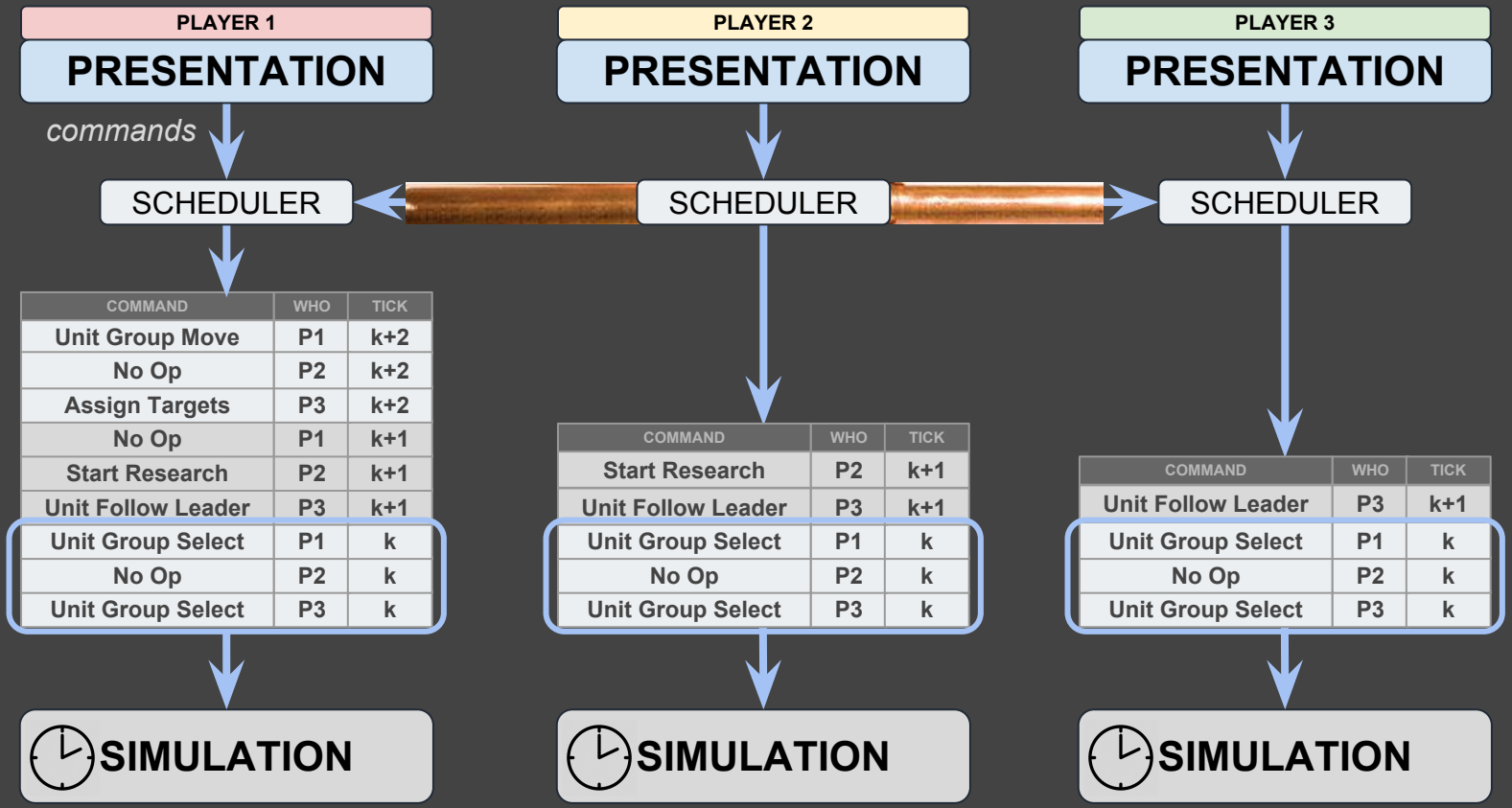
separate assemblies
clear dependencies
internal scope

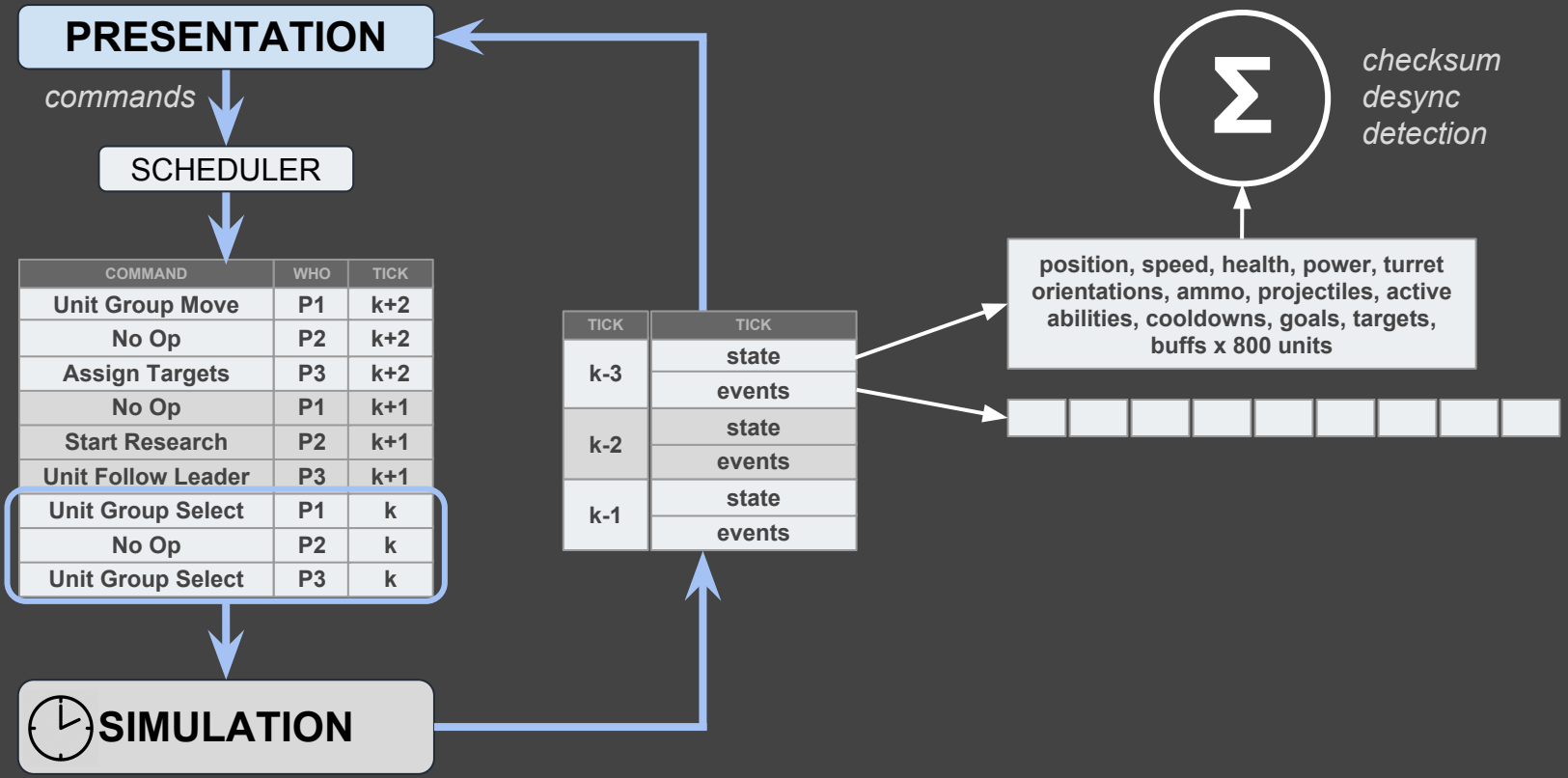


"Dig a pit of success."
Tim Ford

multi-threading




PLAYER 1
PRESENTATION
commands
SCHEDULER
COMMAND | WHO | TICK
Unit Group Move | P1 | k+2
No Op | P2 | k+2
Assign Targets | P3 | k+2
No Op | P1 | k+1
Start Research | P2 | k+1
Unit Follow Leader | P3 | k+1
Unit Group Select | P1 | k
No Op | P2 | k
Unit Group Select | P3 | k
SIMULATION
PLAYER 2
PRESENTATION
SCHEDULER
COMMAND | WHO | TICK
Start Research | P2 | k+1
Unit Follow Leader | P3 | k+1
Unit Group Select | P1 | k
No Op | P2 | k
Unit Group Select | P3 | k
SIMULATION
PLAYER 3
PRESENTATION
SCHEDULER
COMMAND | WHO | TICK
Unit Follow Leader | P3 | k+1
Unit Group Select | P1 | k
No Op | P2 | k
Unit Group Select | P3 | k
SIMULATION


PRESENTATION
commands
SCHEDULER
COMMAND | WHO | TICK
Unit Group Move | P1 | k+2
No Op | P2 | k+2
Assign Targets | P3 | k+2
No Op | P1 | k+1
Start Research | P2 | k+1
Unit Follow Leader | P3 | k+1
Unit Group Select | P1 | k
No Op | P2 | k
Unit Group Select | P3 | k
SIMULATION
TICK | TICK
k-3 | state
events
k-2 | state
events
k-1 | state
events
Σ
checksum
desync
detection
position, speed, health, power, turret orientations, ammo, projectiles, active abilities, cooldowns, goals, targets, buffs x 800 units

1500 Archers on a 28.8: Network Programming in Age of Empires and Beyond, Mark Terrano, Paul Bettner, Ensemble Studios, Gamasutra, March 22, 2001

Deterministic Gameplay - 

Deemed necessary

Fixed point math library

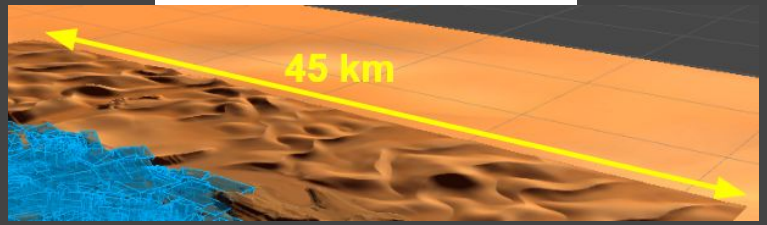
No Unity dependencies

No third party float

```
Fixed64 x = 42;
Fixed64 y = 99;
Fixed64 a = x + y;
```

```
[StructLayout(LayoutKind.Explicit)]
public struct Fixed64
{
    [FieldOffset(0)]private long mRawValue;
    [FieldOffset(0)]private uint mRawLower;
    [FieldOffset(4)]private int mRawUpper;

    public static Fixed64 operator +(Fixed64 x, Fixed64 y)
    {
        long xl = x.mRawValue;
        long yl = y.mRawValue;
        long sum = xl + yl;

        // Overflow?
        if (((~(xl ^ yl) & (xl ^ sum)) & kMinRawValue) != 0)
        {
            return (xl > 0 ? PositiveInfinity : NegativeInfinity);
        }
        return new Fixed64(sum);
    }

    public static implicit operator Fixed64(int value)
    {
        return new Fixed64((long)value << 32);
    }

    ... // lots more!
}
```

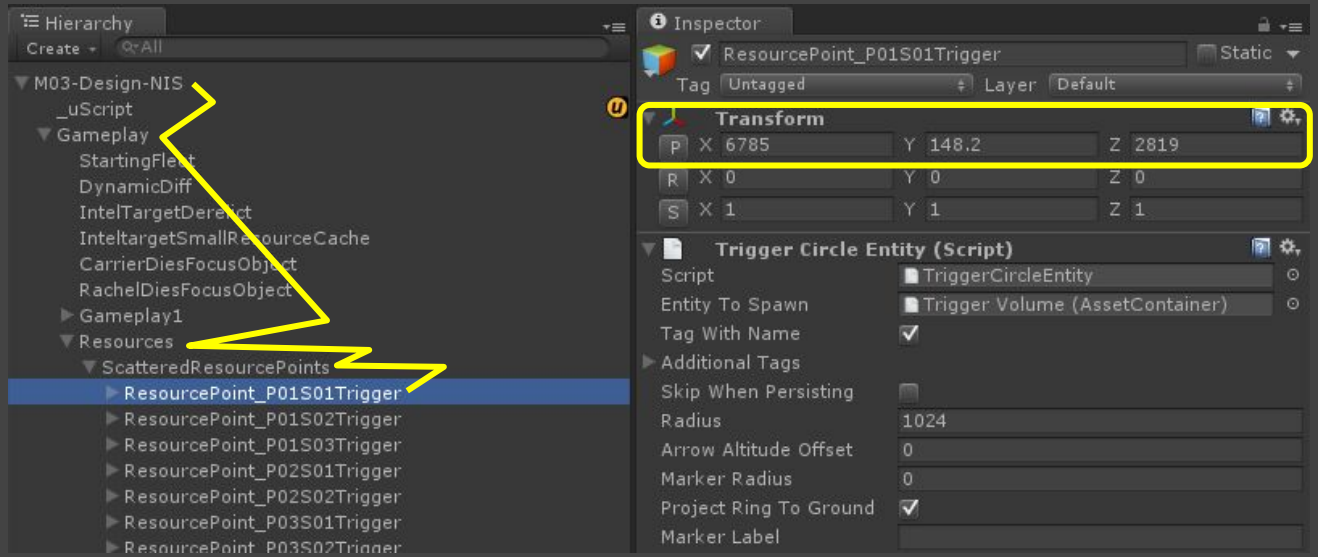
$$\sqrt{2^{31}} = 46,340.95$$



did it work?


SYSTEM REQUIREMENTS
Windows
Mac OS X
MINIMUM:
OS: Windows 7/8/10 32
RECOMMENDED:
OS: Windows 7/8/10 (64-bit)

was it necessary?



Performance - Level of Detail


OPTIONS
VIDEO
SOUND
CONTROLS
GAMEPLAY
GLOBAL SETTINGS
ULTRA
RENDERING
RESOLUTION
3840x2160
FULL SCREEN
V-SYNC
SHADOW QUALITY
HIGH
FOG
BLOOM
ANTI ALIASING
TEXTURE QUALITY
ULTRA
DECAL DRAW DISTANCE
HIGH
ANISOTROPIC FILTERING
x16
FX
DYNAMIC DECALS
HIGH
UNITS
NLIPS
FX QUALITY
ULTRA
FX MAXIMUM LOD
PHYSICS AND ANIMATION QUALITY
ULTRA
HIGH
DEBRIS PERSISTENCE
HIGH
TRACKS LIFETIME
HIGH
DEFAULT SETTINGS
CANCEL
ACCEPT

Performance* - C#

*by which I mean Memory

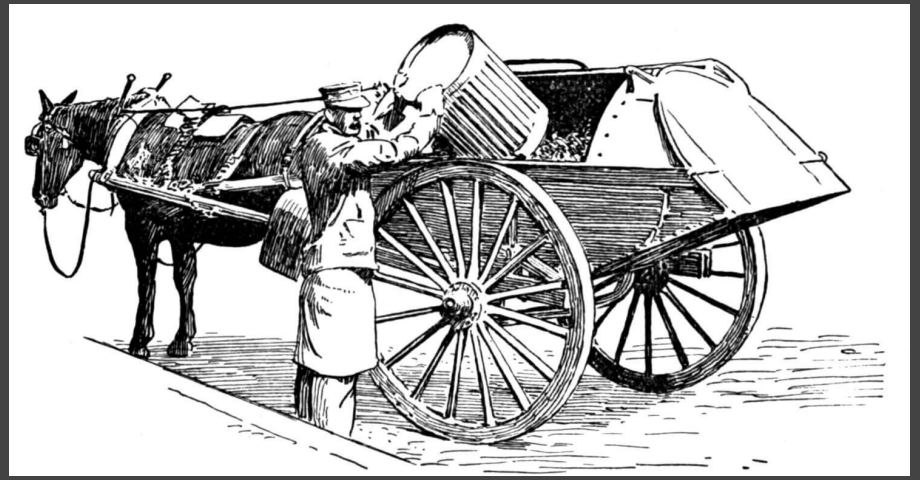
C# uses managed memory

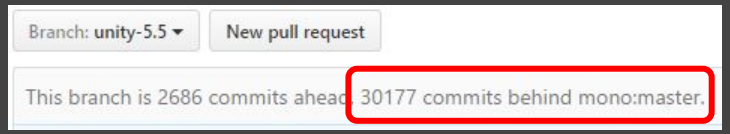
`new` but no `delete`

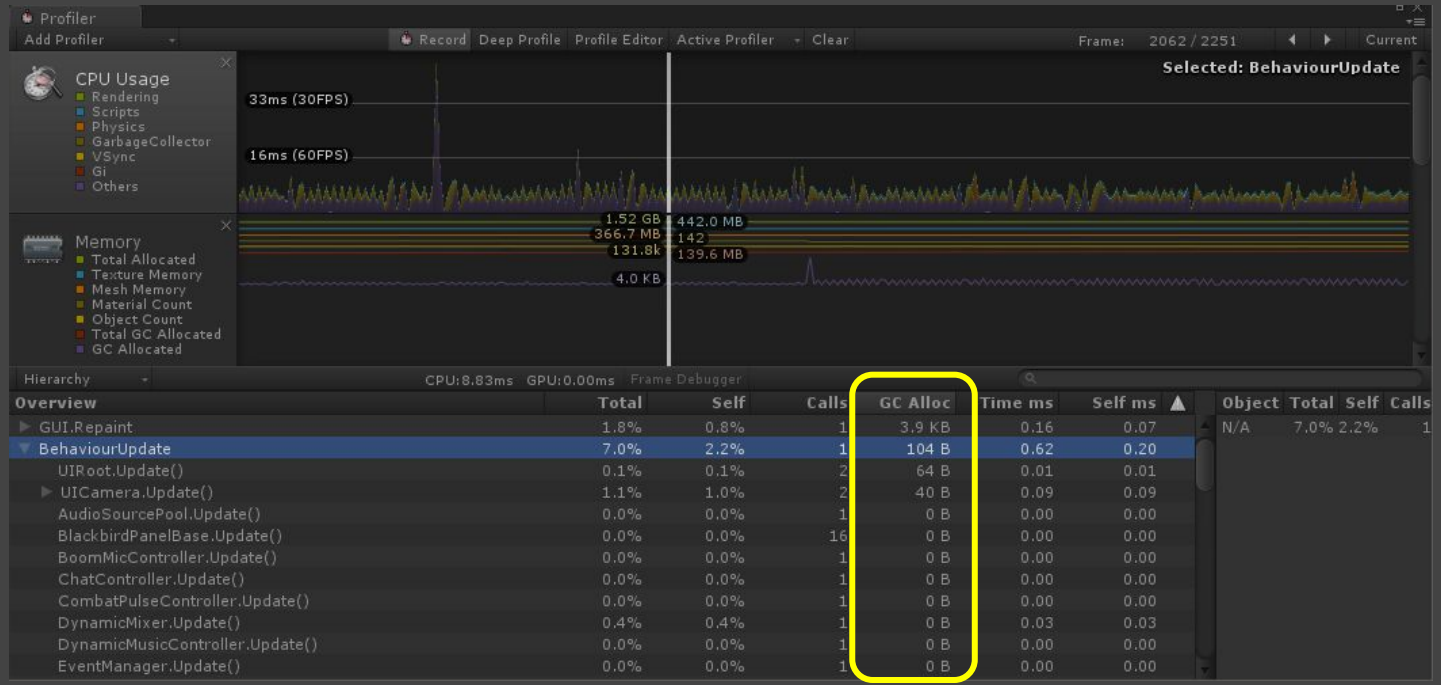
"Hey, don't worry about it, it'll be fine!"

.NET garbage collection

Garbage collection in Unity's Mono






Profiler
Add Profiler
Record
Deep Profile
Profile Editor
Active Profiler
Clear
Frame: 2062 / 2251
Current
CPU Usage
Rendering
Scripts
Physics
GarbageCollector
VSync
Gi
Others
Selected: BehaviourUpdate
33ms (30FPS)
16ms (60FPS)
Memory
Total Allocated
Texture Memory
Mesh Memory
Material Count
Object Count
Total GC Allocated
GC Allocated
1.52 GB
366.7 MB
131.8k
4.0 KB
442.0 MB
142
139.6 MB
Hierarchy
CPU:8.83ms GPU:0.00ms
Frame Debugger
Overview
Total
Self
Calls
GC Alloc
Time ms
Self ms
GUI.Repaint 1.8% 0.8% 1 3.9 KB 0.16 0.07
BehaviourUpdate 7.0% 2.2% 1 104 B 0.62 0.20
UIRoot.Update() 0.1% 0.1% 2 64 B 0.01 0.01
UICamera.Update() 1.1% 1.0% 2 40 B 0.09 0.09
AudioSourcePool.Update() 0.0% 0.0% 1 0 B 0.00 0.00
BlackbirdPanelBase.Update() 0.0% 0.0% 16 0 B 0.00 0.00
BoomMicController.Update() 0.0% 0.0% 1 0 B 0.00 0.00
ChatController.Update() 0.0% 0.0% 1 0 B 0.00 0.00
CombatPulseController.Update() 0.0% 0.0% 1 0 B 0.00 0.00
DynamicMixer.Update() 0.4% 0.4% 1 0 B 0.03 0.03
DynamicMusicController.Update() 0.0% 0.0% 1 0 B 0.00 0.00
EventManager.Update() 0.0% 0.0% 1 0 B 0.00 0.00
Object Total Self Calls
N/A 7.0% 2.2% 1

Collection capacity

```
List<Thing> things = new List<Thing>();
things.Add(thing1);
things.Add(thing2);
…
things.Add(thing17);
```

Allocate doubled array and copy!

Collection capacity

```
List<Thing> things = new List<Thing>(17);
things.Add(thing1);
things.Add(thing2);
…
things.Add(thing17);
```

No allocation.

Consider writing your own collection classes.
e.g. Fixed capacity list wrapping an array.

Temporary lists

```
public void DoThings()
{
    List<Thing> things = new List<Thing>(kMaxThings);
    things.Add(...);
    DoAllTheThings(things);
}
```

Temporary lists

```
private static List<Thing> sThings = new List<Thing>(kMaxCapacity);
```

one-time allocation

```
public void DoThings()
{
    sThings.Add(...);
    DoAllTheThings(sThings);
    sThings.Clear*();
}
```

*Not sure if Clear resets capacity? Check the reference source

Object pools

- Especially for Unity prefab instances, due to instantiation costs.
- Great for other objects too.
- Be sure to "clean up" objects when putting them back in the pool.

```
public class ObjectPool<T> where T : new()
{
    private List<T> mReservedObjects = null;
    private List<T> mAvailableObjects = null;

    public ObjectPool(int capacity)
    {
        mReservedObjects = new List<T>(capacity);
        mAvailableObjects = new List<T>(capacity);

        for (int i = 0; i < capacity; ++i)
        {
            mAvailableObjects.Add(new T());
        }
    }

    public T Reserve()
    {
        ...
    }

    public void Release(T obj)
    {
        ...
    }
}
```

strings, boxing, params

```
int x = 42;
string s = string.Format("x={0}", x);
```

how many allocations?

1. box x
2. temporary array of parameters
3. x.ToString()
4. new string s

C# | C++ | F# | VB

```
public static string Format(
        string format,
        params object[] args
)
```

strings, boxing, params

```
int x = 42;
string s = string.Concat("x=", x.ToString());
```

C# | C++ | F# | VB

```
public static string Concat(
        string str0,
        string str1
)
```

1. ~~box x~~
2. ~~temporary array of parameters~~
3. x.ToString()
4. new string s

Avoid boxing

```
struct StructThing
{
    int Size;
    string Name;
}
```

```
void DoTheThing(object thing)
{
    ...
}
```

boxing!

```
StructThing thing = new StructThing();
DoTheThing(thing);
```

Avoid boxing

```
struct StructThing : IThing
{
    int Size { get; private set; }
    string Name { get; private set; }
}
```

```
interface IThing
{
    int Size { get; }
    string Name { get; }
}
```

boxing!

```
void DoTheThing(IThing thing)
{
    ...
}
```

```
StructThing thing = new StructThing();
DoTheThing(thing);
```

Avoid boxing

```
struct StructThing : IThing
{
    int Size { get; private set; }
    string Name { get; private set; }
}

interface IThing
{
    int Size { get; }
    string Name { get; }
}

StructThing thing = new StructThing();
DoTheThing(thing);
```

no boxing!

```
void DoTheThing<T>(T thing)
    where T:IThing
{
    ...
}
```

Avoid boxing enumerators

```
MyCollection<Thing> things;
...
foreach (Thing t in things)
{
    ...
}
```

how many allocations?

```
class MyCollection<T> : IEnumerable<T>
{
    ...
}
```

Avoid boxing enumerators

```
MyCollection<Thing> things;
...
foreach (Thing t in things)
{
    ...
}
```

how many allocations?

```
class MyCollection<T> : IEnumerable<T>
{
  IEnumerator<T> GetEnumerator()
  {
      return new MyEnumerator();
  }

  private class MyEnumerator:IEnumerator<T>
  {
    ...
  }
}
```

Avoid boxing enumerators

```
MyCollection<Thing> things;
...
foreach (Thing t in things)
{
    ...
}
```

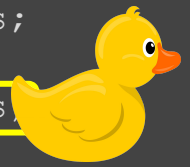
how many allocations?

```
class MyCollection<T> : IEnumerable<T>
{
  IEnumerator<T> GetEnumerator()
  {
      return new MyEnumerator();
  }

  private struct MyEnumerator:IEnumerator<T>
  {
     ...
  }
}
```

boxing!

Avoid boxing enumerators

```
MyCollection<Thing> things;
...
foreach (Thing t in things)
{
    ...
}
```



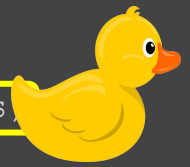
how many allocations?

```
class MyCollection<T> : IEnumerable<T>
{
  MyEnumerator GetEnumerator()
  {
      return new MyEnumerator();
  }

  public struct MyEnumerator:IEnumerator<T>
  {
     ...
  }
}
```

Avoid boxing enumerators

```
IEnumerable<Thing> things;
...
foreach (Thing t in things)
{
    ...
}
```

how many allocations?



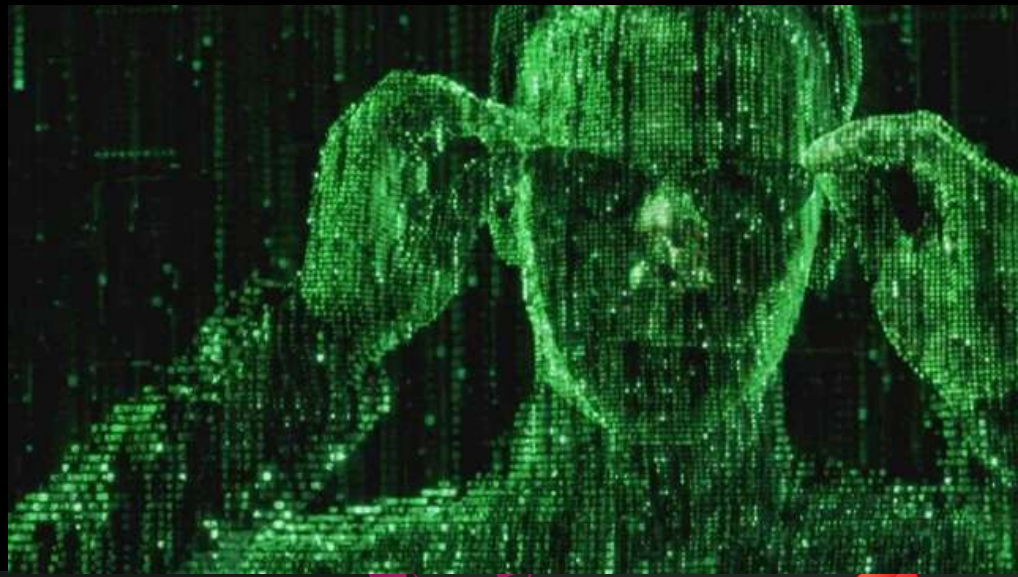
```
class MyCollection<T> : IEnumerable<T>
{
  MyEnumerator GetEnumerator()
  {
      return new MyEnumerator();
  }

  public struct MyEnumerator:IEnumerator<T>
  {
    ...
  }
}
```

Entity-Component System

```
entitySystem.Query()
    .Has(Component.Position)
    .Has(Component.Mover)
    .HasNot(Component.Immobile)
    .HasNot(Component.Death)

    .Do(MovementProcessor.Process);
```

```
static class MovementProcessor
{
  static void Process(Entity e)
  {
    Mover mover =
      e.GetComponent(Component.Mover);
    ProcessMover(mover);
  }
  ...
}
```

```
struct Entity { int ID; }

class EntitySystem
{
    StructCollection<EntityInfo> mEntities = new …(kMaxEntities);

    EntityQuery Query() { ... }
}

struct EntityQuery : IEnumerable<Entity>
{
    public EntityEnumerator GetEnumerator() { ... }
}

struct EntityEnumerator : IEnumerator<Entity> { ... }
```



Summary

Unity games can look as unique as we are capable of making them.

Deterministic simulation is still the state of the art for RTS multiplayer.

C# is ready for AAA game development.

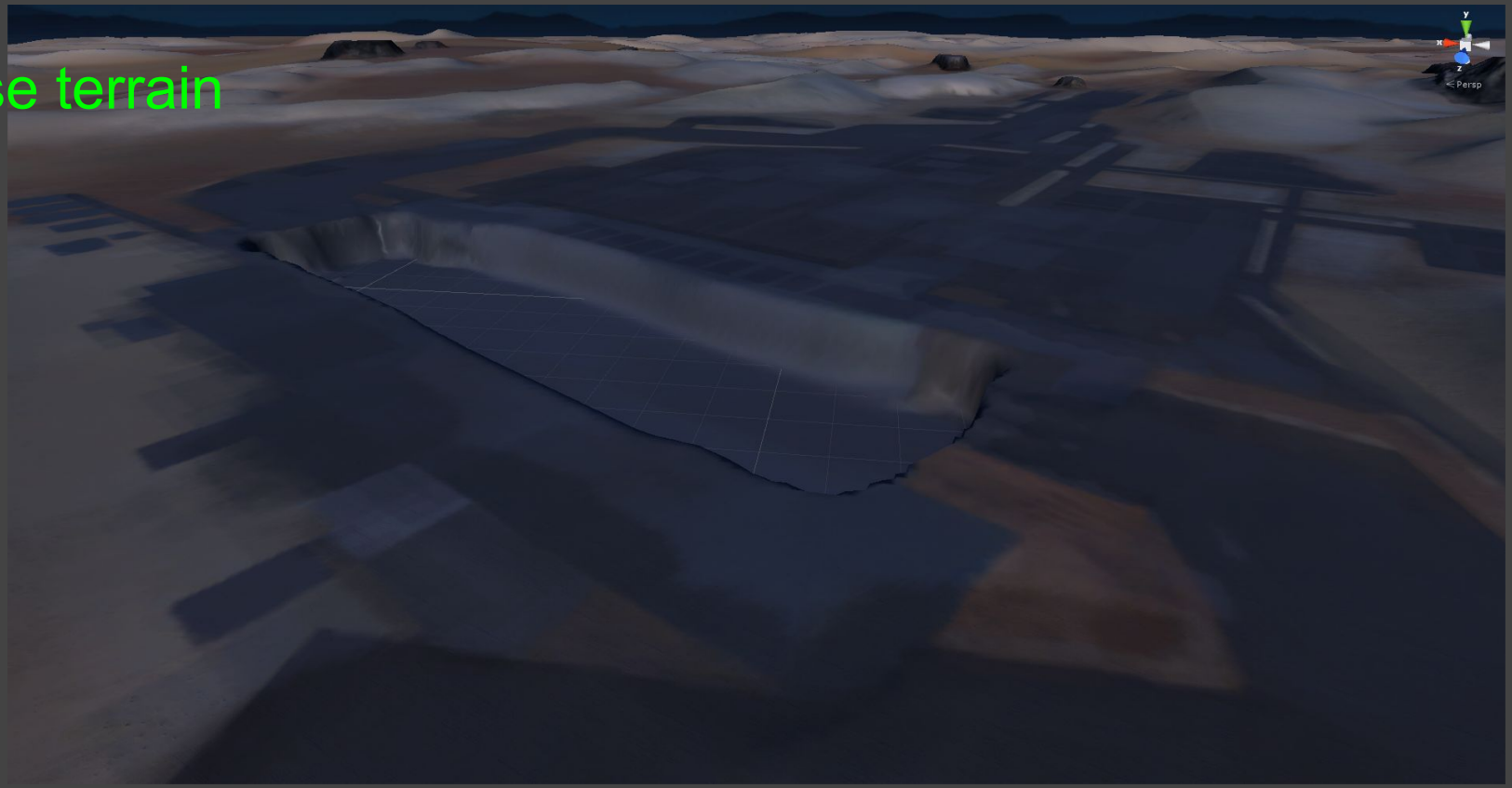
Many thanks!

- Everyone at Blackbird
- Gearbox Software
- The giants on whose shoulders we stand
- You for watching!

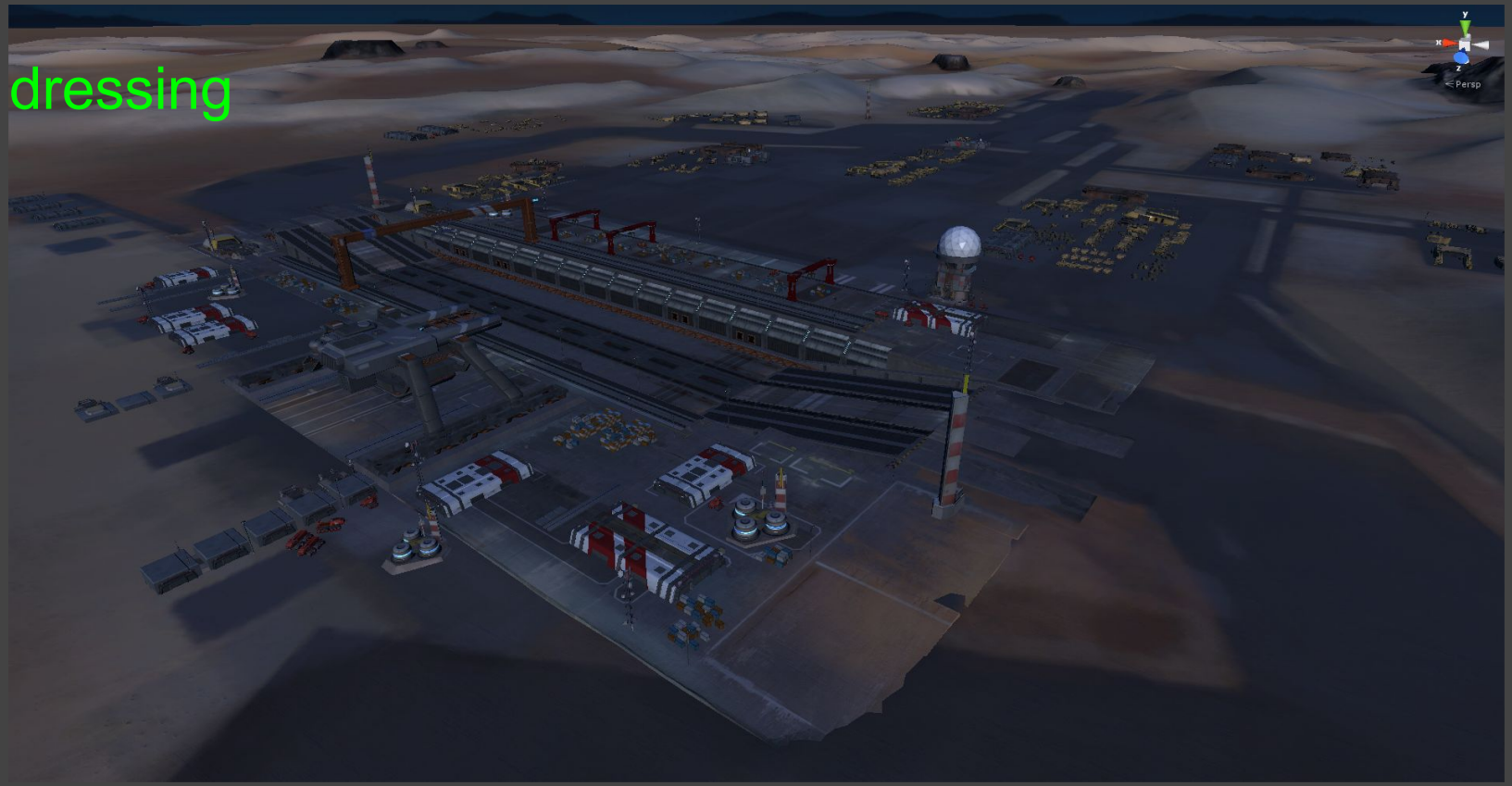
Magpie842, "Deserts of Kharak - Artifact Cup #5 Grand finals", YouTube

night-time example

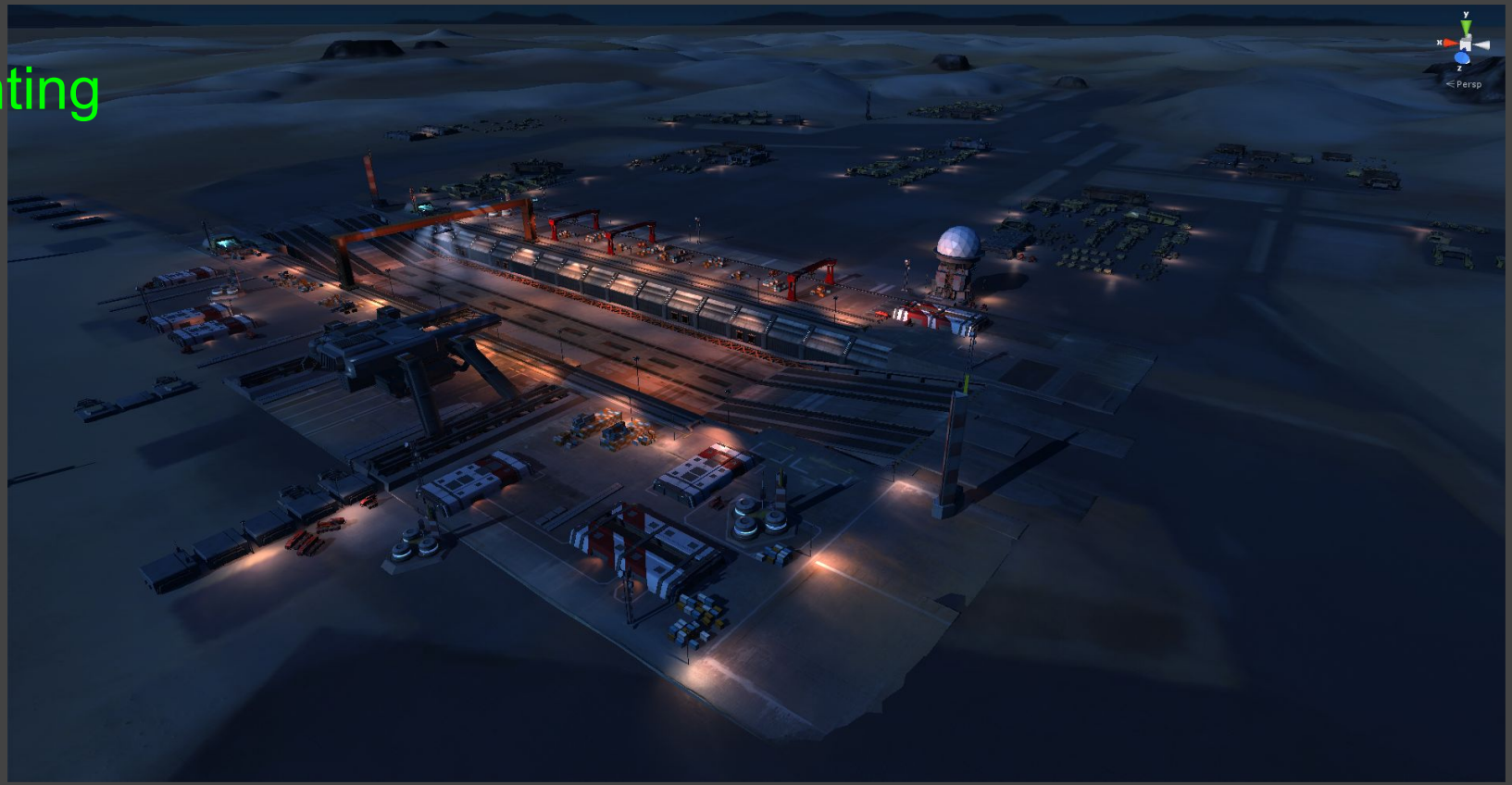
base terrain



set dressing



lighting



decals