



< Persp

Stefan Kraft

▼ Main Camera

Hello
Hello Einführung
18.10.2021

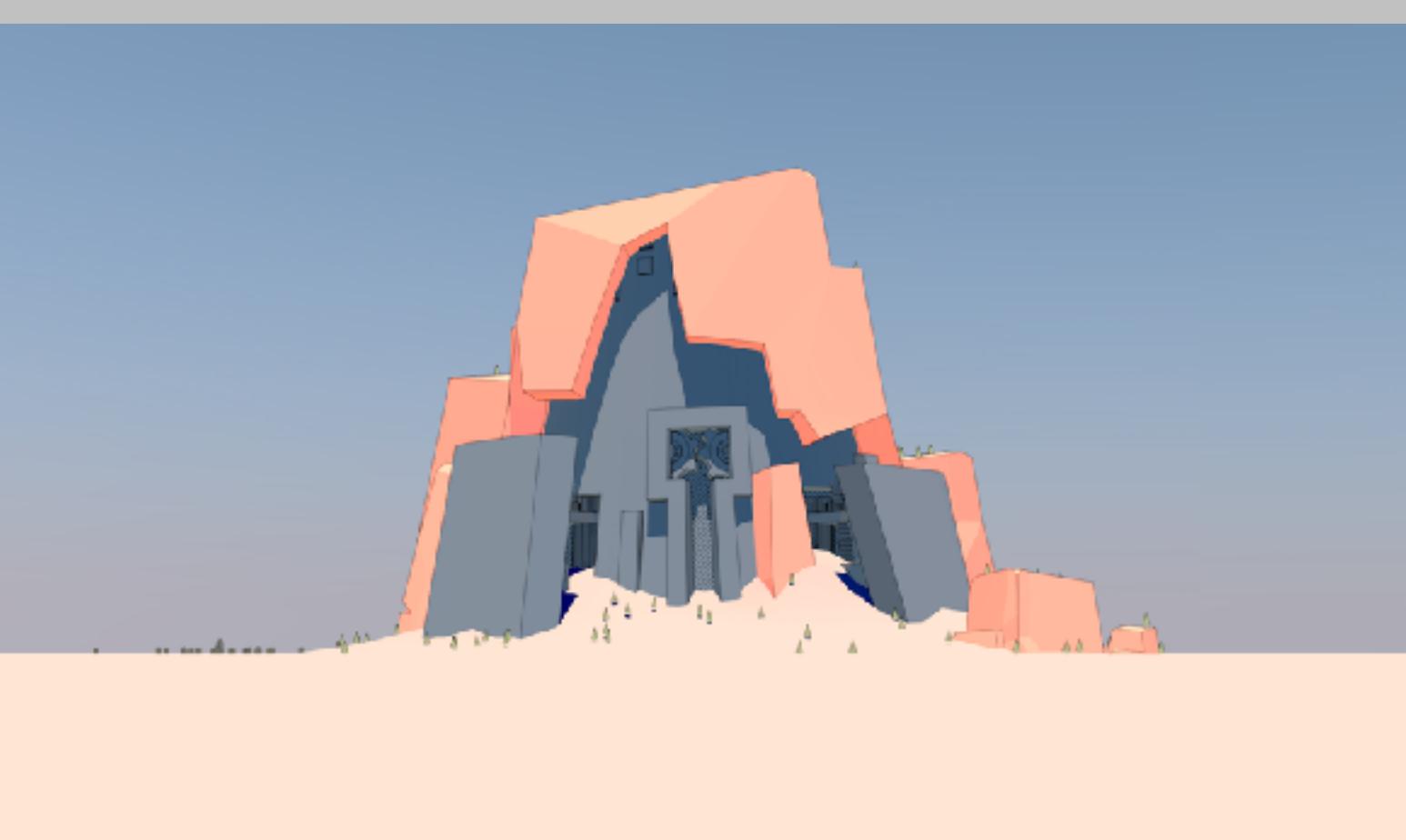
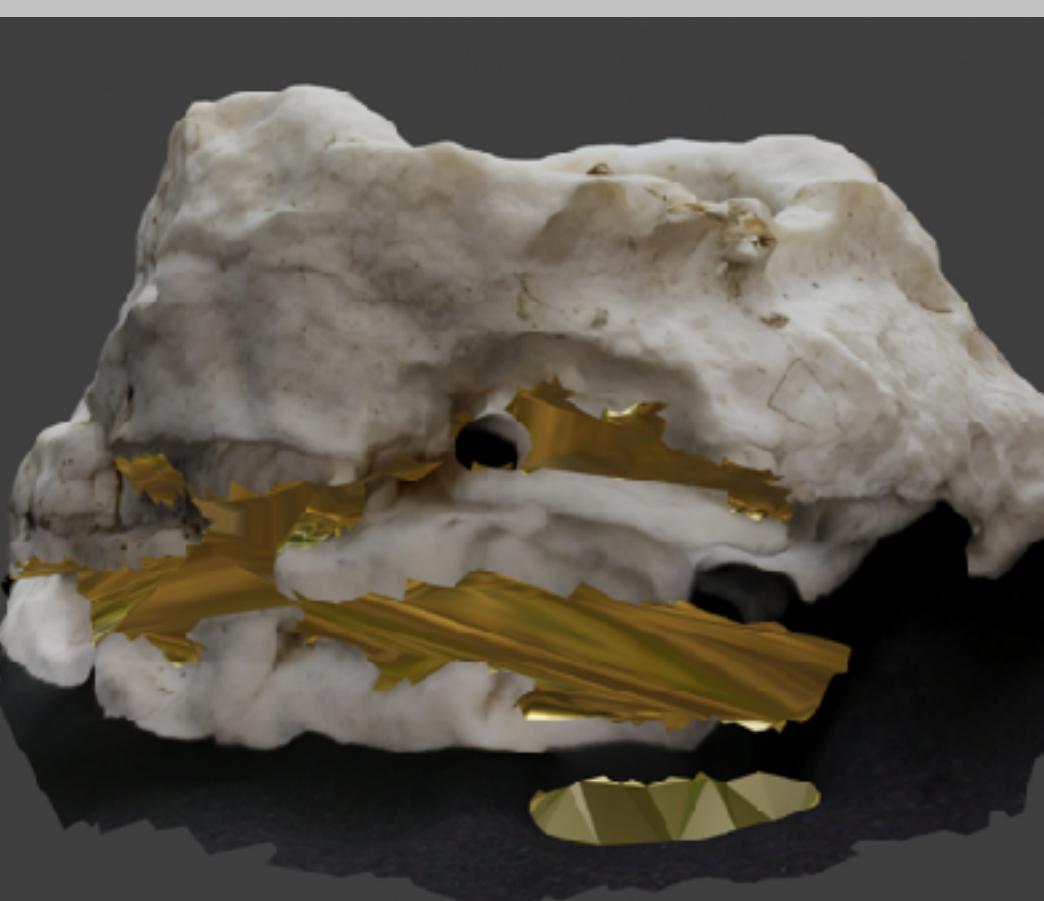
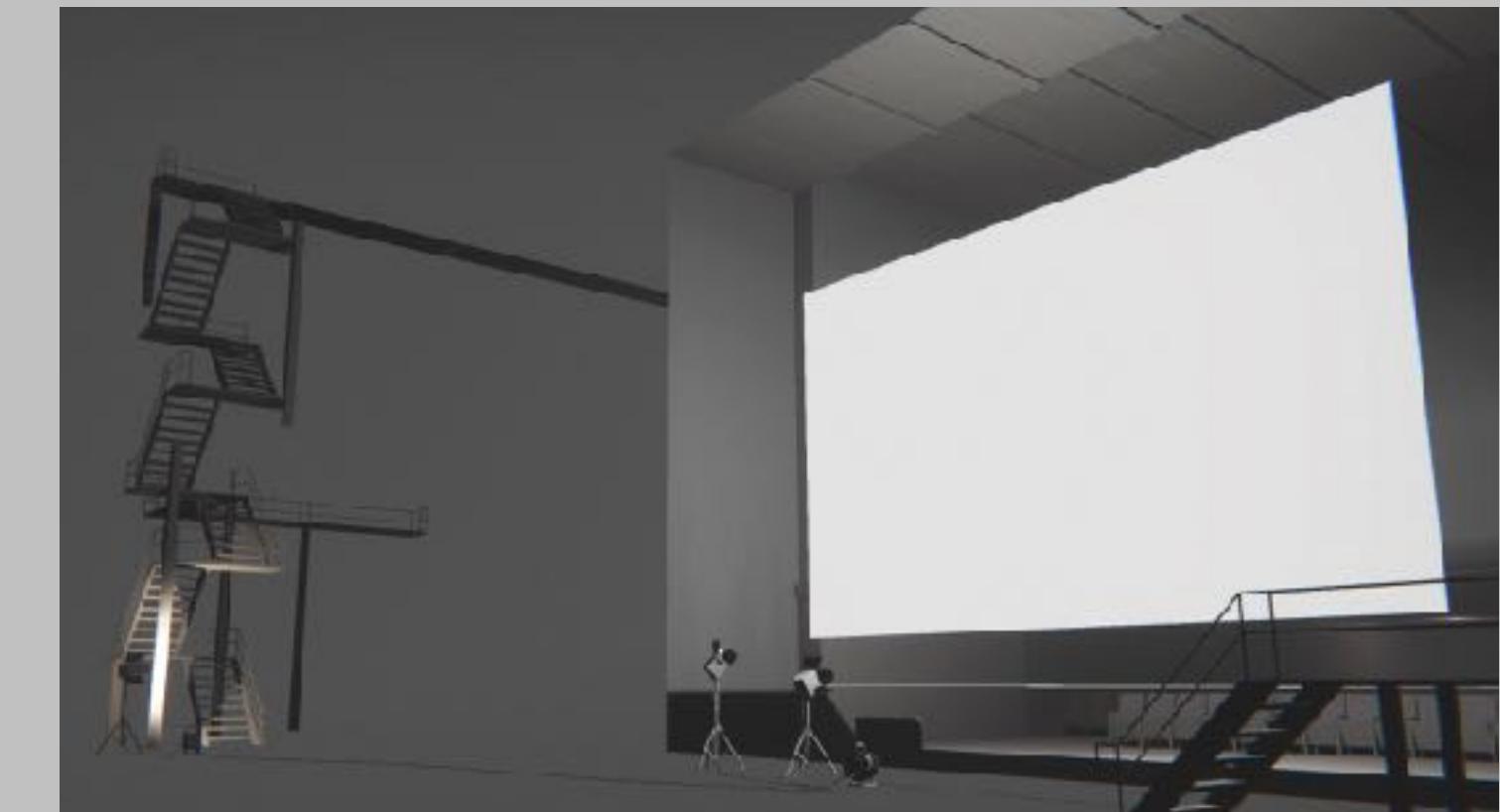
Unity Einführung
18.10.2021

Unity Version 2020.3.xx (LTS)

oder neuer

oder älter, spielt nicht so eine Rolle

Bei den Modulen mindestens *Visual Studio* installieren, sonst nach belieben.
(Kann man alles auch im Nachhinein noch hinzufügen.)





Dichtestress, 2020

<https://www.zhaw.ch/index.php?id=13049>

👉 Game Engines

👉 Einführung Unity

👉 Tutorial: Walking Sim

 Game Engines

 Einführung Unity

 Tutorial: Walking Sim

 Game Engines?

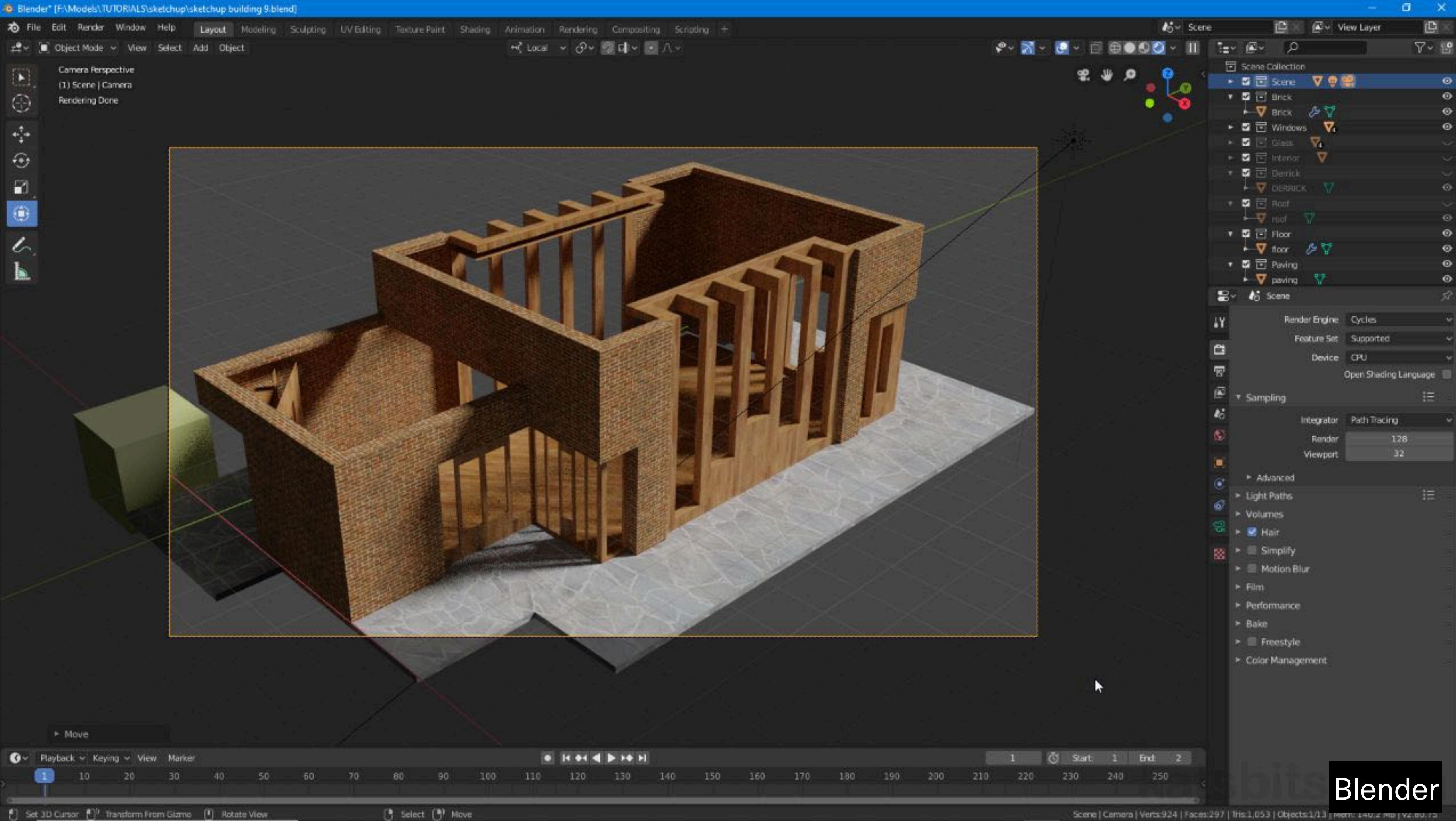
 Einführung Unity

 Tutorial: Walking Sim

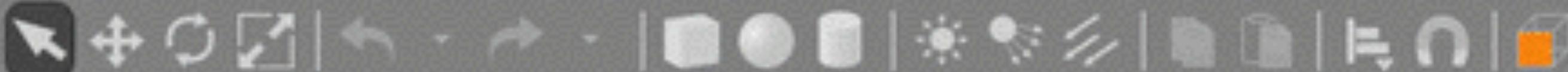
Unity ist eine Game Engine

Was ist eine Game Engine?

Was kann eine Game Engine?



World Insert Layers



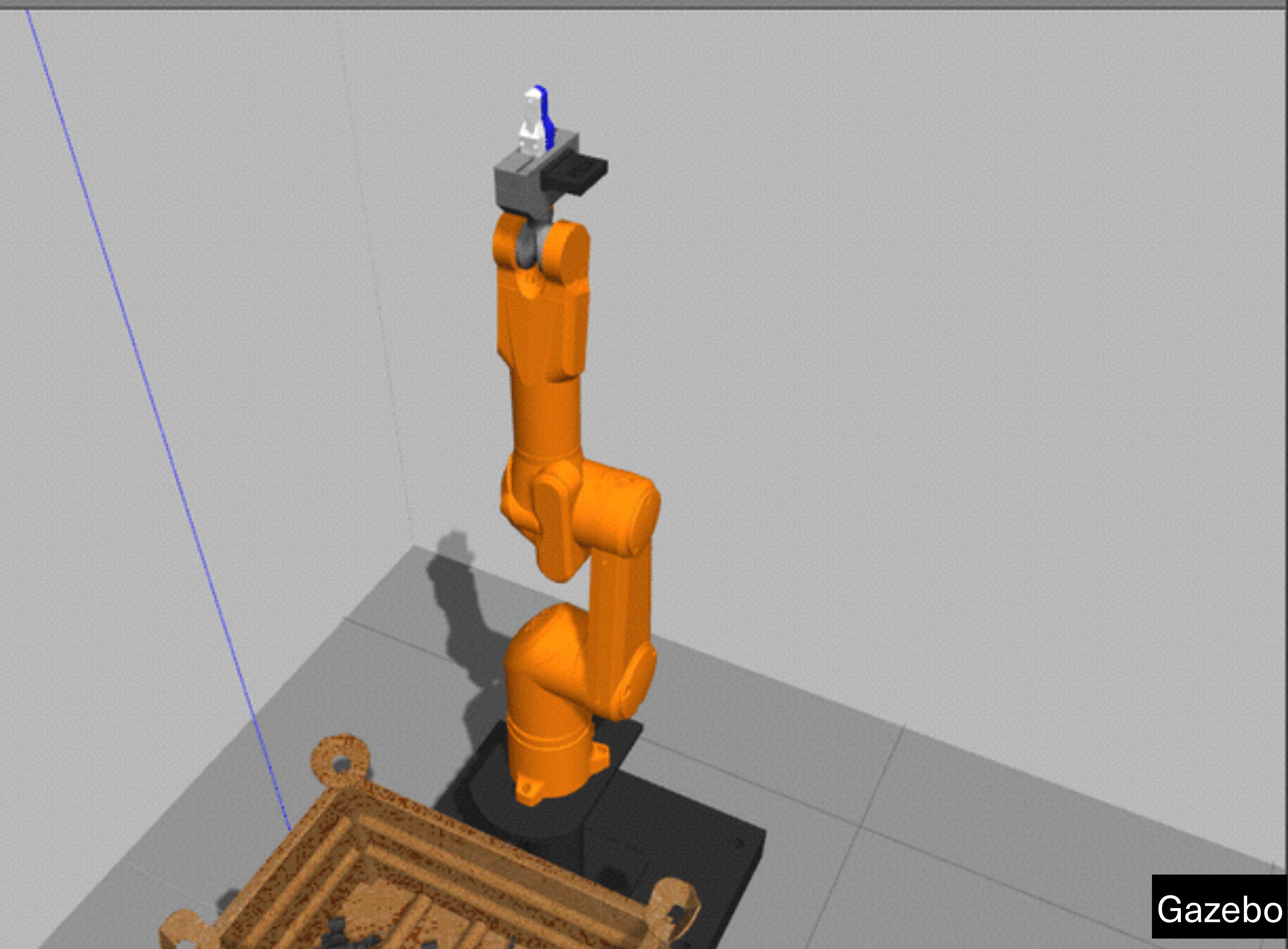
- pieza_b1_clone_0
- pieza_b1_clone_7
- pieza_b1_clone_8
- pieza_b1_clone_9
- pieza_b1_clone_10
- pieza_b1_clone_11
- pieza_b1_clone_12
- pieza_b1_clone_13
- pieza_b1_clone_14
- pieza_b1_clone_15
- pieza_b1_clone_16
- pieza_b1_clone_17
- pieza_b1_clone_18
- pieza_b1_clone_19
- ▼ ground_plane

link

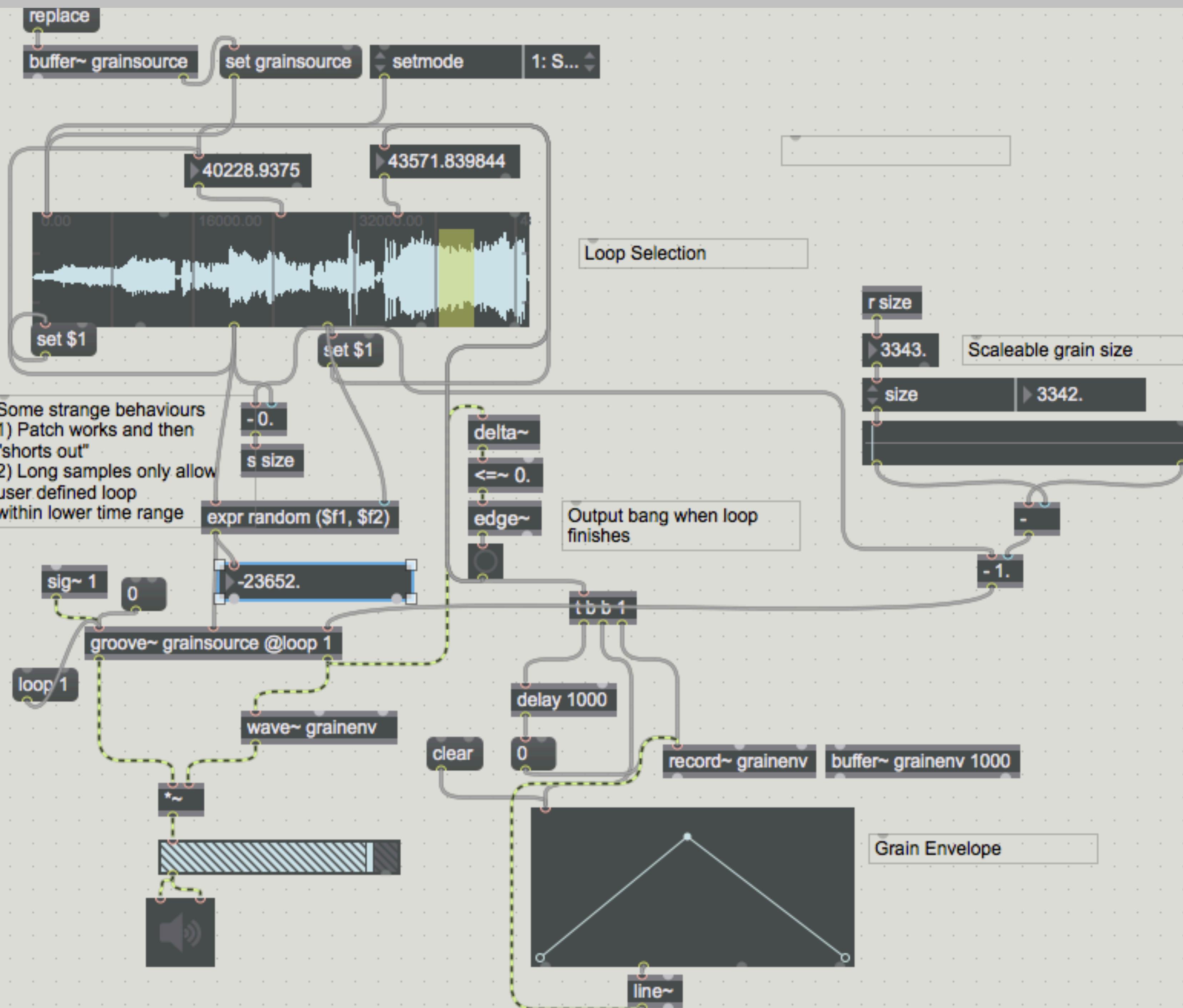
- Muro1

...

Property	Value
name	ground_plane
is_static	<input checked="" type="checkbox"/> True
self_collide	<input type="checkbox"/> False
► pose	
► link	ground_plane::link



Gazebo



Travel > iPhone 11 Pro Travel | Build Travel: Succeeded | Today at 9:41 AM

Travel

ReadMe.md

Travel

Travel.entitlements

AppDelegate.swift

Model

Discover

DiscoverView.swift

DiscoverTileView.swift

GlobeView.swift

DestinationsListView.swift

PagingScrollView.swift

CardsController.swift

Globe Scene

Plan

Journal

Journal.storyboard

StoryboardHostView.swift

JournalView.swift

JournalViewController.swift

JournalViewCell.swift

JournalTableController.swift

JournalDetailTableViewCell.swift

JournalPreviewViewCell.swift

JournalHeaderView.swift

JournalAddController.swift

JournalImagePicker.swift

RoundedCornerButton.swift

Weather

Login Screen

Login.storyboard

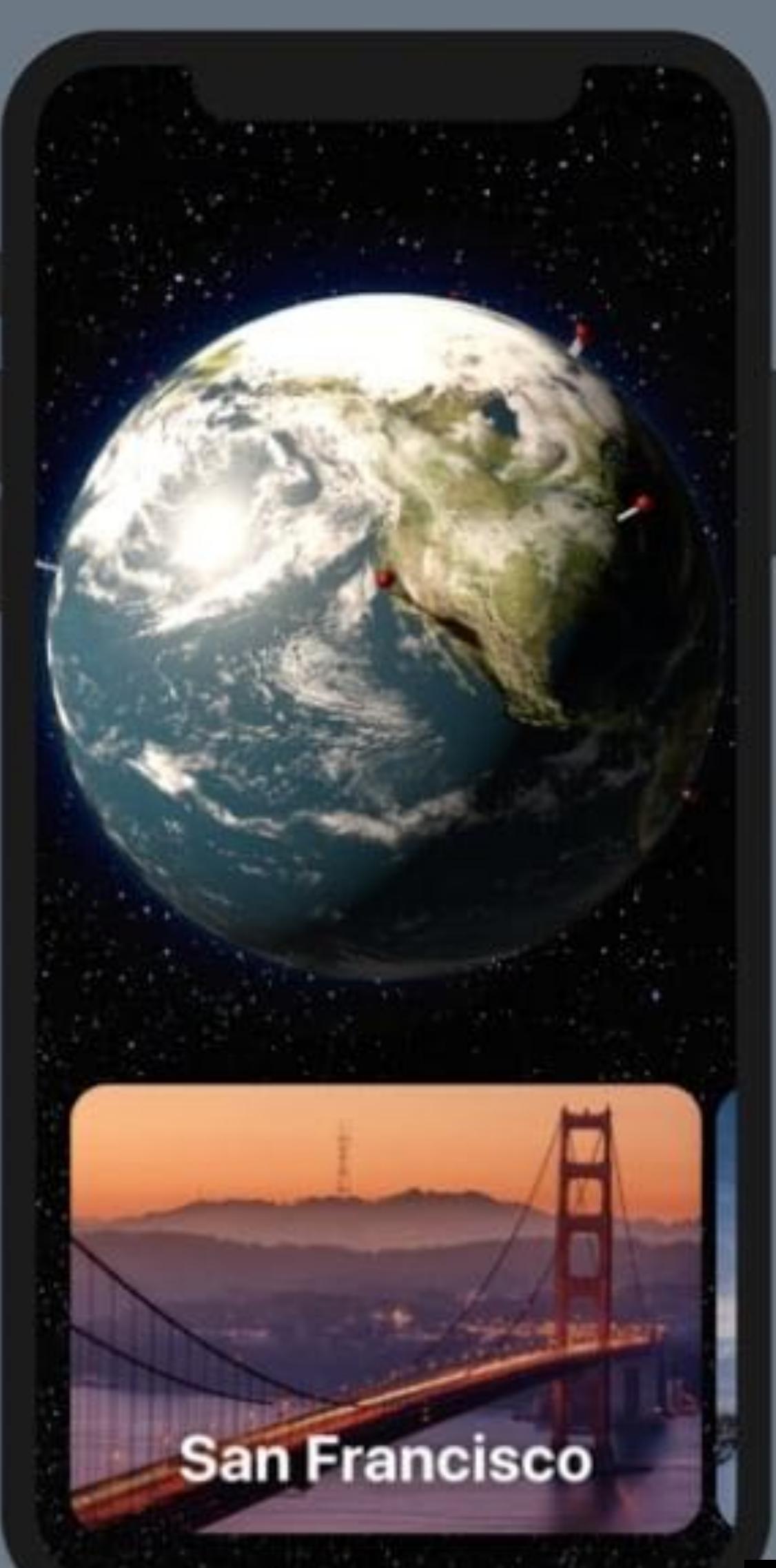
LoginViewController.swift

ForgotPasswordController.swift

ForgotPasswordController.xib

ForgotPasswordStatusView.swift

```
8 import SwiftUI
9
10 struct DiscoverView : View {
11     let sceneController: GlobeSceneController
12     @State private var selection: Region? = nil
13
14     var body: some View {
15         let pagingScrollViewController =
16             sceneController.pagingScrollViewController
17         pagingScrollViewController.didChangeToPageHandler = { page in
18             self.selection = DataSource.shared.regions[page]
19         }
20
21         return GeometryReader { container in
22             return ZStack(alignment: .bottom) {
23                 GlobeView(
24                     selection: self.$selection,
25                     sceneController: self.sceneController
26                 )
27
28                 PagingTilesView(
29                     containerSize: container.size,
30                     pagingScrollViewController: pagingScrollViewController
31                 ) { region in
32                     self.selection = region
33                 }
34             }
35             .background(Color.black)
36         }
37     }
38
39 struct PagingTilesView<T> : View where T : PagingScrollViewController {
40     let containerSize: CGSize
41     let pagingScrollViewController: T
42     var selectedTileAction: (Region) -> ()
43
44     var body: some View {
45         let tileSize = containerSize.width * 0.9
```



San Francisco

Rendering Engine

Animation

Physics Engine

Artificial Intelligence

Input

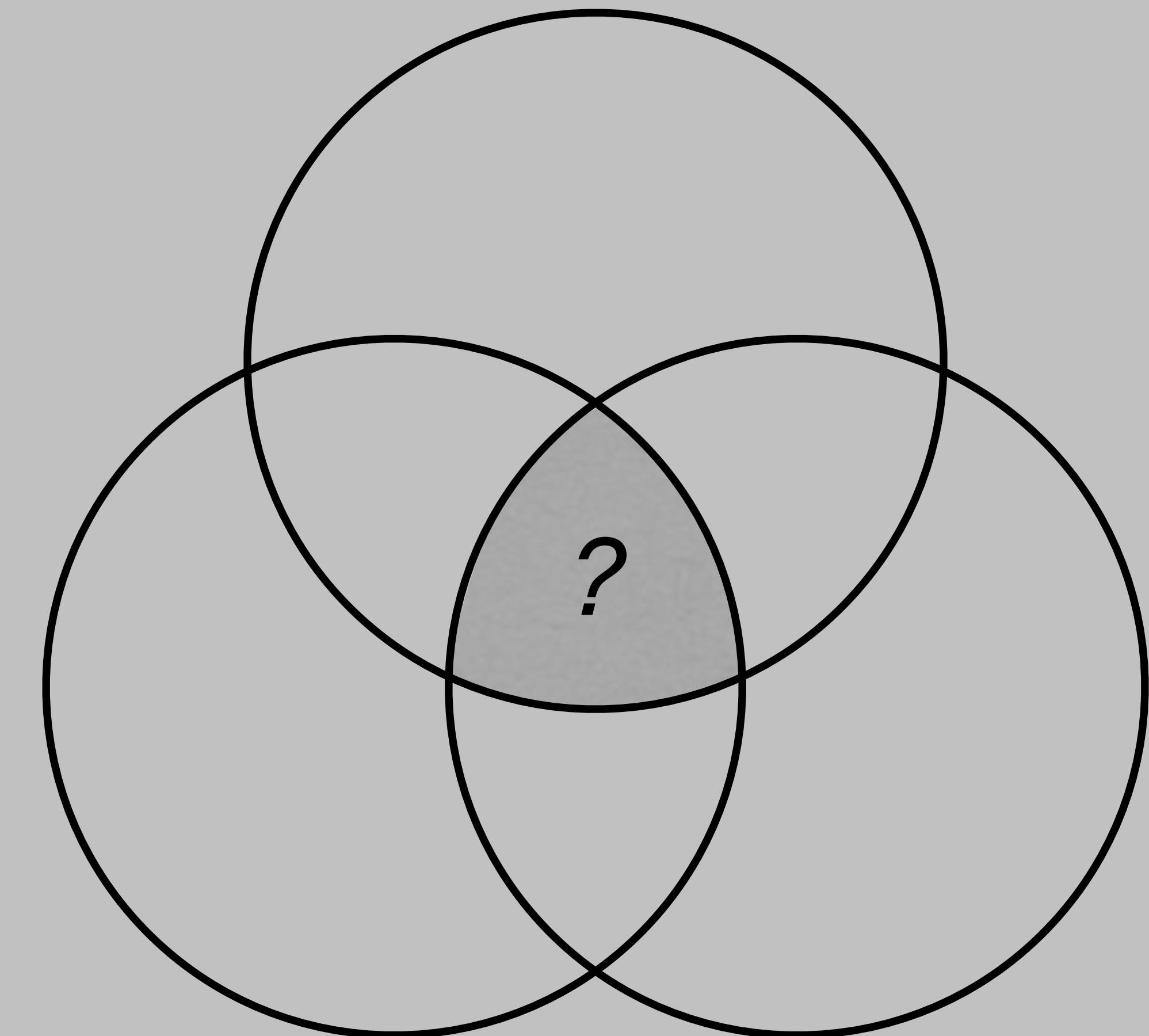
Networking

Sound

Memory Management

Scripting

Threading



Für was ist das Tool gedacht?

“Unity ist darauf ausgelegt, Games zu machen.”

“Unity ist darauf ausgelegt, (alle?/nur?) Games zu machen.”

Was ist ein Game für *Unity Technologies Inc.*?



Unity

Products

Solutions

Learni

Games

Games - Grow, Engage, Monetize

Automotive, Transportation & Manufacturing

Film, Animation & Cinematics

Architecture, Engineering & Construction

Government & Aerospace

Gambling

Accelerate Solutions

Verified Solutions Partners

Tools & Services of the game development lifecycle

Case Studies

Beispiel-Prioritäten einer Game Engine

- 👉 Rendering- und Berechnungs-Effizienz (Echtzeit)
- 👉 Photorealistisches 3D
- 👉 Einfache Implementation von Aspekten und Dingen aus bestehenden “Games”
- 👉 ...

 Game Engines

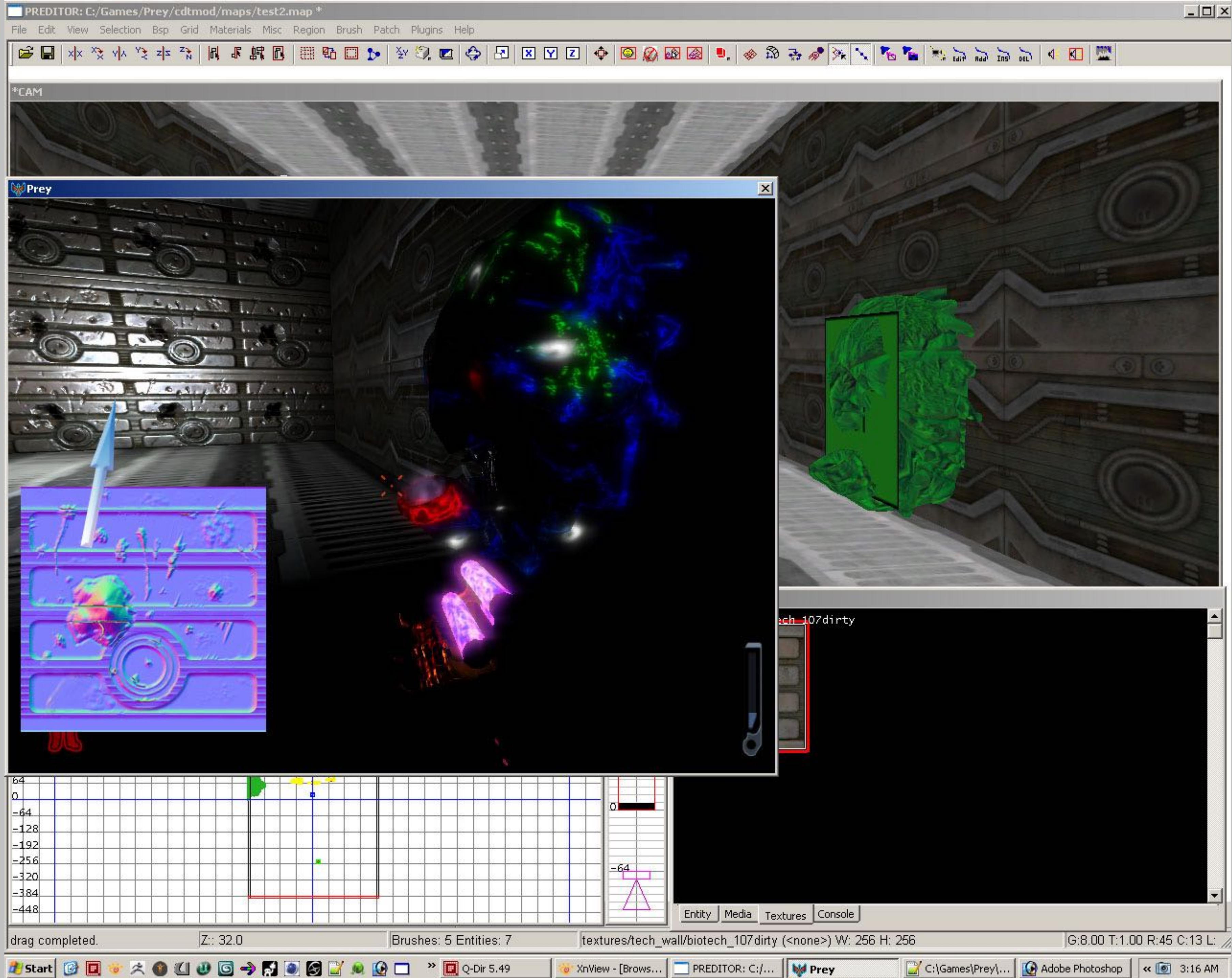
 Wieso Unity?

 Einführung Unity

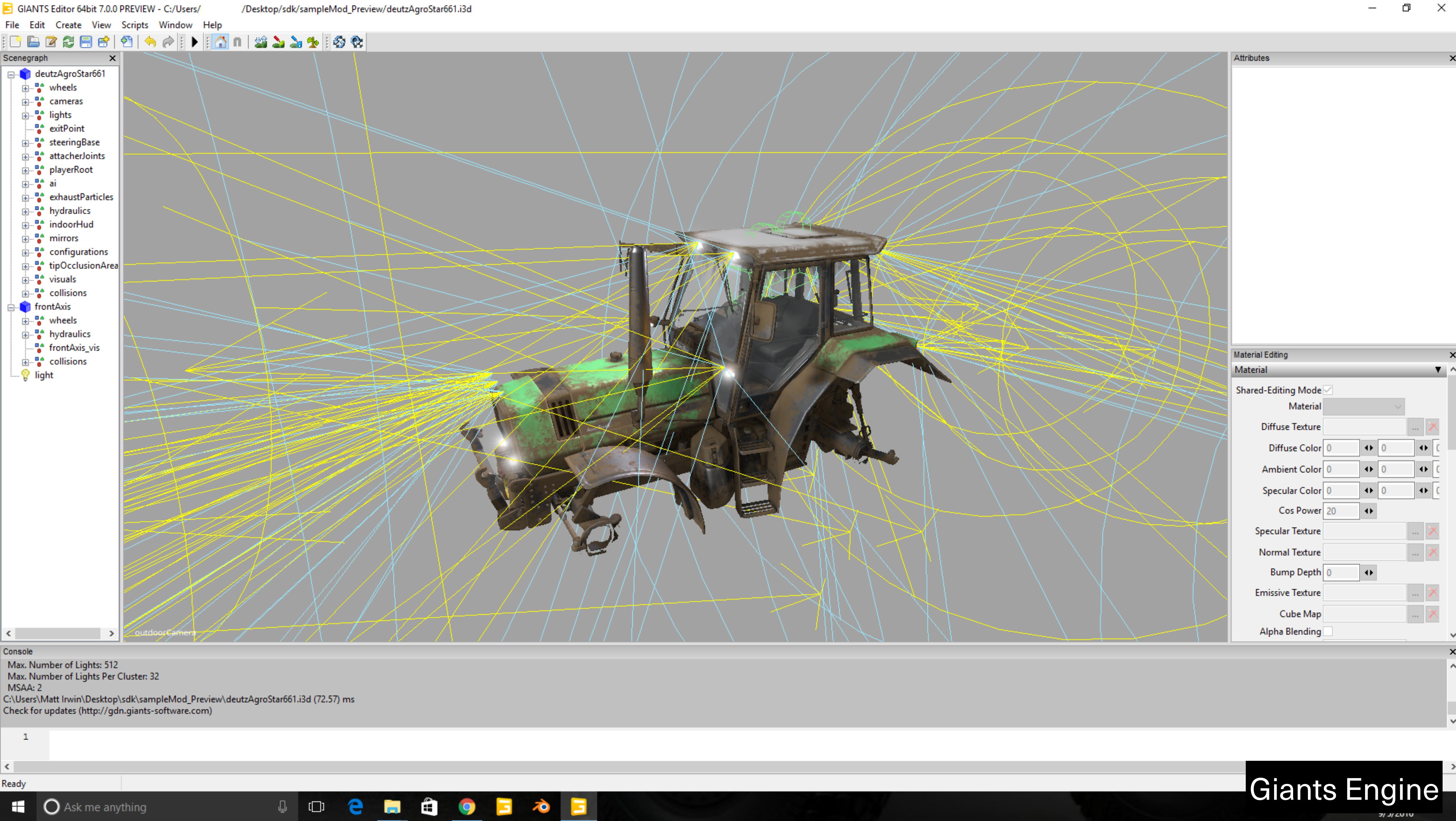
 Tutorial: Walking Sim

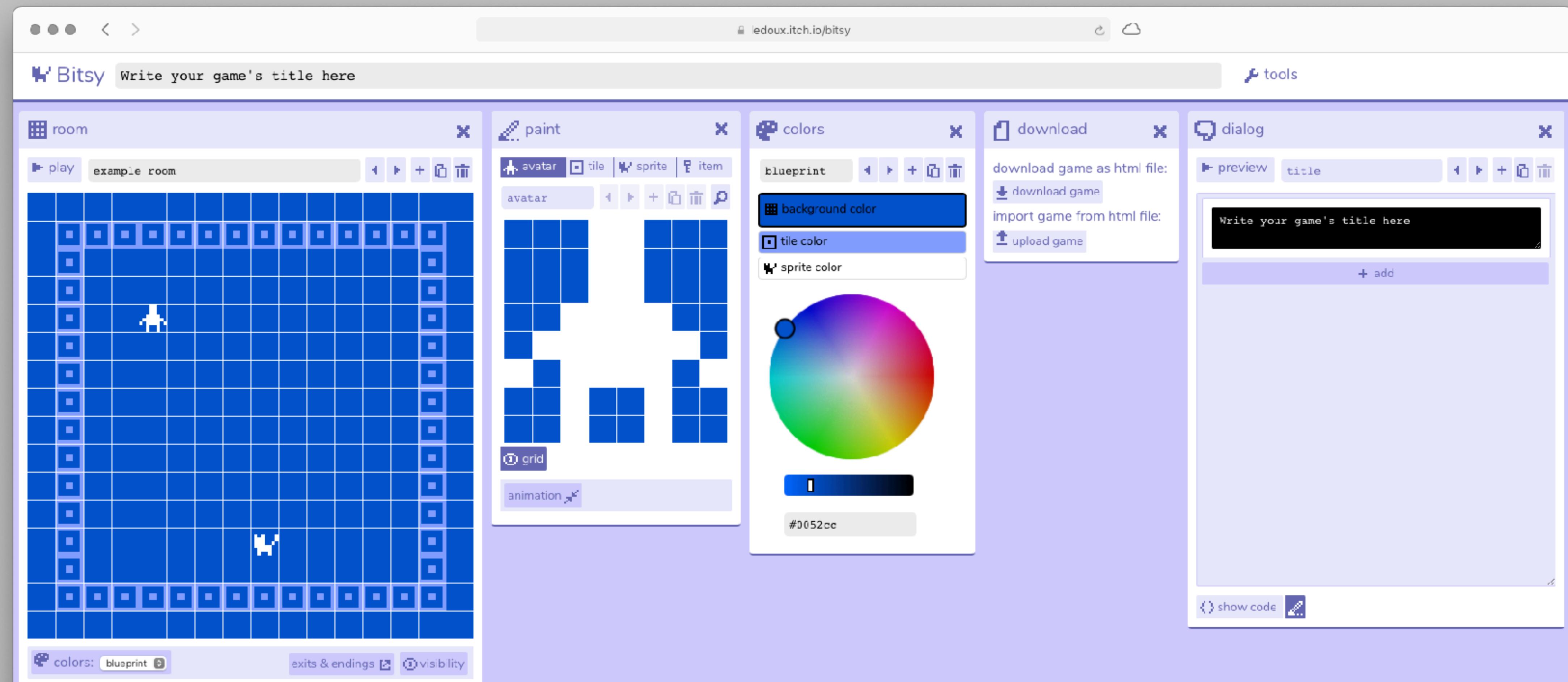


Unreal Engine 4

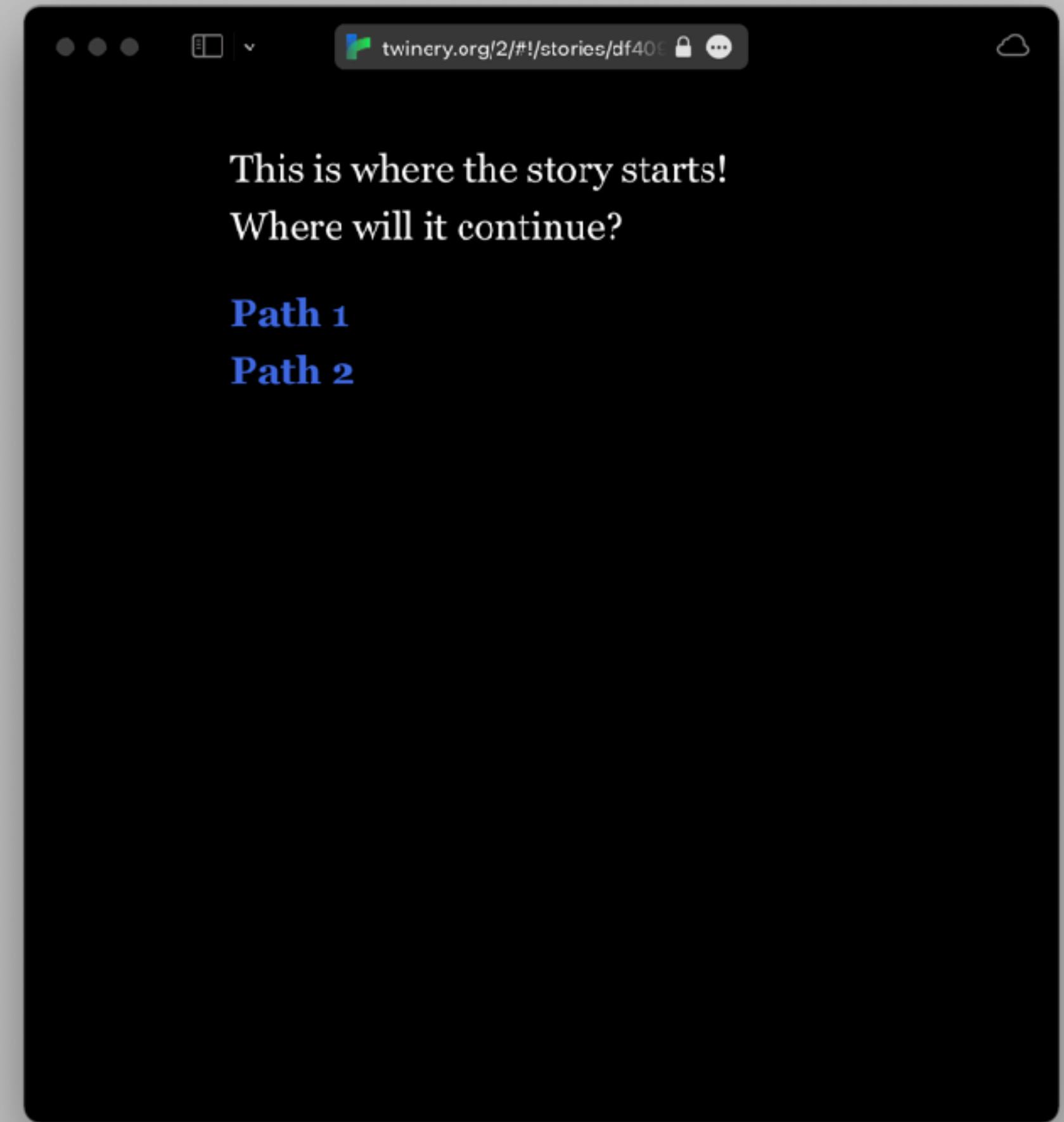
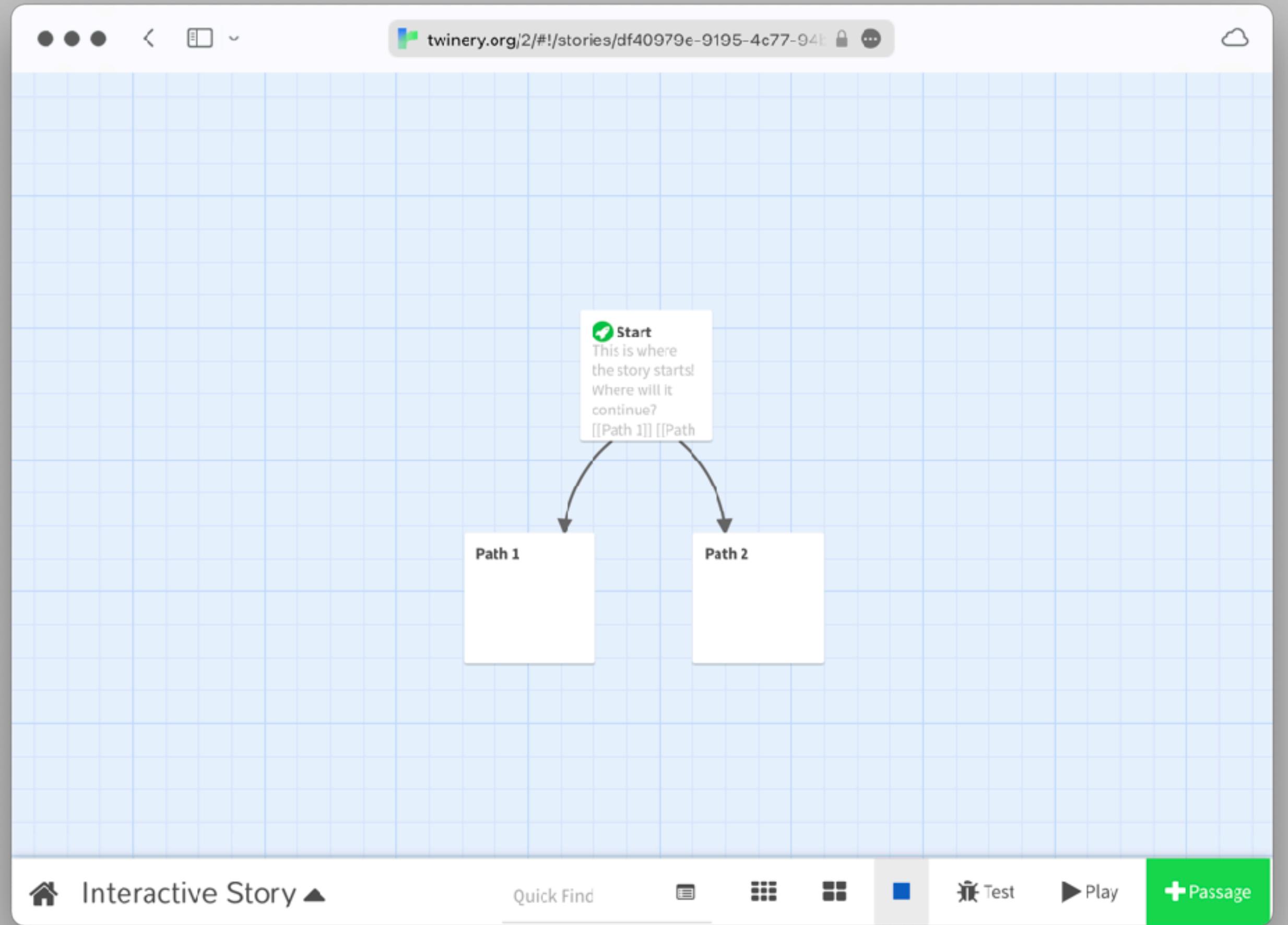


id Tech 4





Bitsy



Unity (Unity Technologies)

Unreal Engine (Epic Games)

Godot (Open Source Community <3)

GameMaker (YoYoGames)

Scratch (MIT Media Lab)

Phaser

LÖVE

ORGRE

Bitsy

PICO-8

Flickgame

Ren'Py

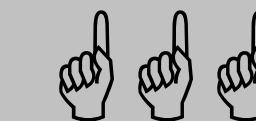
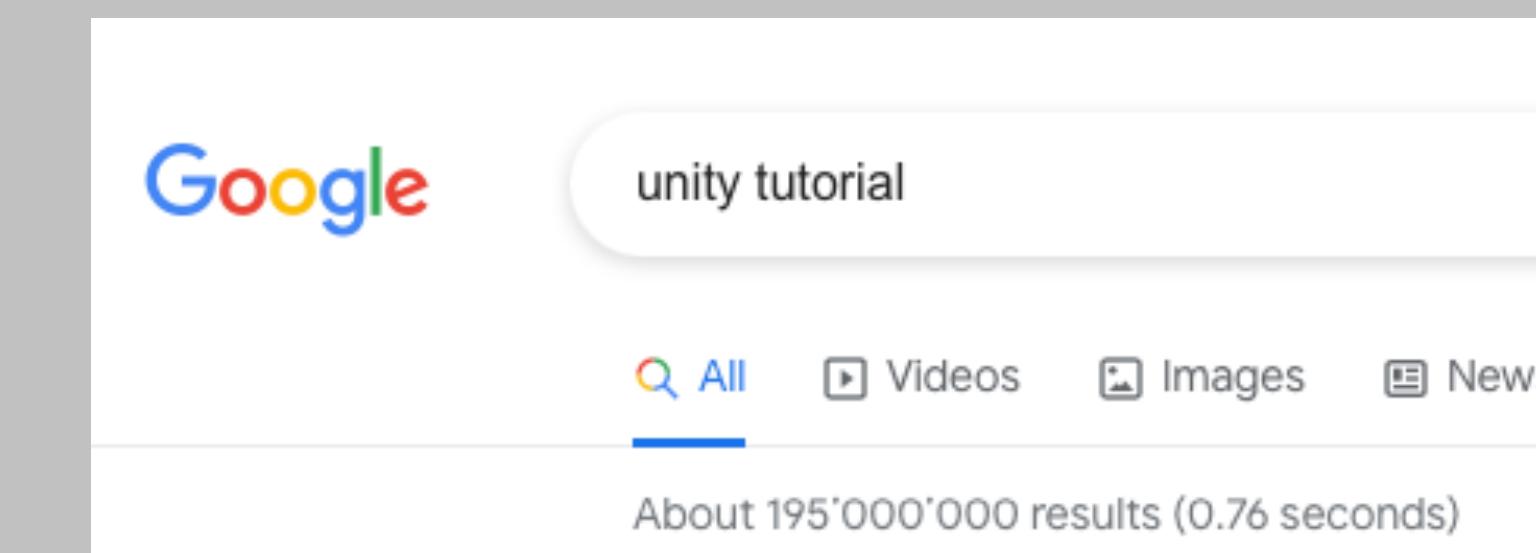
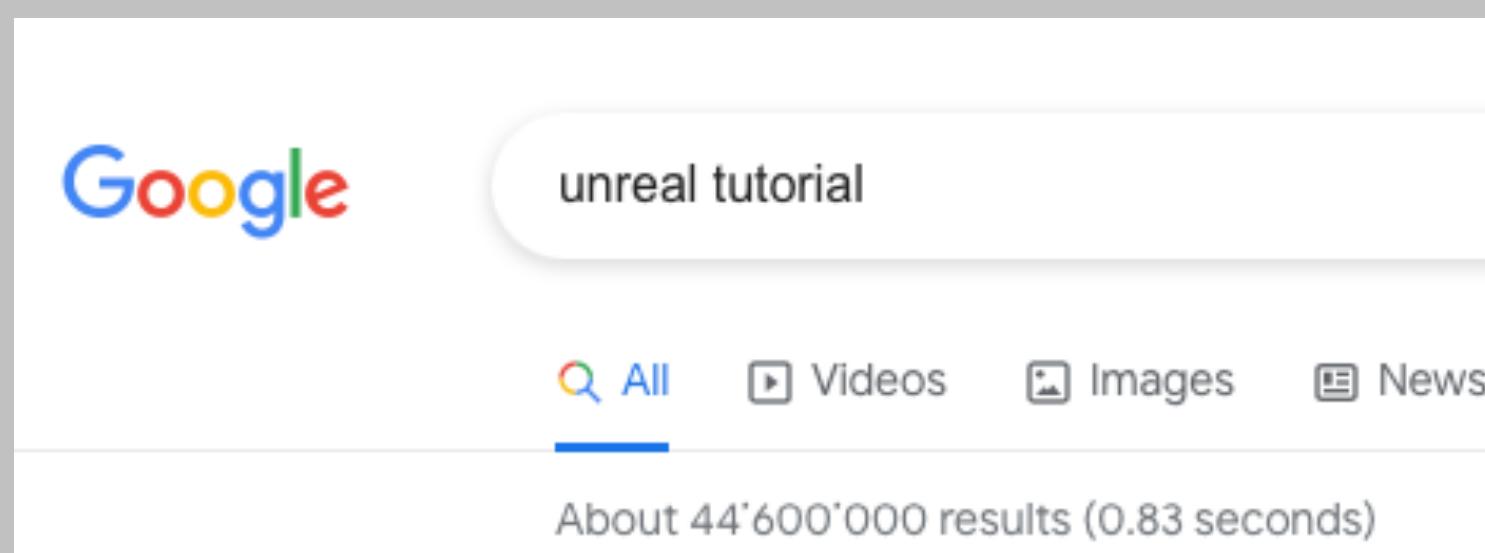
Twine

Roblox Studio

Nintendo Game Builder Garage

Dreams

Prävalenz, Ressourcen & Community



 Game Engines

 Einführung Unity

 Tutorial: Walking Sim



Navigation

Q, W, E, R, T für diese Werkzeuge in Folge:



Sicht Rotieren:

Alt + Drag (zentriert) / Rechtsklick (umhersehen)

Sicht Zoomen:

Alt + Ctrl + Drag / Mausrad

Sicht Bewegen:

Alt + Cmd / Mittlere Maustaste

First-Person Fly Navigation:

Rechtsklick + WASD, Q & E, Shift

Ausgewähltes Objekt zentrieren:

F

Hierarchy/Scene Graph

Scene

GameObject

GameObject

GameObject

GameObject

GameObject

GameObject

Hierarchy/Scene Graph

Scene

 Transform

 Transform

 Transform

 Transform

 Transform

 Transform

Komponenten eines 3D Objektes

Transform (notwendig für jedes GameObject) *Position, Rotation, Scale*

Mesh Filter *3D Geometrie*

Mesh Renderer *Wie wird die Geometrie dargestellt (Material, Licht, Schatten, ...)*

Weiteres, z.B.: Box Collider, Mesh Collider, Rigidbody, eigenes Script!, ...

 Game Engines

 Einführung Unity

 Tutorial: Walking Sim

Aufbau

1. Modell(e) importieren
2. First Person Controller
3. *Spiel mit Form, Position, Grösse*
4. Szenenwechsel
5. Build & Veröffentlichen

Aufbau

1. Modell(e) importieren

2. First Person Controller

3. *Spiel mit Form, Position, Grösse*

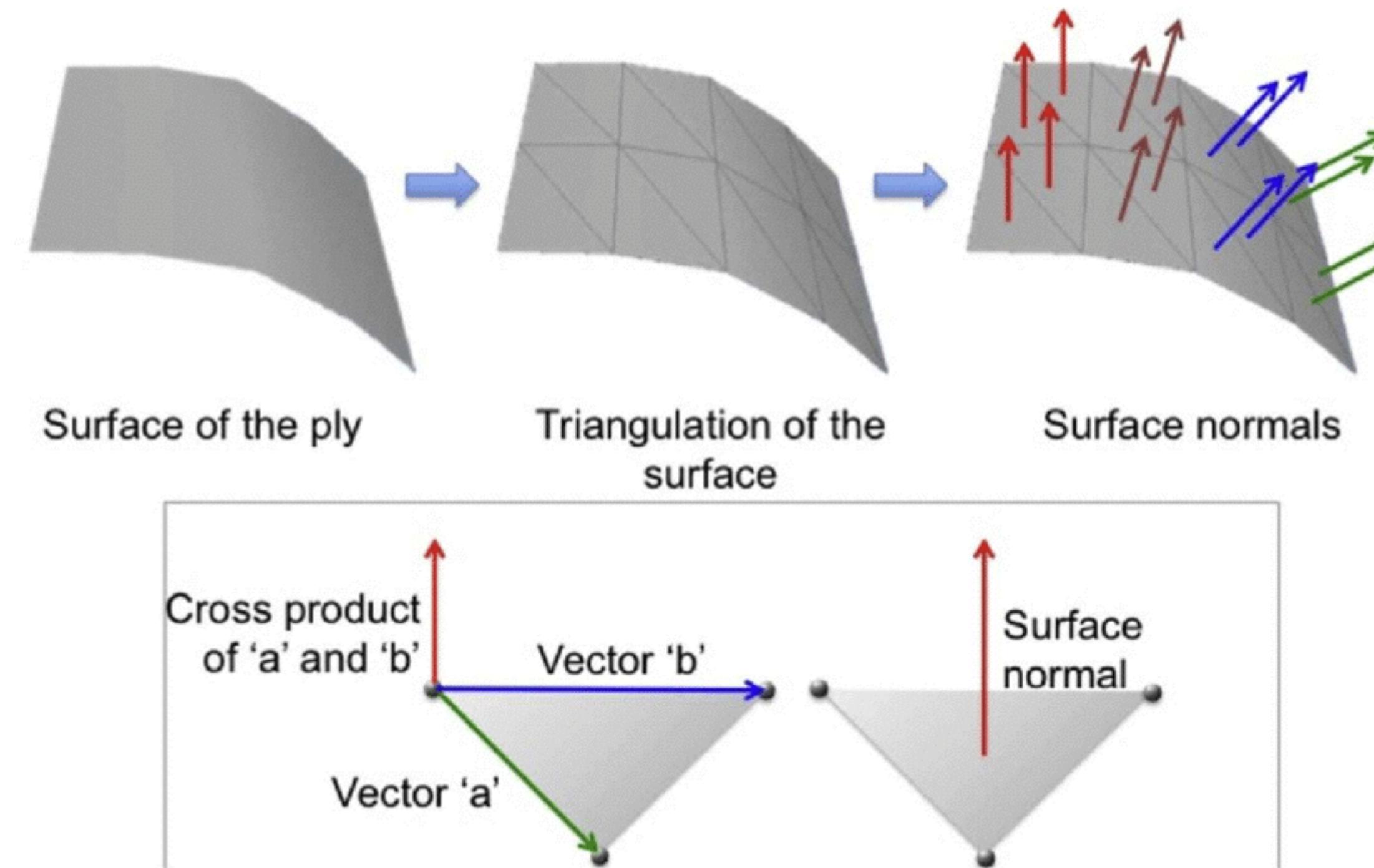
4. Szenenwechsel

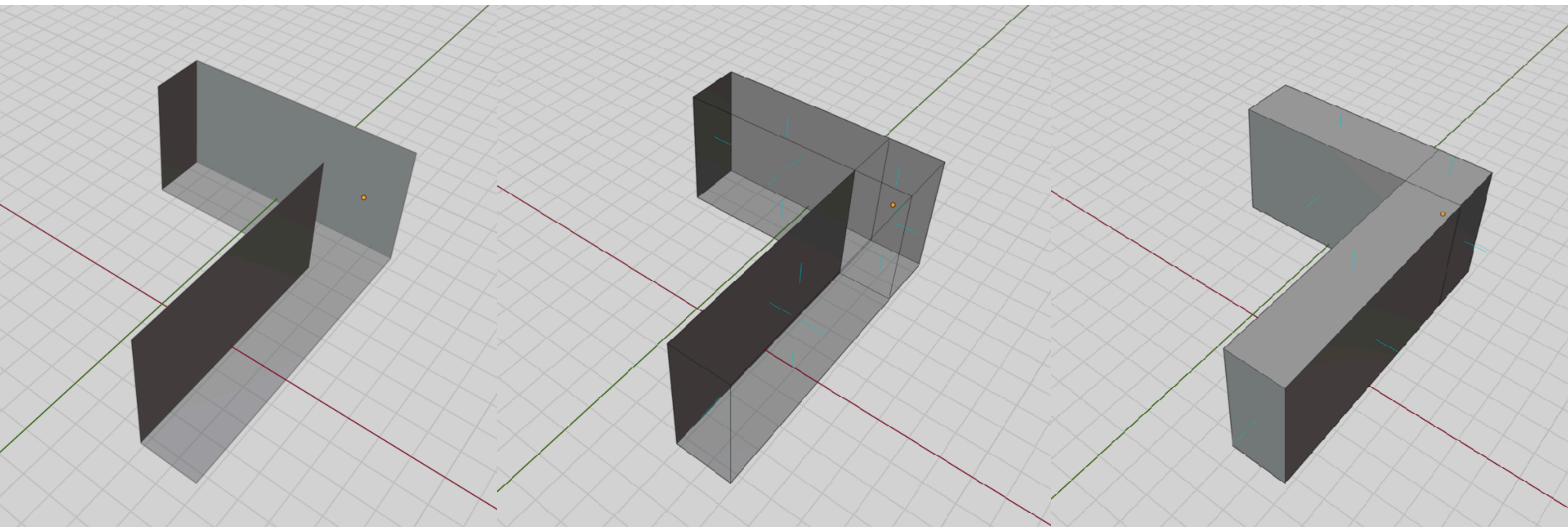
5. Build & Veröffentlichen

Modelle importieren ➡ Geometrie

1. Flächennormalen überprüfen

☞ Unity rendert (normalerweise) nur Vorderseite (positiv)



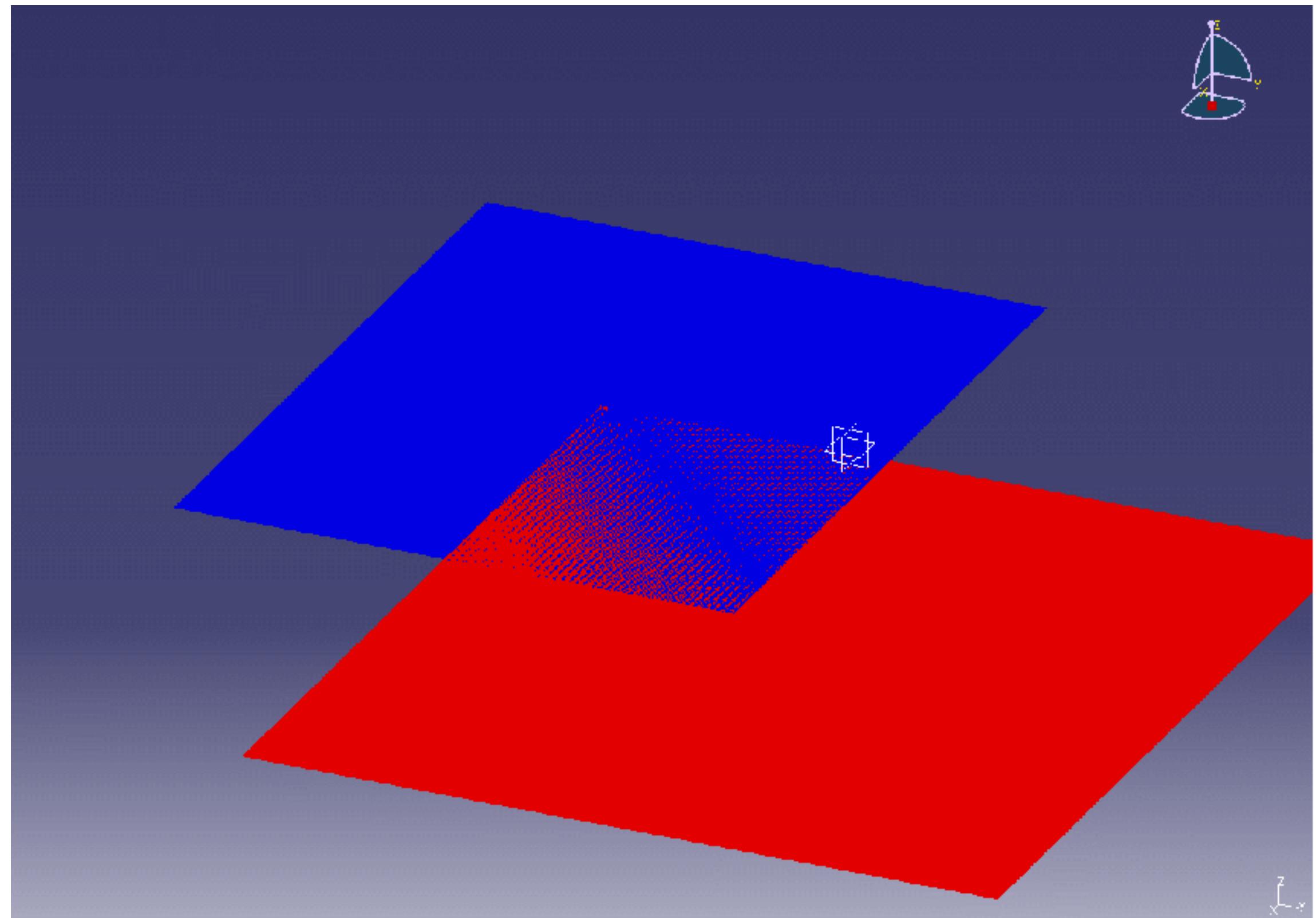


Modelle importieren ➔ Geometrie

1. Flächennormalen überprüfen

2. Doppelte Geometrie vermeiden

➔ Z-Fighting



Modelle importieren ➔ Geometrie

1. Flächennormalen überprüfen
2. Doppelte Geometrie vermeiden
3. Polygon-Anzahl möglichst tief halten

Modelle importieren ➡ Geometrie

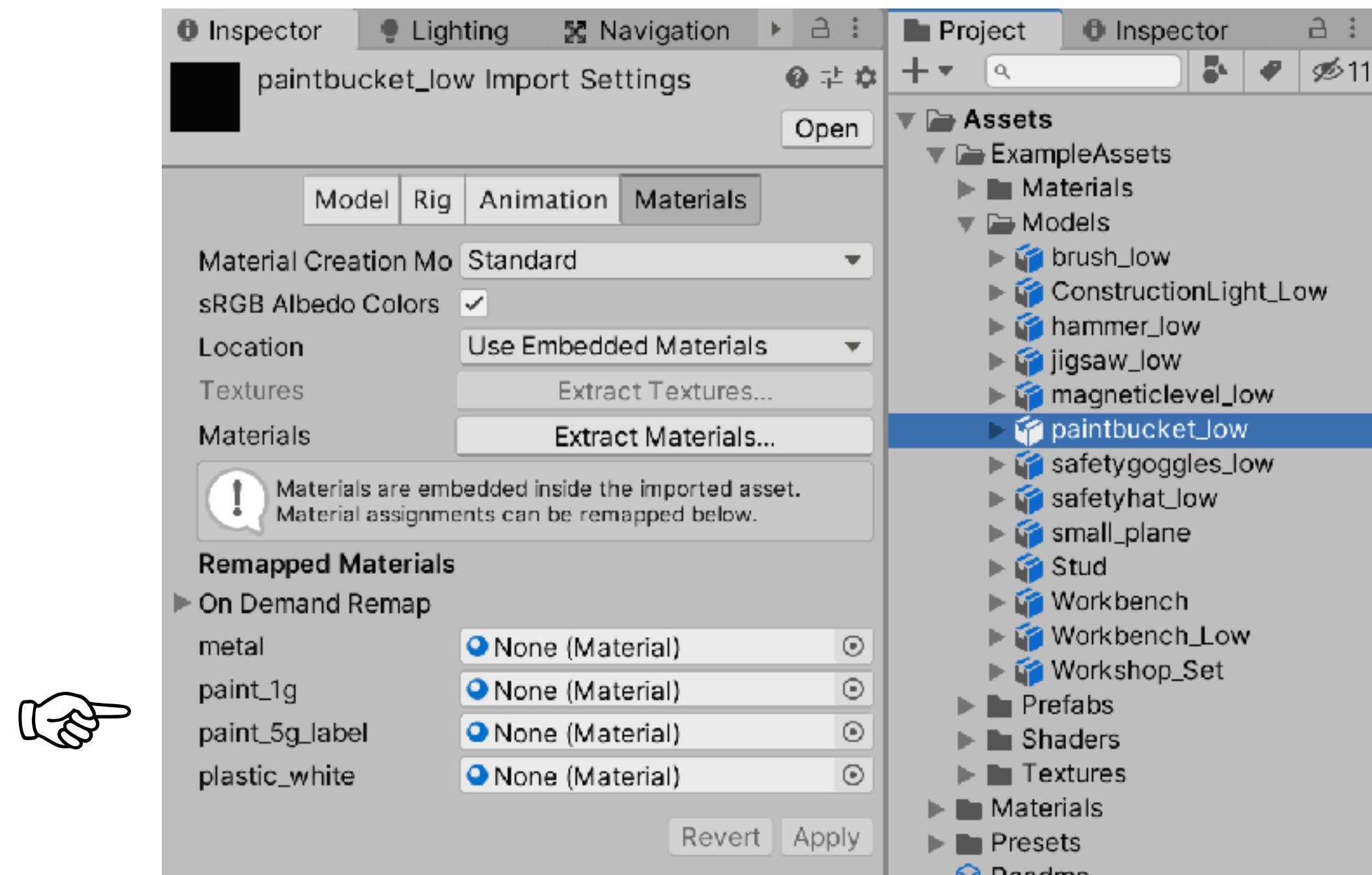
1. Flächennormalen überprüfen
2. Doppelte Geometrie vermeiden
3. Polygon-Anzahl möglichst tief halten
4. .fbx Files sind 

Modelle importieren ➔ Materialien

1. Materialien für das Modell müssen in Unity ersetzt/zugewiesen werden

Modelle importieren ➡ Materialien

1. Materialien für das Modell müssen in Unity ersetzt/zugewiesen werden
2. Unity kann Materialien von importierten .fbx erkennen und ersetzen



Modelle importieren ➔ Materialien

1. Materialien für das Modell müssen in Unity ersetzt/zugewiesen werden
2. Unity kann Materialien von importierten .fbx erkennen und ersetzen
3. Ausprobieren! :)

Modelle importieren ➔ Licht

1. Jedes Licht in der Szene kostet Performance

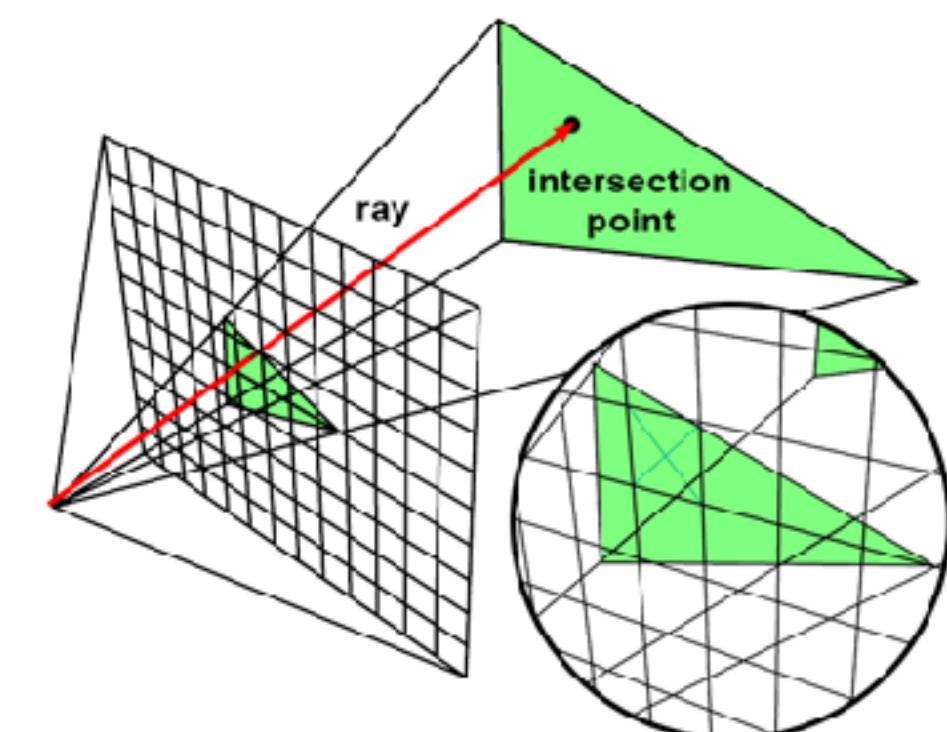
➔ Rendert die ganze Szene aus Sicht des Lichts

Modelle importieren ➔ Licht

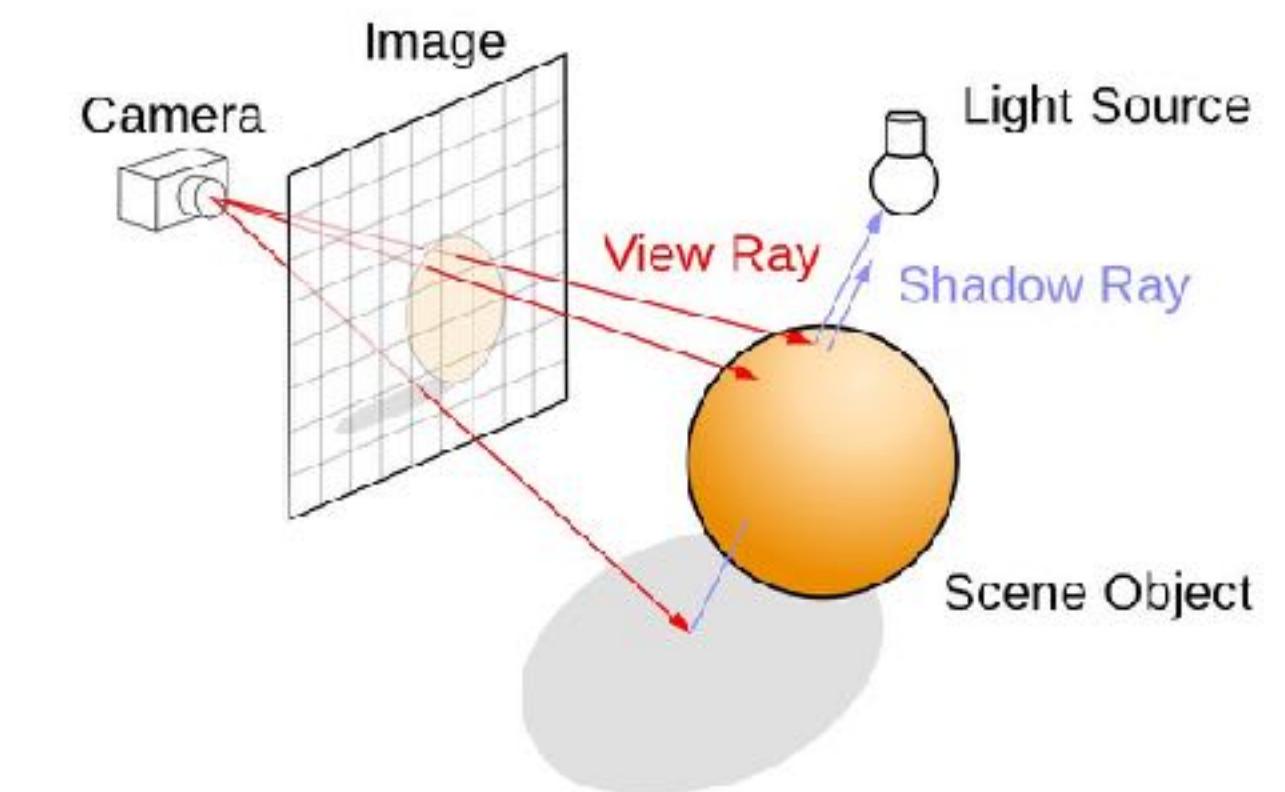
1. Jedes Licht in der Szene kostet Performance

2. Licht Settings erforschen (*Window > Rendering > Light Settings*)

➔ Light baking



Rasterisierung



Ray Tracing

First Person Controller

1. *FirstPersonDrifter.unitypackage* herunterladen und importieren
👉 <https://github.com/stekra/Unity-Intro-Packages>
2. Prefab “*Player*” in die Szene ziehen
3. Allfällige andere Kameras in der Szene löschen

Aufbau

1. Modell(e) importieren
2. First Person Controller
3. *Spiel mit Form, Position, Grösse*
4. Szenenwechsel
5. Build & Veröffentlichen

Form, Position, Grösse ➔ OnTriggerEnter()

ChangeMesh.cs

Wechselt das Mesh eines Objektes aus

```
using UnityEngine;

public class ChangeMesh : MonoBehaviour
{
    public GameObject targetObject;
    public Mesh newMesh;

    void OnTriggerEnter(Collider other)
    {
        targetObject.GetComponent<MeshFilter>().mesh = newMesh;
    }
}
```

ChangePosition.cs

Verschiebt ein Objekt

```
using UnityEngine;

public class ChangePosition : MonoBehaviour
{
    public GameObject targetObject;
    public Vector3 newRelativePosition;

    void OnTriggerEnter(Collider other)
    {
        targetObject.transform.position = targetObject.transform.position + newRelativePosition;
    }
}
```

Form, Position, Grösse ➡ OnTriggerStay()

RotateOnStay.cs

Rotiert ein Objekt solange ein Collider im Trigger ist

```
using UnityEngine;

public class RotateOnStay : MonoBehaviour
{
    public GameObject targetObject;
    public Vector3 rotationPerAxis;

    void OnTriggerEnter(Collider other)
    {
        targetObject.transform.Rotate(rotationPerAxis * Time.deltaTime);
    }
}
```

Form, Position, Grösse ➡ Distanz

SizeByDistance.cs

*Vergrößert/Verkleinert Objekt je nach
Distanz zu einem anderen Objekt*

```
using UnityEngine;

public class SizeByDistance : MonoBehaviour
{
    public GameObject otherObject;
    public Vector3 targetScale = Vector3.one;
    public float maxDistance = 20;
    Vector3 originalScale;

    void Start()
    {
        // Save the original scale of this object in the beginning
        originalScale = transform.localScale;
    }

    void Update()
    {
        // Calculate the distance between this object's position and the other's
        float distance = Vector3.Distance(transform.position, otherObject.transform.position);

        // Calculate how much percent of the way from 0 to maxDistance it is
        float percentage = Mathf.InverseLerp(0, maxDistance, distance);

        // Set the current scale of this object
        transform.localScale = Vector3.Lerp(targetScale, originalScale, percentage);
    }
}
```

FOVByDistance.cs

*Verändert Kamera-Bildwinkel (Field of View)
je nach Distanz zu einem anderen Objekt*

```
using UnityEngine;

public class FOVByDistance : MonoBehaviour
{
    public GameObject otherObject;
    public float FOVWhenClose = 25;
    public float maxDistance = 5;
    float originalFOV;

    void Start()
    {
        originalFOV = GetComponent<Camera>().fieldOfView;
    }

    void Update()
    {
        float distance = Vector3.Distance(transform.position, otherObject.transform.position);

        float percentage = Mathf.InverseLerp(maxDistance, 0, distance);

        GetComponent<Camera>().fieldOfView = Mathf.Lerp(originalFOV, FOVWhenClose, percentage);
    }
}
```

Aufbau

1. Modell(e) importieren

2. First Person Controller

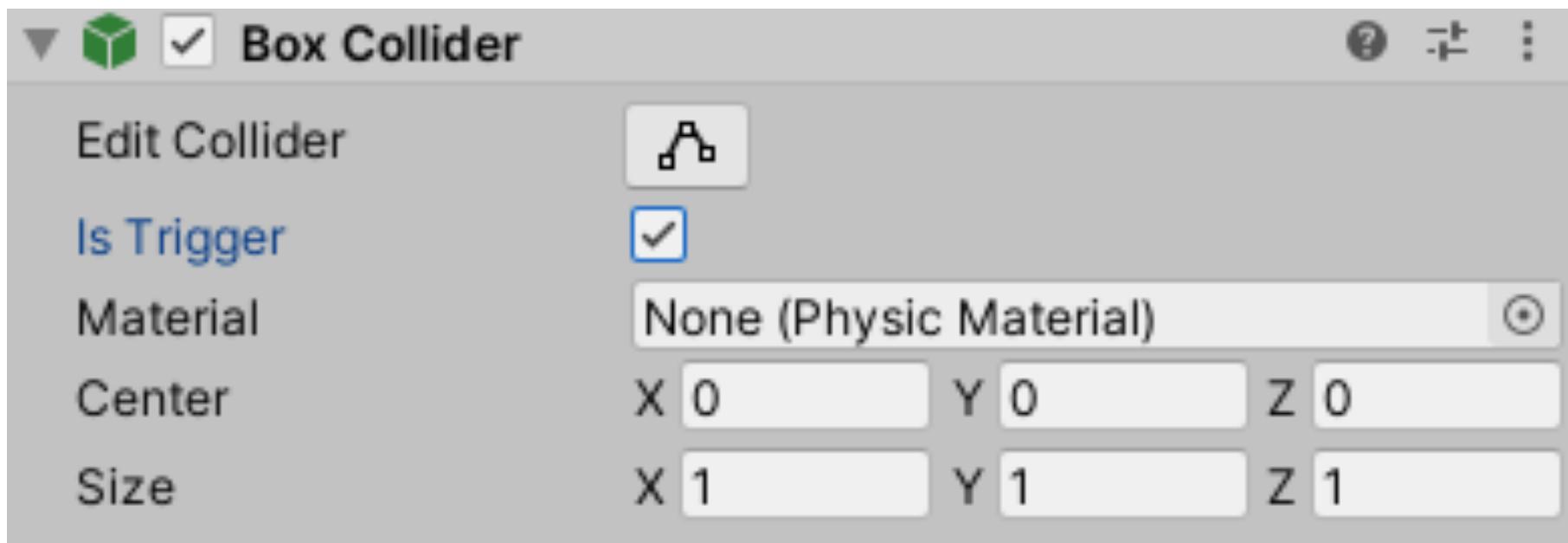
3. *Spiel mit Form, Position, Grösse*

4. Szenenwechsel

5. Build & Veröffentlichen

Szenenwechsel

1. Neues GameObject mit Collider (z.B. Box Collider) erstellen
2. In der Collider Komponente “Is Trigger” aktivieren



Szenenwechsel

1. Neues GameObject mit Collider (z.B. Box Collider) erstellen

2. "Is Trigger" aktivieren

3. Neues Skript anhängen

```
1  using UnityEngine;
2  using UnityEngine.SceneManagement;
3
4  public class ChangeSceneOnTrigger : MonoBehaviour
5  {
6      public string SceneToChangeTo;
7
8      private void OnTriggerEnter(Collider other)
9      {
10         if (other.CompareTag("Player"))
11         {
12             SceneManager.LoadScene(SceneToChangeTo);
13         }
14     }
15 }
16 }
```

Szenenwechsel

1. Neues GameObject mit Collider (z.B. Box Collider) erstellen
2. “Is Trigger” aktivieren
3. Neues Skript anhängen
4. Alle gewünschten Szenen in den Build Settings (*File > Build Settings...*) hinzufügen

Aufbau

1. Modell(e) importieren

2. First Person Controller

3. *Spiel mit Form, Position, Grösse*

4. Szenenwechsel

5. Build & Veröffentlichen

Build

1. Im Unity-Hub unter *Installs* für gewünschte version “Add Modules” auswählen und gewünschte Module hinzufügen
2. In den Build Settings die Platform wählen
3. In den Build Settings und Player Settings allenfalls Anpassungen machen (z.B. Icon hinzufügen)
4. Teilen!

WebGL

Einfach zu teilen

☞ Spieler muss nichts herunterladen, wenig(er) Kompatibilitätsprobleme

Auf Performance achten! Unterstützt keine Mobile-Browser

Veröffentlichen, z.B. auf *itch.io*

macOS/Windows/Linux

Beste Performance

Auf Permissions achten! Auf einem anderen Gerät zu starten versuchen vor dem Teilen.

Kann man auch online als Download bereitstellen, z.B. auf *itch.io*

iOS

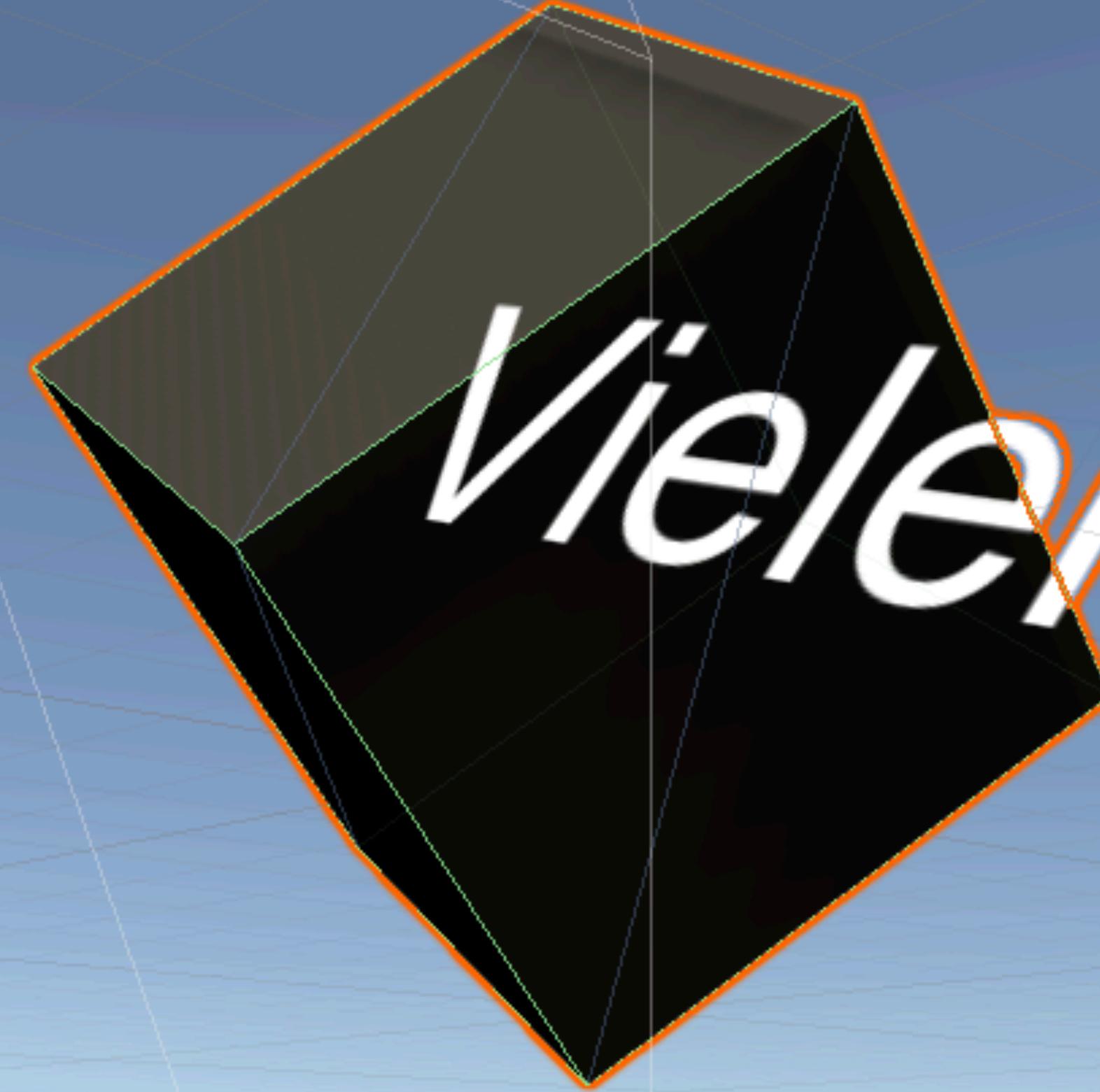
Builden exportiert ein .xcodeproj

In XCode lässt es sich (relativ) einfach auf dem eigenen Gerät testen

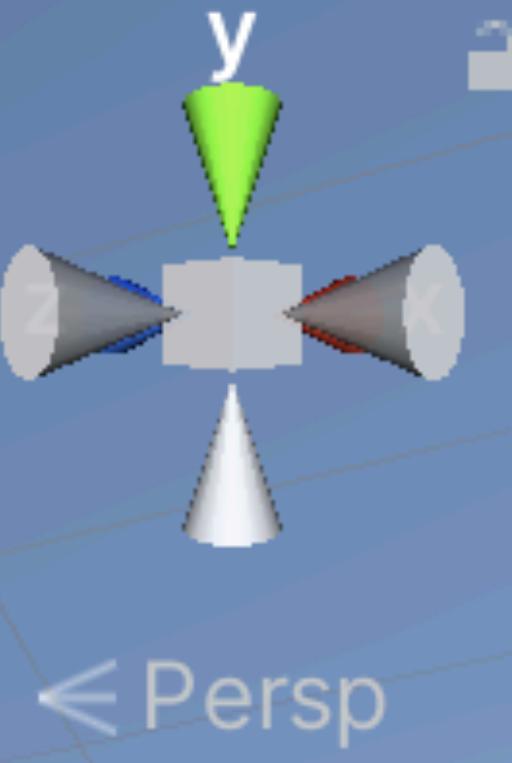
Zum Veröffentlichen muss es auf dem iOS App Store publiziert werden.

Dazu muss man im Apple Developer Program sein und die App muss den App Store Guidelines konform sein.

Keine Erfahrung mit Android — sorry! Sollte nicht all zu anders sein (und wahrscheinlich nicht schwieriger...)



Vielen Dank!



Slides und Scripts erhältlich unter <https://github.com/stekra/Unity-Intro-Packages>
Unity Fragerunde @ Montag, 22.11.2021
Fragen jederzeit an stekra@me.com (Antwortzeit nach Kapazität)