Einführung in die Informatik für Games Engineering

Tutorials

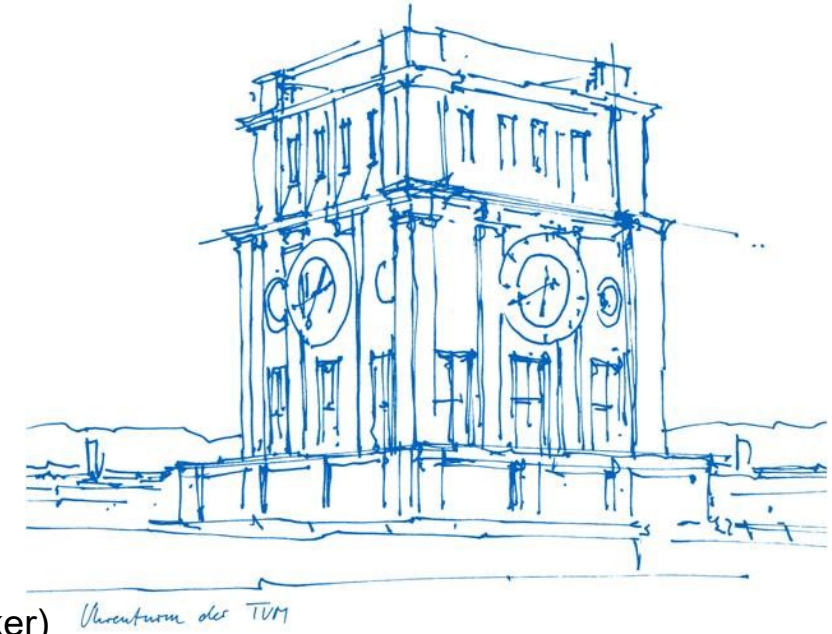
Gudrun Klinker (klinker@in.tum.de

Sven Liedtke (sven.liedtke@cit.tum.de)

Technical University of Munich

School of Computation, Information and Technology

Associate Professorship of Augmented Reality (Prof. Klinker)



What do you remember from last week?

What do you remember from last week?

- Local vs. World Space
- Differences Trigger / Collision
- Using Debug.Log

Lessons Learned – Issues

-> for Viewport you have to generate floats (not int) `Random.Range(0.0f,1.0f)`

Lessons Learned – Scopes

```
public class Projectile : MonoBehaviour
{
    [SerializeField] private float speed = 10;

    private void Update()
    {
        Vector3 move = Vector3.up * (Time.deltaTime * speed);
        transform.Translate(move);
    }

    private void OnTriggerEnter(Collider other)
    {
        Enemy enemy = other.GetComponent<Enemy>();
        if (enemy != null)
        {
            enemy.SetSpeedAndPosition();
            string message = "We hit: " + other.name;
            Debug.Log(message);
        }
    }
}
```

Let's revisit the basics of C# programming.

variables:	Storage for a value of a defined data type (eg. `float`)
scopes:	Mostly visualized by `{}`. Scopes define the lifetime of variables. Variables created inside a scope don't exist outside this scope.
method:	Sequence of operations that is run step by step if the method is called. Characteristic for methods is the `()` after its `MethodName`. Methods are linked to a class, so it can only be defined inside a class scope.
global variable:	Variable that is declared inside a class scope. It can be set to `public` instead of `private` to make it also visible outside of the instance.
local variable:	Variable that is declared in an inner scope inside of a class scope (eg. method scope).

Lessons Learned – Common Mistakes

```
public class Projectile : MonoBehaviour
{
    private void Update()
    {
        Vector3 move = Vector3.up * (Time.deltaTime * speed);
        transform.Translate(move);
    }
}
```

The variable `speed` hasn't been declared anywhere. Either create a local variable `float speed` in the `Update` method/scope, or declare a global variable `public float speed` in the class scope (<- better).

```
public class Projectile : MonoBehaviour
{
    public float projectileSpeed;

    private void Update()
    {
        Vector3 move = Vector3.up * (Time.deltaTime * speed);
        transform.Translate(move);
    }
}
```

Be cautious when copy&pasting code! Use the names that are used in your Project!

Lessons Learned – Common Mistakes

```
public class Projectile : MonoBehaviour
{
    [SerializeField] private float speed = 10;

    private void Update()
    {
        Vector3 move = Vector3.up * (Time.deltaTime * speed);
        Move();
    }

    private void Move()
    {
        transform.Translate(move);
    }

    private void OnTriggerEnter(Collider other)
    {
        Enemy enemy = other.GetComponent<Enemy>();
        if (enemy != null)
        {
            enemy.SetSpeedAndPosition();
            string message = "We hit: " + other.name;
        }
        Debug.Log(message);
    }
}
```

The variable **move** has been declared in the **Update** method/scope. So it doesn't exist in the **Move** method/scope and it can't be used there.

Same problem. The variable **message** has been declared in the **if** scope and doesn't exist outside of it.

Lessons Learned – Common Mistakes

```
public class Projectile : MonoBehaviour
{
    private void Update()
    {
        [SerializeField] private float speed = 10;
        transform.Translate(Vector3.up * (Time.deltaTime * speed));

        private void OnTriggerEnter(Collider other)
        {
        }
    }
}
```

Only global variables use the keywords **`private`**/**`public`**. And global variables can only be declared within a class scope.

Even though local functions do exist. We mainly define them inside a class scope (=> “method”). Unity events are part of the MonoBehaviour class and thus only be called if they’re defined in the class scope.

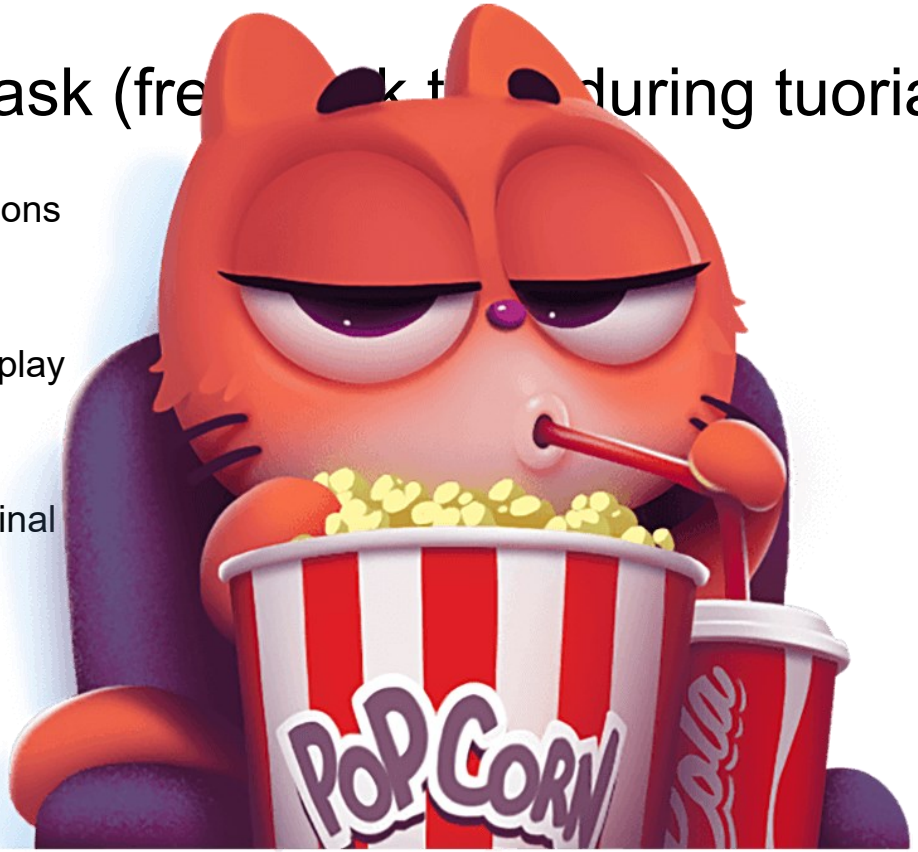
Hint: All these problems are highlighted in your IDE. The problem description and sometimes also their solution are shown if you hover over the highlights with your mouse. So make sure your IDE works!

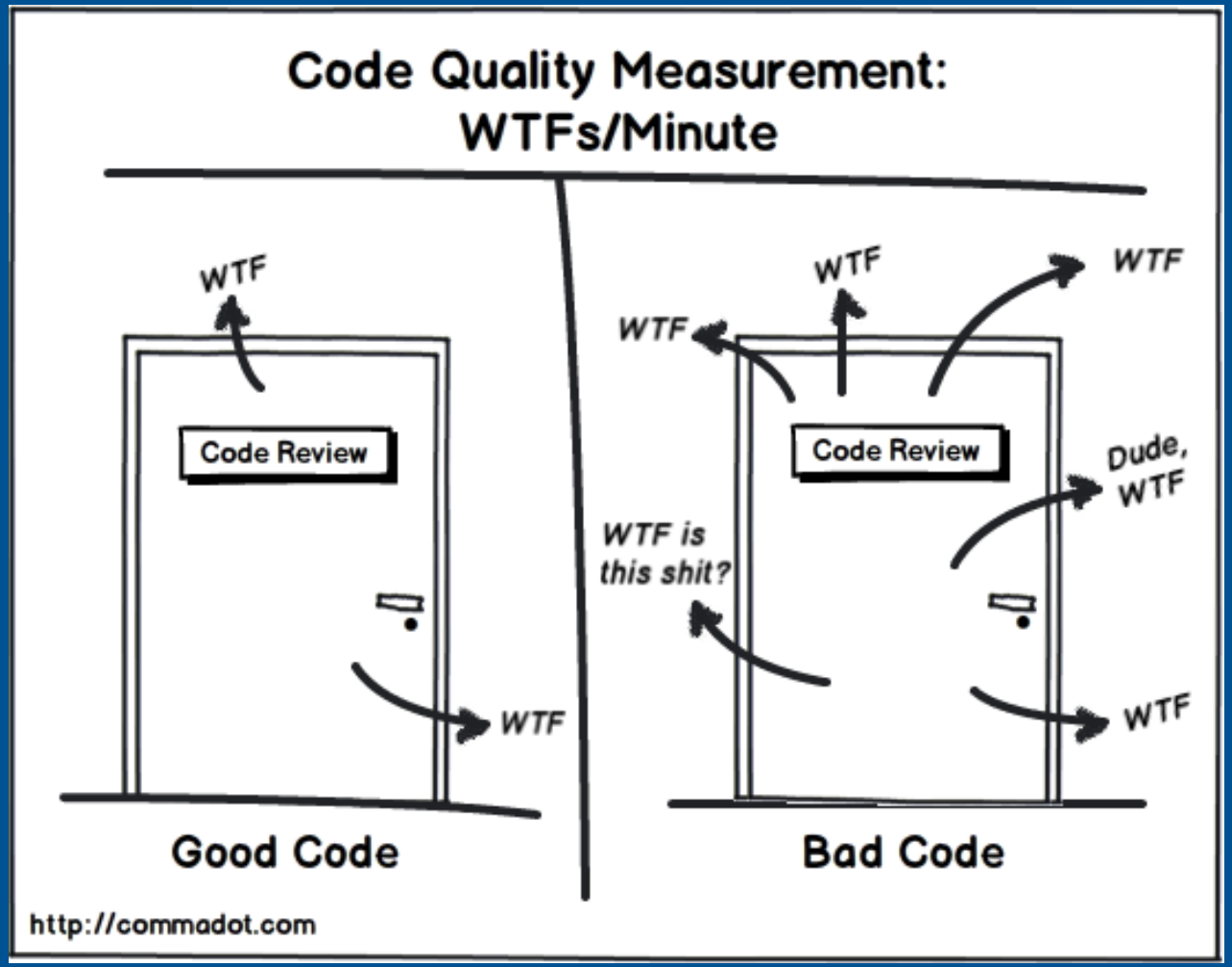
Additional Task (fre[illegible]k t[illegible]during tuorial or at home)

#1 Show your explosions

#2 How is your gameplay

Please present your final



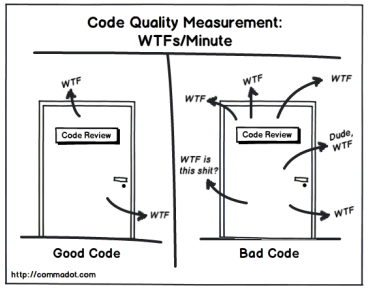

Code Quality Measurement:
WTFs/Minute
WTF
Code Review
WTF
Good Code
WTF
WTF
WTF
WTF
Code Review
Dude,
WTF
WTF is
this shit?
WTF
Bad Code
http://commadot.com

Topic: Debugging

To identify issues, bugs or not intended behavior you need to debug your applications

Therefore, **Debugging** is a core skill EVERY developer needs!

How did you solve your problems until now?



Topic: Debugging

Already known?

- Ask Friend
- Ask Tutor
- Use Google
- Use Software Documentation

- But still, you need to guess, what is your problem! Only doesn’t work does not help

Topic: Debugging

Identify Issues

- Debug.Log() -> Unity console (without unity there are also console)
  - Can print arbitrary content per frame (during runtime)
  - Informs you about issues in your scene setting (Multiple audio listeners)
  - Informs you about compile issues (before run)
  - Different error levels (Info | Warning | Error)
- Breakpoints with your IDE
  - You can use breakpoints to stop the current process
  - Step through code lines
  - Check values of variables

Topic: Debugging

Identify Issues

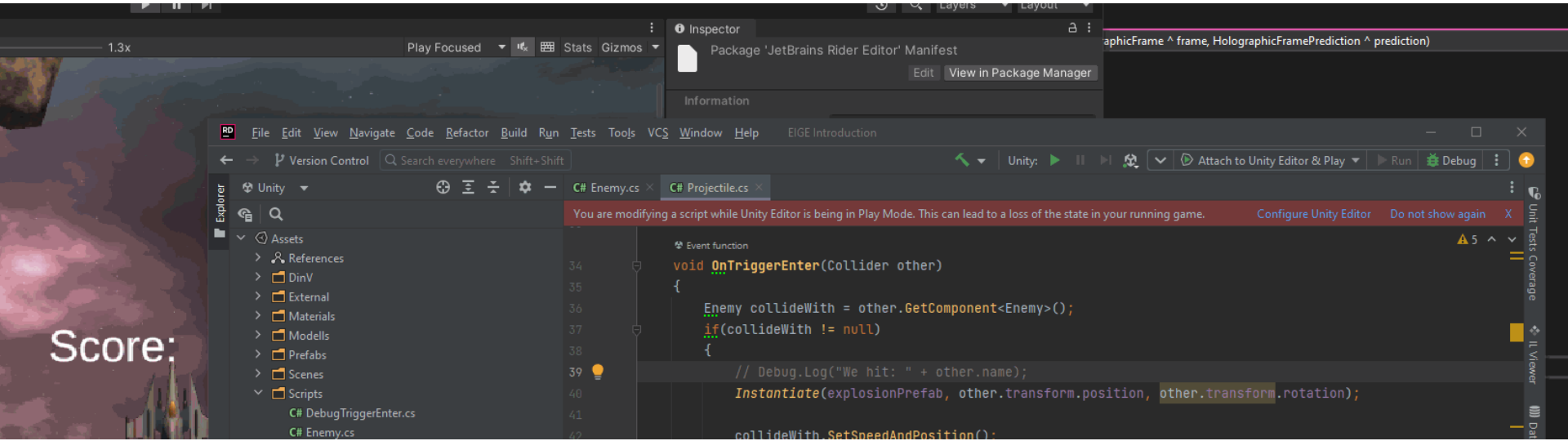
- Debug.Log() -> Unity console (without unity there are also console)
  - Can print arbitrary content per frame (during runtime)
  - Informs you about issues in your scene setting (Multiple audio listeners)
  - Informs you about compile issues (before run)
  - Different error levels (Info | Warning | E
- Breakpoints with your IDE
  - You can use br the current process
  - Ste
  - C of variables

Works with Java, C++, Shaders…

Topic: Debugging

Select a line to break, “attach to unity editor & play”, use debug to start “Play” in unity

- On break, you are able to directly read content of variables



Topic: Collision detection

Collision detection and physical resolving cost a lot of cpu usage of your game

To reduce the load to an optimal we can support the engines internal process

Collisions are detected not every frame (only during fixed update step)

- Also Physic calculation is done there
- During other frames current velocity is used to update transforms

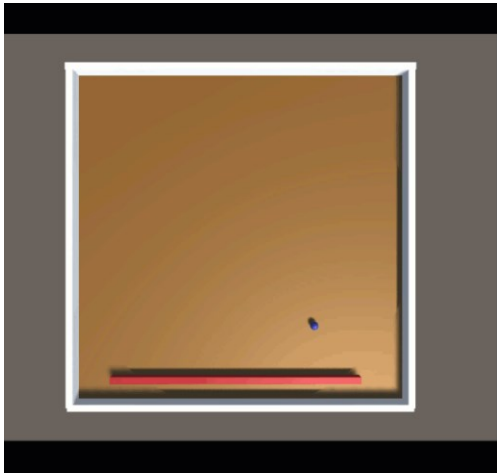
Topic: Collision detection

But what happened in between?

In general, there are only minor disadvantages with a sparse collision detection

- Maybe the contact point is not 100% accurate
- Maybe resulting forces are a bit off
- Maybe resolved movement is slightly off

- BUT: with really small and fast objects, maybe the collision isn't detected



Topic: Continuous collision detection (CCD)

https://docs.unity3d.com/Manual/ContinuousCollisionDetection.html

But what happened in between?

There are several methods to resolve these conflicts (Physics lecture 5th Semester).

For now:

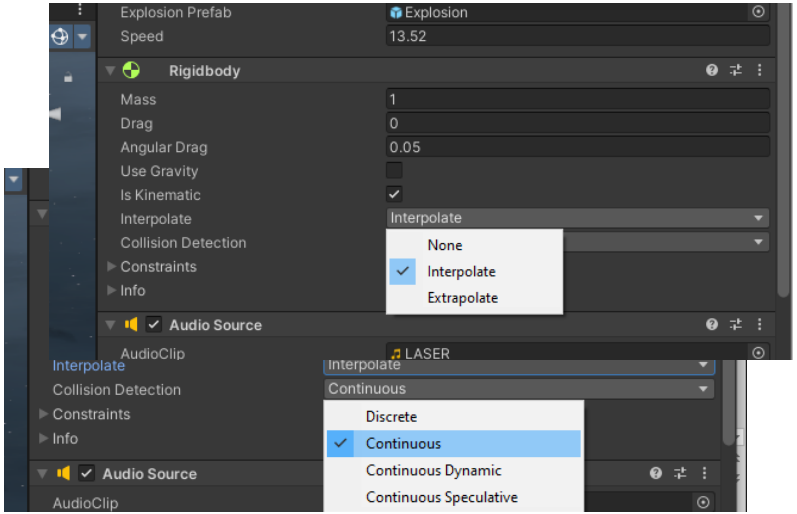
- Be aware of this problem
- Use Trigger in Situation where you don’t need physical behavior
- Void OnTriggerStay (just use Enter/Exit and store a local bool variable)
- reduce use of rigidbodies

Topic: Continuous collision detection (CCD)

https://docs.unity3d.com/Manual/ContinuousCollisionDetection.html

How to overcome discrete collision checks?

- Rigidbody can be adjusted to check for fast movement
  - ContinuousDynamic: fast moving object
  - Continuous: for objects a fast object should collide



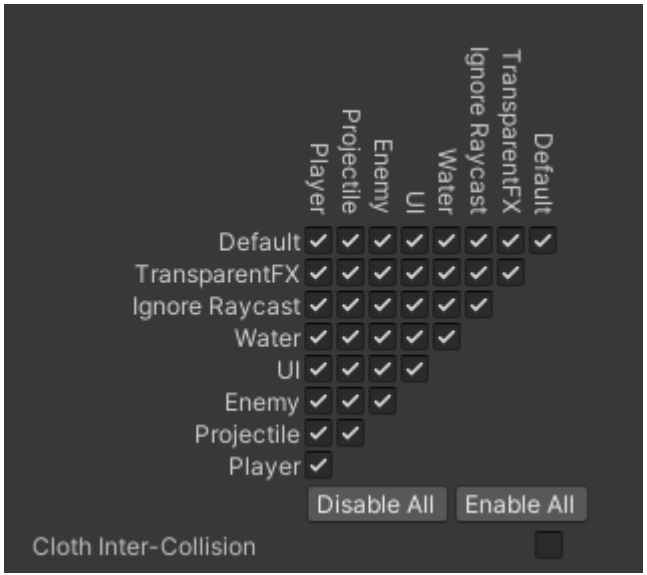
- Cheaper during physics calculation
  - For kinematic objects: ContinuousSpeculative

Does not work with mesh colliders (only primitives)

Topic: Collision matrix

https://docs.unity3d.com/Manual/LayerBasedCollision.html

Edit -> Project Settings, then select the **Physics** category to open the Physics window



Prevent collision between layer they should not interact with each other

Reduce false positive triggers of collision during physic step

Increase game play experience, e.g.

- Player in a multiplayer game shouldn't block each other
- projectile is should only interact with enemies and not with other projectiles

Making explosion to work

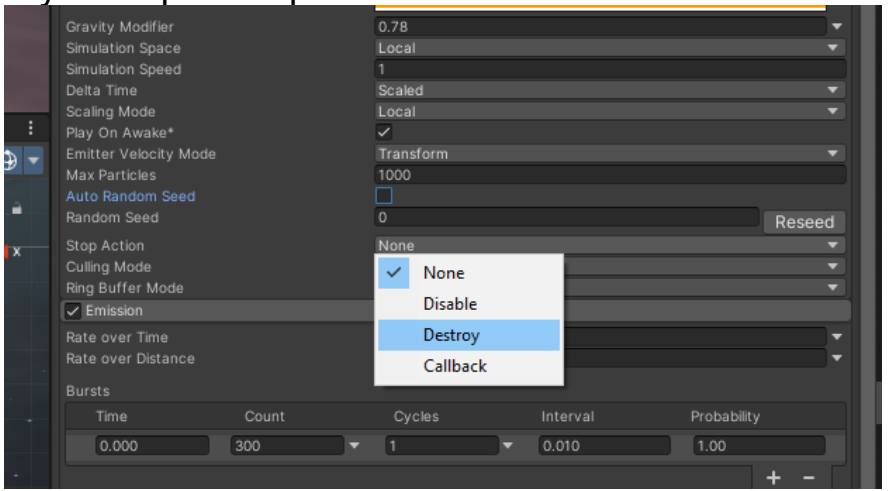
#1 Use your explosion as new prefab

#2 Create a new public variable in your projectile class for your explosion prefab

```
public GameObject explosionPrefab;
```

#3 Instantiate your explosion if your projectile hits an enemy and gets destroyed

- make your explosion destroyed after ending



Topic: UI

- Create UI object in Hierarchy => Choose a premade UI element (Image and Text – TextMeshPro are the most common)
- For displaying the player lives and score, choose the Text – TextMeshPro
  - It automatically creates a canvas, to display the UI on, and an event system to interact with the UI (e.g., Pressing UI Buttons)
  - If TextMeshPro is not imported, Unity will ask you to import them => Import TMP Essentials
- The text gameobject already got a text component which controls the text, a special renderer to show it on the canvas, and a rect transform (instead of regular transform)

Topic: UI

- Rect transform controls position on the canvas, the width and height of the displayed object, scale and achor points
- Anchor points can be set relative to parent objects or the canvas via alignments (elements can also be stretched there)

Topic: UI

- The canvas component on the generated canvas object has 3 different render modes
  - Screen Space – Overlay puts the canvas (and all it's objects) onto a specific region of the screen
  - Screen Space – Camera is similar to overlay, but attaches to a specific camera
  - World Space places the Canvas directly in the world, independent of the viewpoint

- Scaling of the canvas can be weird (it will appear huge in your scene). The recommended scaling should be looked up in the unity docs for every use case.
- The Canvas Scaler component scales the UI for different screen sizes. It offers Constant Pixel Size, Constant Physical Size or Scaling with Height/Width. What's best for your game needs to be decided individually for every game.
- It can be hard to make a clean UI with self made assets (but don't worry, everybody struggles with this in Unity). https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/HOWTO-UIMultiResolution.html

Display Lives and Scores

Now we prepare your script to display current score on the screen

Create a new public variable in your player script (editable via editor), use unity to link with your score label

```
public TMPro.TextMeshProUGUI scoreUI;
```

You can change the text form your label using `scoreUI.text = "Score: ”` …

#1 Display current score and lives on your screen

Multiple scenes

Now let's create some more scene. Save the current one and create a new scene (save as “main menu”)

- You can use an empty scene and add just a single camera, change to orthographic projection

#1 Add a quad within the viewport of your camera, add start texture (from asset pack to that quad)

#2 Save your scene as Main Menu

#3 Repeat it for a loose and win scene

Multiple scenes

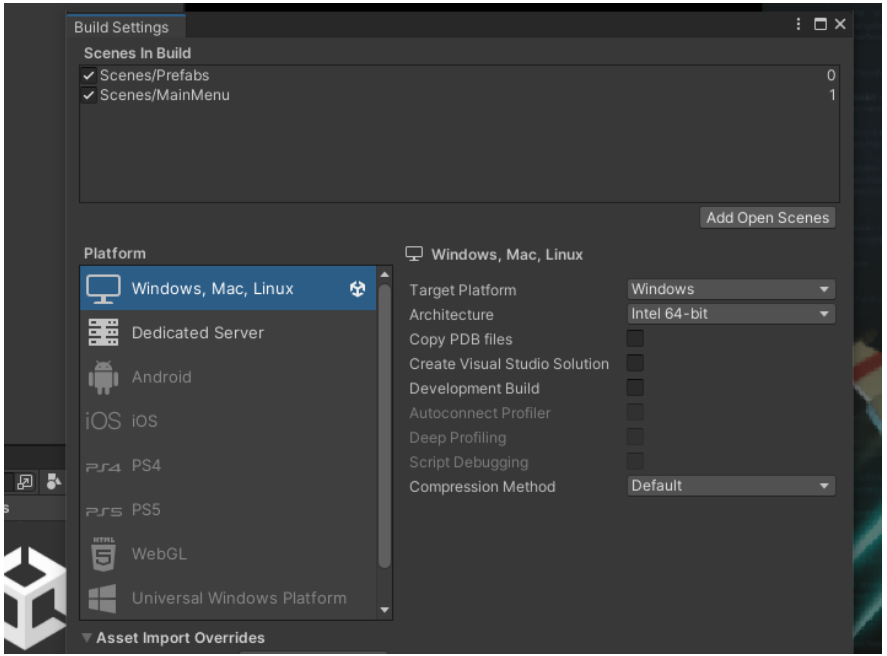
Open build settings (File -> Build Setting…)

Drag and drop all your scene to the scenes that should include in your final build (scene 0 is used as start scene)
- MainMenu should be 0 (your game scene 1)

You change the current scene using

void **LoadScene**(int **sceneBuildIndex**);

Maybe `if(Input.anyKey)` can help you



Additional Task (free work time during tutorial or at home)

#1 Use (our) textures to make the win/loose scene more interesting

#2 Build your game with calling the main menu as first scene

- Try it!

#3 Reduce your live if an asteroid hits your spaceship