Einführung in die Informatik für Games Engineering

Tutorials

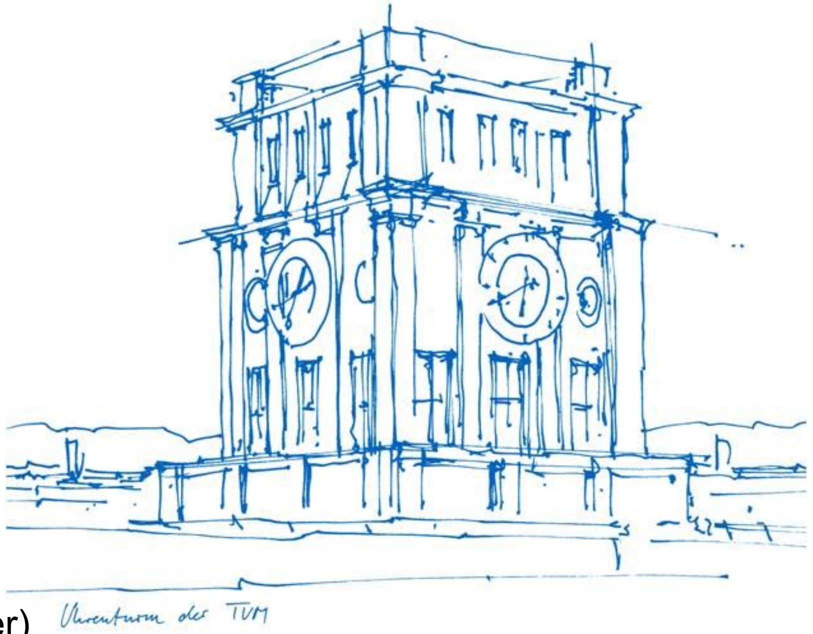
Gudrun Klinker (klinker@in.tum.de

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?

- Move objects by user input
- Unity functions for scripts
- Using a IDE

Additional Task (fre   during tutorial or at home)

#1 Adjust projectile spawn positi

#2 Adding Sound

Please present your final

Lessons learned – so far

- Bring your own Mouse! Helps to work with Unity a lot!
- Use `KeyCode.Space` instead of “`space`” for GetKey functions
- Variables shared between functions or with Unity needs to be in class scope e.g. class { public float test;}
  But other class may be able to change these values (bad programming style)

Better: Use private float test; and write getter or setter functions to allow access
To allow editing via inspector use SerializeField

```
[SerializeField]
private float speed;
```

Lessons learned – so far

- **Transform** contains information about current position, rotation, scale
- You can change position by overwrite with =, change with += or use .Translate() function

A transform component has a `position` parameter and also a `localPosition`, what can be the difference?

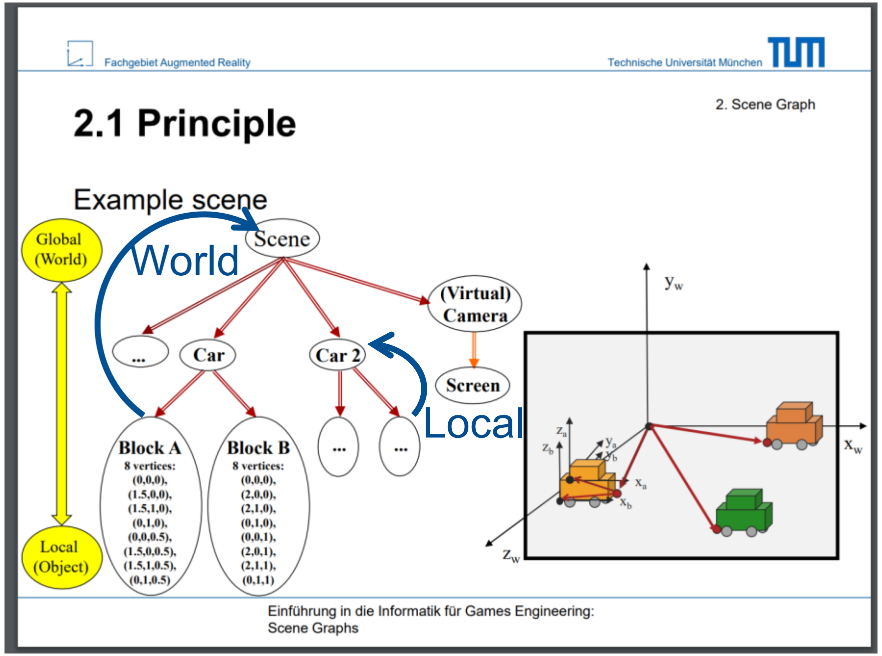
World Space vs. Local Space

Local Space

Gives the current position relative to the direct parent the game object, thus the previous node in the scene graph

World Space

Get or set the position to the world space a.k.a. to the root of your scene graph


Fachgebiet Augmented Reality
Technische Universität München

2. Scene Graph

2.1 Principle

Example scene

Einführung in die Informatik für Games Engineering:
Scene Graphs

Fighting against Asteroids

With the Projectile you will be able to shoot at the asteroids. We will now create these enemies – and add some action to your game.

- Use spheres or the asteroids from moodle asset package
  - Adjust size to a nice value, like 1.3
  - Create a nice material called m_enemy in your Material folder, use a red color or the provided textures
  - Create a new enemy script in your folder

Fighting against Asteroids

#1 Your asteroid should spawn with random speed add two class member variable called minSpeed maxSpeed of type float

both variables should be visible in the inspector

#2 Create a new function inside your enemy class

```
void SetSpeedAndPosition() { }
```

Use `Random.Range(min, max);` to get a random speed value

This function can be used from `Start()` and also from `Update()` without producing duplicate code, functions can be reused.

`Private` variables can be access by functions of your class, but not from other classes.

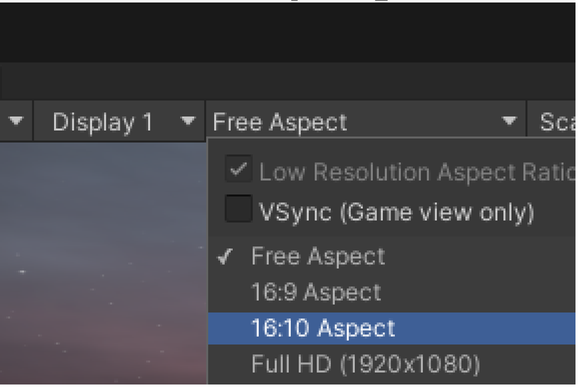
Associate Professorship of Augmented Reality
School of Computation, Information and Technology
Technical University of Munich

Fighting against Asteroids

#3 Move your asteroid downwards (similar to projectiles)

#4 In case the asteroids leave the screen call SetPositionAndSpeed
Generate new position also randomly

You can set your game view to a fixed size or get the world space position of your top camera border



Viewport of the camera is always between 0 and 1

```
Camera.main.ViewportToWorldPoint( new
Vector3(Random.Range(0f,1f), 1f, 0f));
```

Projectile Collisions: Colliders and Triggers

If you want GameObjects to react to each other you need to detect the collision between them: You need to edit the script of your Projectile, but before that some changes in the GameObjects components are necessary to make the collision detection work.

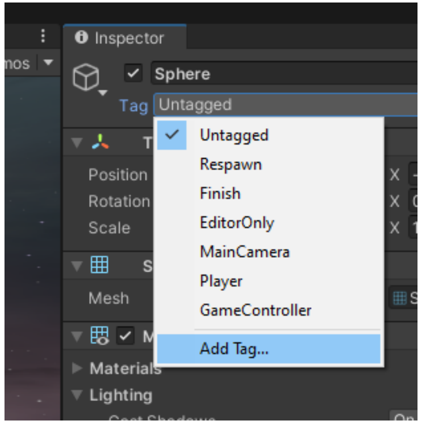
- Check if both objects, your projectile and the asteroid, have good colliders
- Use “IsTrigger” on the enemy’s collider
  - Trigger fire the OnTrigger event for collisions, but **without** complex calculation of speed, intersection point; physical interaction on other objects are disabled (e.g. object will not bounce but pass through the other collider
- The projectile should has IsKinematic checked (add a rigidbody if there is none)

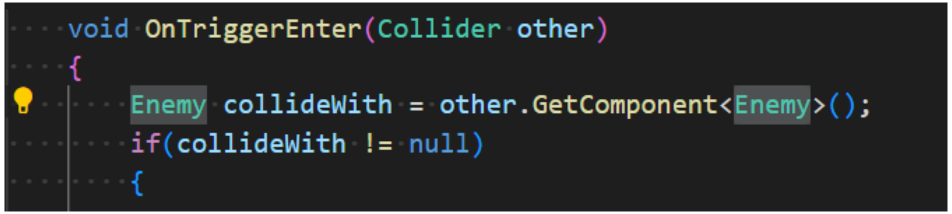
Projectile Collisions: Colliders and Triggers

To identify the other object during an OnTrigger event, you can use

- Tags
  - Use the inspector to give your objects a meaningful tag, e.g. Enemy

- `GetComponent<Class>()`
  - Check if the other object has the right script as component




```
void OnTriggerEnter(Collider other)
{
    Enemy collideWith = other.GetComponent<Enemy>();
    if(collideWith != null)
    {
```

Associate Professorship of Augmented Reality
School of Computation, Information and Technology
Technical University of Munich

Projectile Collisions: Colliders and Triggers

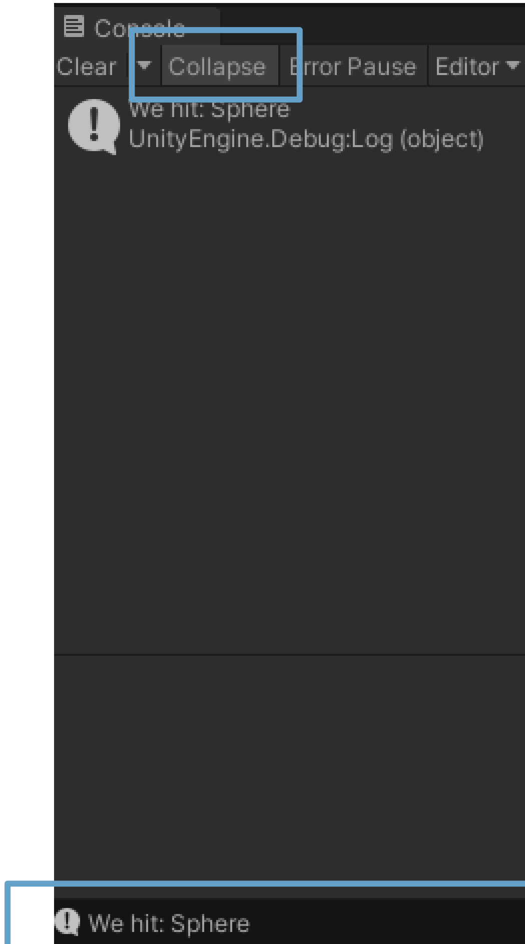
OnTrigger can detect different events
…Enter | …Exit | ..Stay
https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnTriggerEnter.html

Add to your Projectile script `void OnTriggerEnter(Collider other)` to get notified if a collision with an IsTrigger object happens.

Check if you collide with an enemy object and print it to Unity's console

```
Debug.Log("We hit: " + other.name);
```

http://docs.unity3d.com/ScriptReference/Debug.Log.html



```
We hit: Sphere
UnityEngine.Debug:Log (object)
```

```
We hit: Sphere
```

Projectile Collisions: Colliders and Triggers

If the projectile hit the asteroid, the enemy should vanish and a new one enters

How to solve this?

Projectile Collisions: Colliders and Triggers

If the projectile hit the asteroid, the enemy should vanish and a new one enters

Destroy and create new?

- Produces a lot of overhead in Unity’s frame loop to clean old stuff and allocate new memory

Better:

- Reuse existing object
  - Call `SetSpeedAndPosition` of the enemy from projectile script (e.g. with GetComponent
  - Either use `GetComponent("Enemy")` or with generic function call `gameObject.GetComponent<Enemy>()` <- better in performance

Now you can call SetSpeedAndPosition

https://docs.unity3d.com/ScriptReference/Component.GetComponent.html

Score and Lives: Static Variables

We will manage the game's score and the Player's lives in the Player script

For destroyed asteroid the player should get points, if hit by an enemy they loose one life

Use static variables for life and score counter

- Static variables are equal for all instances of a class, you don’t need to get a specific instance for editing

```
public static int score = 0;
public static int lives = 3;
```

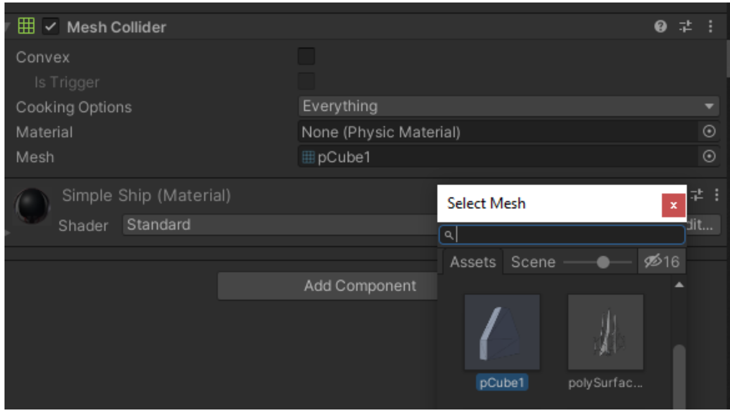
Score and Lives: Static Variables

#1 Increase points by 10 if a asteroid gets destroyed by projectile

- Call `Player.score += 10;`
- Print current score to console

#2 Decrease lives if asteroid hits your spaceship

- Use the simpflified spaceship as collision mesh
- Detect with OnTriggerEnter a collision with an enemy
- Reduce lives
- Print lives to console



Mesh Collider
Convex
Is Trigger
Cooking Options: Everything
Material: None (Physic Material)
Mesh: pCube1
Simple Ship (Material)
Shader: Standard

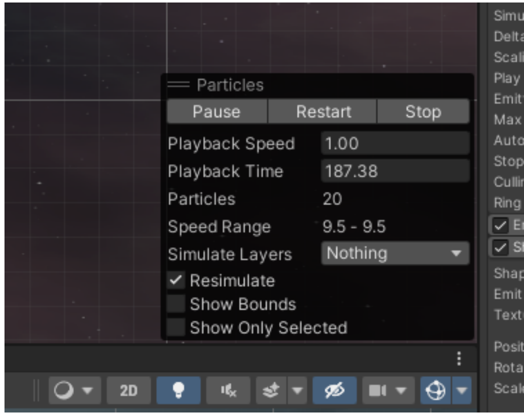
Unity's Particle System: Star background

To give the player the impression to fly fast through space we need some motion. To make it easier we don’t move the spaceship but the background.

With a particle system as background, we create the effect of motion without any additional scripting.

Place your particle system above the camera view port.
Check out the control panel to simulate particles during editing

Adjust your particle system:
- Use Box (large x value) shape to emit, rotate game object so particles fly downwards
- Use small start size for particles


Particles

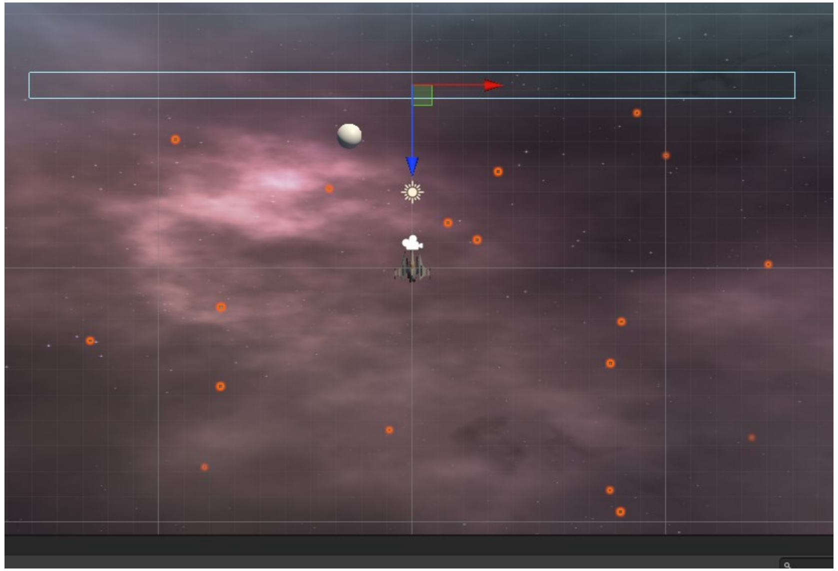
Pause | Restart | Stop

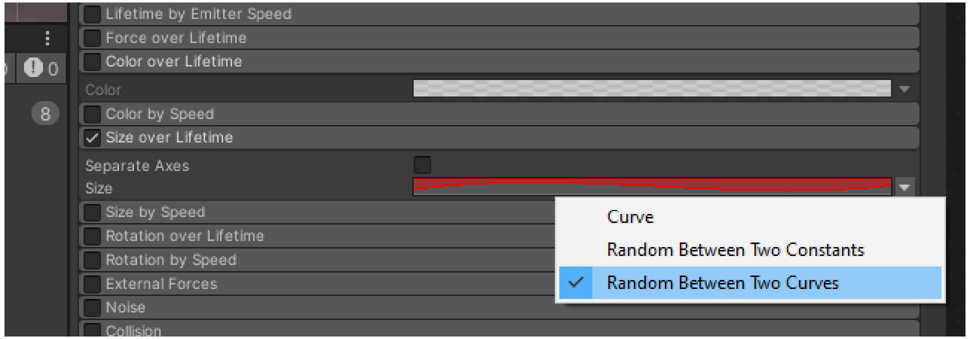
Playback Speed 1.00
Playback Time 187.38
Particles 20
Speed Range 9.5 - 9.5
Simulate Layers Nothing
✓ Resimulate
Show Bounds
Show Only Selected

Associate Professorship of Augmented Reality
School of Computation, Information and Technology
Technical University of Munich

Unity's Particle System: Star background

Try out different values, can change values also over lifetime (good for color change)
Or use random values… try it out





- [ ] Lifetime by Emitter Speed
- [ ] Force over Lifetime
- [ ] Color over Lifetime

Color

- [ ] Color by Speed
- [✓] Size over Lifetime

Separate Axes
Size

- [ ] Size by Speed
- [ ] Rotation over Lifetime
- [ ] Rotation by Speed
- [ ] External Forces
- [ ] Noise
- [ ] Collision

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

#1 Creating the Explosion Effect

- Set "Duration" to 0.5. "Duration" states how long the particle system will run.
- Uncheck "Looping". This will make sure the explosion only occurs once.
- Set "Start Lifetime" to "Random between two Curves" and make a curve that has a maximum at 0.4 (see image in previous slide). "Start Lifetime" determines how long a particle remains on screen after it is emitted.
- Set "Start Speed" to 3.
- Set "Start Size" to a similar Curve as "Start Lifetime" and set its maximum to 0.5.
- Set "Start Color" to "Random Between Two Colors"; choose red and yellow.
- In "Emission" set "Rate to 0. Make three bursts with 100, 100, 50 at 0.00, 0.03 and 0.06 (seconds). This will make sure the particles are only emitted at three points in time.
- In "Shape" set "Shape" to "Sphere" and its radius to 0.5.
- Also experiment with "over Lifetime" settings to make your particles more dynamic (for example you could make their size decrease over lifetime).

Further Reading: Of course you can manipulate and control particles via script. Unity's tutorial offers you fun with explosions (and scripting):
https://learn.unity.com/tutorial/introduction-to-particle-systems

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

#2 Make your spaceship fly vertical

Make movement also possible for y-axis, limit movement to about 2/3 (no wrap just stop forward movement)