Physic Engine Documentary

1. Create collision geometry in the Tiled map editor (2 points).
    - Assign collision geometry (AABB) to the level boundaries so that no object can leave.
    - Place at least four other collision geometries in the editor that will be loaded into the game as GameObjects with collision geometry and texture (e.g., planets)

Can be found in folder assets > Asteroids.tmx
5 Asteroids, Ships and all Colliders are parsed from the tilemap.

2. Load the collision geometry from the file and create appropriate components for game objects (3 points).
    - The goal is to initialize the physics of the objects completely from the input file.
    - Create a component for the rigid body (RigidBodyComponent), which holds the physical data (mass, velocity, …).
    - Create a collider component (BoxCollisionComponent) that holds the collision geometry (AABB) and the associated rigid body component.
    - A game object can have one rigid body but multiple collision components.
    - Add a simple boolean switch to the collider component that allows you to use it as a logic trigger, i.e., collisions are registered in the physics engine, but there is no physical response.

See RigidBody.h/cpp
RigidBodys is responsible for delegating collision events.
Implemented BoxCollider.h/cpp, including collision detection AABBvsAABB(), circlevsAABB()
Implemented CircleCollider.h/cpp, including collision detection circleVsCircle()
If a Collider is instanciated, a RigidBody component is also instanciated, but only once. See Collider.cpp onAdd()
Logic trigger to enable physics is called mHasPhysics. See Collider.h/cpp

3. Create a PhysicsManager (3 points).
    - The manager checks for collisions, resolves collisions, updates rigid bodies and notifies rigid bodies of collisions of the collision geometry associated with their GameObject.
    - Note again that triggers do not have a physical response, but notifications are still sent to rigid bodies
    - You can use the physics system of the exercise, or create one of your own.
    - When a collision occurs call an onCollision method of RigidBody so that the collision can be handled by observing components, e.g., by a point counting component, or player/asteroids.

See PhysicManager.h/cpp
PhysicManager is updated with 100 fps. Manifolds are created in findCollisions() and then resolved in resolveCollisions(), see PhysicManager::fixedUpdate()
The RigidBody component registers all components that inherit from IrigidBodyObserver that are attached to the same GameObject in the onStart().
RigidBodys can be Kinematic, effectively setting their inverse Mass to 0 (standing still).
RigidBodys and ColliderComponents register and unregister themselves to the PhysicsManager in the onAwake() / ~Destructor

4. Implement Asteroids gameplay using physics. Keep the physics simulation clean of gameplay code! Implement an observer pattern to connect components to rigid bodies so that the game play can react to physical collisions and triggers. (5 points)
    - Both player ships are controlled by applying physical forces for moving them, so that they are influenced by inertia.
    - Players shoot projectiles (Player 1: E, Player 2: O) into their forward direction. If a projectile hits an asteroid it is destroyed and a point counter is increased by 10 points for the player shooting the projectile. If a player hits another player, the hit player loses 10 point. The player is destroyed when hit by a projectile when the score is zero. You can just output the score in the console.
    - You can simply make destroyed asteroids invisible/inactive instead of actually deleting them. You can create a pool for projectiles when starting the game and do the same for projectiles to avoid dynamic creation of projectiles during the game. How you solve this is up to you.
    - All asteroids move in randomly chosen directions. You can set these directions when loading the game state. Inertia moves the asteroids forward.
    - When asteroids collide with each other the physics engine takes care of a correct physical response. Take care that the asteroids do not lose momentum, i.e., do not remove energy when collisions occur.
    - Players also bounce away from each other when they collide.
    - When player ships collide with asteroids they are destroyed and lose all points.
    - When players are destroyed they respawn

All points implemented.

Most handled in the onCollision() Methods in Projectile.h/cpp and Ship.h/cpp
Ships losing Momentum, Asteroids don’t.