Tile Engine Documentary

1. Create a tile map with at least two layers in Tiled (2 points)
    - Create a background layer (space, planets, etc.)
    - Create a foreground/top layer of non-gameplay relevant objects (satellites, moons, …)
    - Create occlusions so that the player objects are hidden behind geometry of foreground/top layer

Can be found in folder assets > Asteroids.tmx
Order (from back to front)

1. Background > Stars/Moons/Satellites
2. Player
3. Asteroids
4. Foreground > PlanetStatues

2. Load the tile map (see Exercise Tiled) and fill layers into LayerComponents. LayerComponent renders a single tile layer (2 points)
    - One instance of LayerComponent is responsible for rendering one layer.
    - Aggregate all LayerComponents into a GameObject, which represents the map GameObject

LayerComponent can be found in LayerRenderer.h/cpp
Every LayerComponenet is bound to a separate Gameobject and childed to the map GameObject - GameOjbect instantiation happens in ContentLoader.cpp (where the parsing from tmx file happens)

3. Implement a RenderManager class that renders sprites/tile layers in the correct order (4 points)
    - Implement an abstract RenderComponent interface to be used in the RenderManager. The manager does not know about the specific types behind RenderComponents and just uses this abstract class.
    - All renderable components are derived from RenderComponent, e.g., LayerComponent, SpriteRenderComponent, …

RenderComponent can be found in RenderComponent.h/cpp
RenderManager is implemented as Singleton and can be found in RenderManager.h/cpp

- In the RenderManager, each RenderComponent is assigned to a layer to have control over the render order and occlusions.
- The render order of layers is controlled using a custom attribute defined in the Tiled editor. For instance, note how each Tile Layer in Tiled has a name that you could use. You can define a render order by defining custom attributes of the map using the option Map Properties (accessible via Map -> Map Properties)

Every Tilelayer has a custom property called renderlayer and every Object in the Objectlayer has a property for spriterenderer and other components, also layer.
Asteroids.tmx
Components get added to the GameObject with their associated parsing constructor

- Rendering is performed by calling the draw method of the RenderManager, which calls draw the methods of the respective RenderComponent.

See draw() in game.cpp

4. Add objects to an object layer of the tile map in the Tiled Map Editor and load them into the game (4 points)
    - Use the existing object attributes (e.g., position, size, …) and your custom attributes (e.g., texture name for sprite sheet, rectangle, …) to initialize the GameObject and associated components in code.
    - Place player-controlled objects, i.e., the spaceships, in the editor.
    - Place 5 asteroids within the bounds of the playing field using the Tiled editor. Note that the code for loading asteroids should be the same for each one. Give them a color attribute in the editor and color them red.

Tiled Object Properties such as position, width, height, and name are parsed, and its values are copied to the game objects. Furthermore, Custom properties can be defined which enable you to create additional components for the game objects.
New functions for parsing custom properties are added in the constructor of the content loader, this is handled by generic function pointers. See ContentLoader.cpp
When adding new template components to the function map, the corresponding component must define a specific constructor. The constructor initialises the member variables.

- Place an additional 5 asteroids randomly within the bounds of the playing field. This cannot be done in the Tiled editor but should be done in code. The code can be part of the respective GameState, after loading from the Tiled editor file. Color them blue.

See Gamestate.h/cpp

- Asteroids can be simple geometry. Check out SFML shapes

See PrimitveRenderer.h/cpp

- The goal is to initialize game objects completely from the data of the object layer.
- For better readability and maintainability of the code encapsulate creation into helper functions, e.g., createPlayerObject, createEnemyObject, …

The only Object not created from parsing the tiled map, are the random asteroids in GameState.h/cpp with method createAsteroidObject()
Regarding encapsulation see ContentLoader.h/cpp Creator functions

- As for the LayerComponents, the loaded objects are rendered using the RenderManager.

See RenderManager.h/cpp

5. Think about when the RenderComponents should be assigned to the RenderManager.
    - Option A: The RenderManager could parse each game object after initialization of the GameObjects from the file.
    - Option B: Use a helper function to create a renderable component. The helper function creates the component, registers it at the RenderManager and returns the correct component for further processing.
    - Option C: Each RenderComponent registers itself to the RenderManager in the constructor. Physik + Observer
    - Other options? There are multiple solutions to this problem.

We went for Option C. See RenderComponent.h/cpp