Envision Tanks

How to Play

Controls

Up and down arrow for rotation.

Hold space to load a shot.

Weapons

TankRifle: straight shot, low dmg

BallisticRifle: parabel shot, high dmg

FakeTankRifle: angled shot, places fake tank that covers you, but you can shoot through

The Makings

Engine

The first thing that was to do in my opinion was to flash out the engine a little bit more since there was nothing in there. If I had simply started scripting the whole project would have become a mess.
The base of my whole concept is the GameObject system, though after I was done I did regret that I did not also make a Behaviour system ontop. The idea was that one Gameobject holds one Behaviour in itself (but I realized later that not every object needs a Behaviour), also it would not necessarily need a loaded graphic like an image so I excluded that. This I believe would also have been better if I had made a RenderComponent and assigned that to the necessary Gameobjects.
Okay but since I did it this way it works like the following. Every Gameobject has a Logic Update and a Graphics Update. Upon creation, they assign themselves to the engines GameObject List and unassign themselves automatically on delete. GameObject is abstract, allowing me to make classes deriving from it and still functioning as their own when called.
The next big part is ComponentSystem. I made a GameComponent class whose only purpose here is to unify components and making it possible to add them with one method using Generic type T.
These Components have their Systems updating them all the time and then they relay the update to the GameObject they are attached to. This way I solved physics and Collison. For Collision I also subdivided collider into static and nonstatic. This helps since static collider does not move and can not be the one causing a collision.
Finally, we have UISystem. This is just a the GameObejct system with less feature, that is being rendered beforehand and does not have a Behaviour update.

Most Important Classes

Tanks, Barrel, Weapon, Projectile

The tank holds a barrel and the barrel has a set of Weapons that can fire specific projectiles.
The tank gives commands to the barrel, that executes them using its own logic and then gives commands to the weapon, that then uses its own logic to shoot its specific projectile, that then is an independent object that holds the tag of the barrel. This way logic in every single object is kept to a minimum and responsibilities are well divided.

GameLogic

The tanks and barrels are made by the GameLogic, this acts as my sort of “scene”. It creates necessary objects and holds the logic to progress and end the game upon winning condition.

Terrain

The Terrain is being made at the start, creating a predefined path.
I kept this quite simple, first find a place for the tanks to stand, then connect these two platforms with a set number of points using a few simple rules to make sure the terrain does not block too much. After that, I simply connect the sides of the screen to the tank platforms in a straight line, since that terrain has no gameplay value anyway. Of course in a real game, this would not be okay since it can look very strange. Finally, I can add all the point to a path and create a collision for this. The path will then be drawn automatically every frame because I made the Terrain itself a GameObject (this could also help with making destructible terrain since I already have collision and the ability to react in place).

XML loading

I am not adept at loading this format and went for a straight forward implementation. I do believe JSON would have been the better choice to load weapons in my scenario. This is because weapons are their own class anyway, which means that after loading the JSON the class would directly be ready to use costing only 1 method with 2 lines of code.