AUTHOR	Michael Soler
CONTACT	michael.soler.beatty@gmail.com
Unity Ver.	2018.2

Index

1.Description of the package.

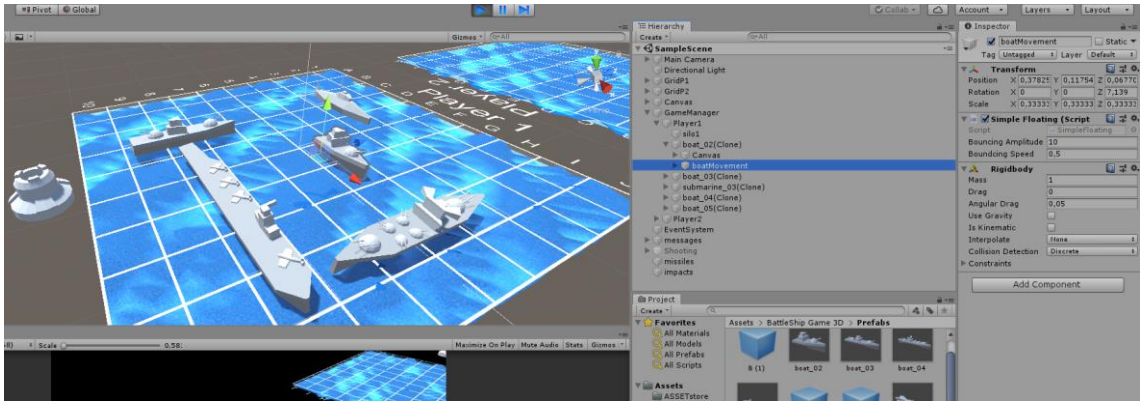
With this addictive package you will be able to recreate a futuristic battleship game in which the objective is to sink the enemy ships by launching several missiles to the unknown enemy zone. After each try the grid will show if you have missed or hit an objective. When all the ships are destroyed the game ends.

This package includes the following features:

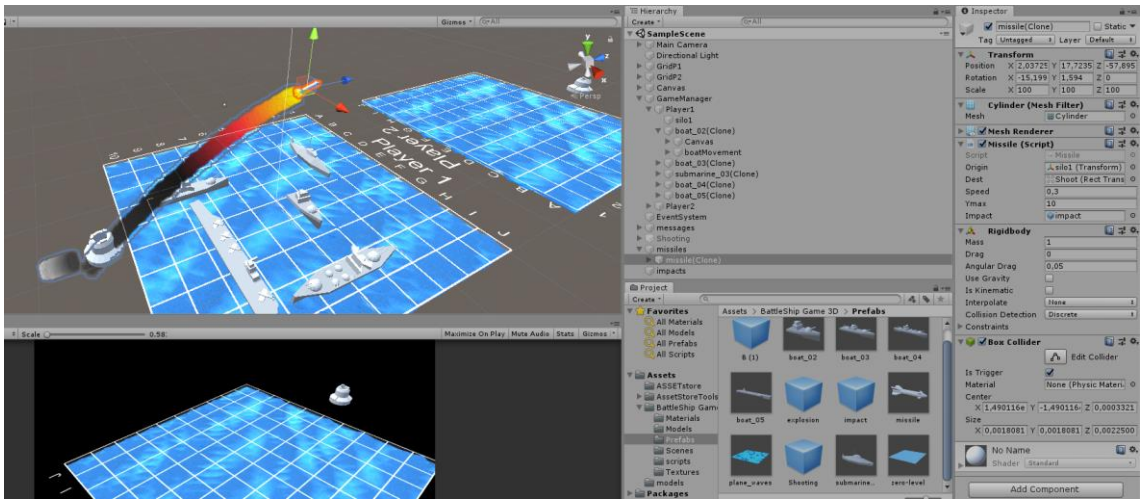
- 3D ship models.
- Camera movement, zoom and rotation.
- Game manager that shows several canvases with instructions.
- Missile generation, movement and management.
- Impact management in water and in ships with different markers.
- Transparent and friendly-code.
- Water simple dynamics.
- Boat floating effect.

2.Colliders, tags and physics

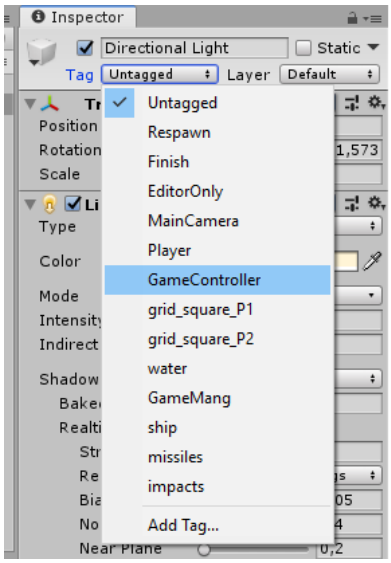
Rigidbodies are used for boat movement and trigger interactions when placing the ships and when launching missiles. Here we can see where the RB is attached to a boat.



Here we can see where is attached to the missile:



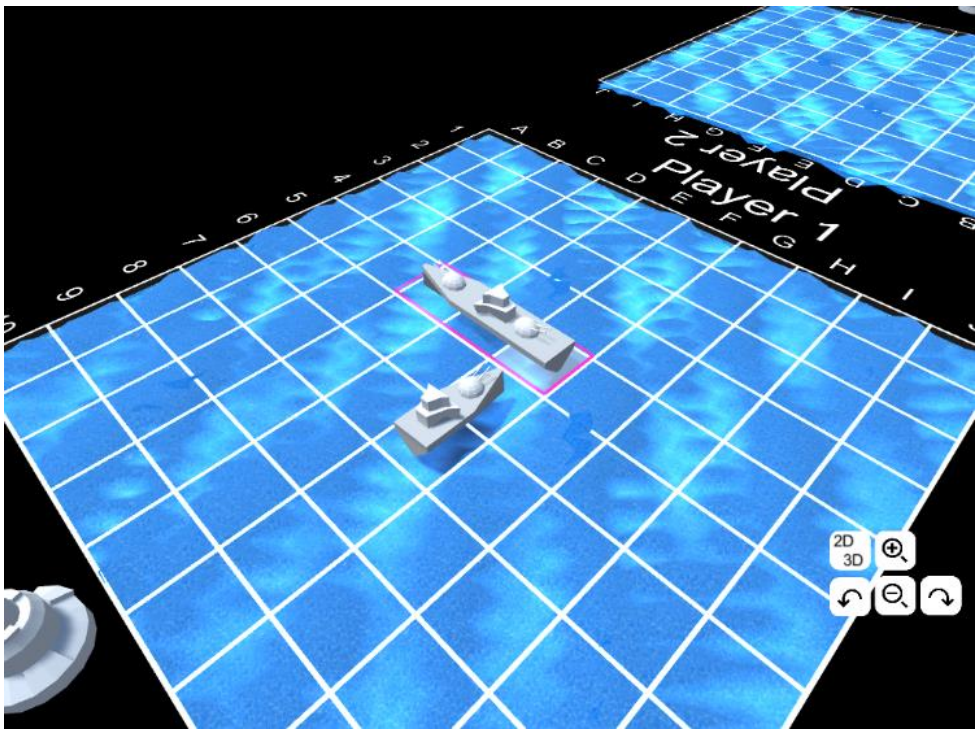
The following tags must appear in your project:



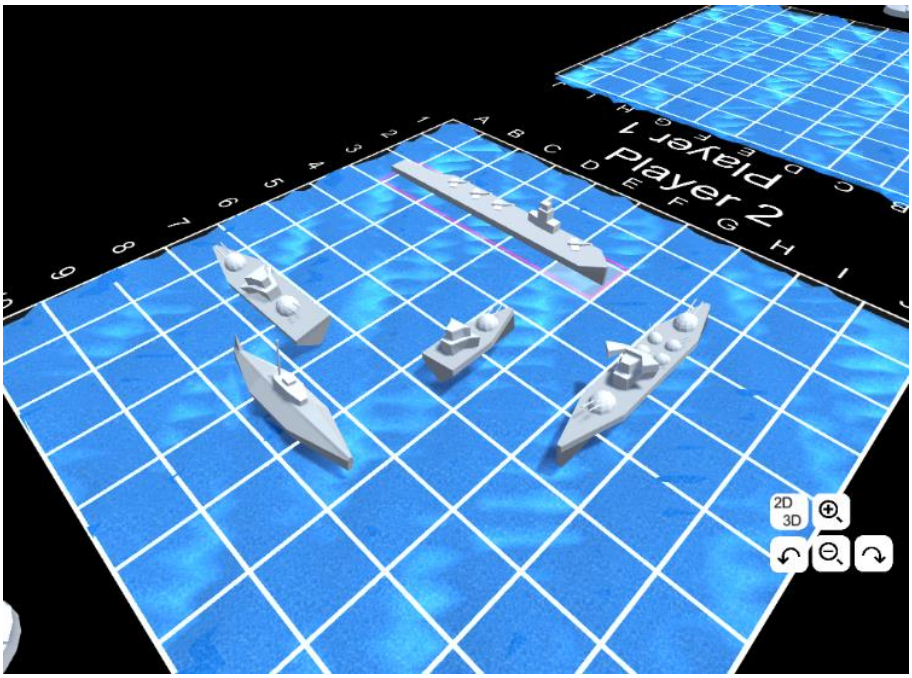
3. Game Evolution

The game is divided into different stages.

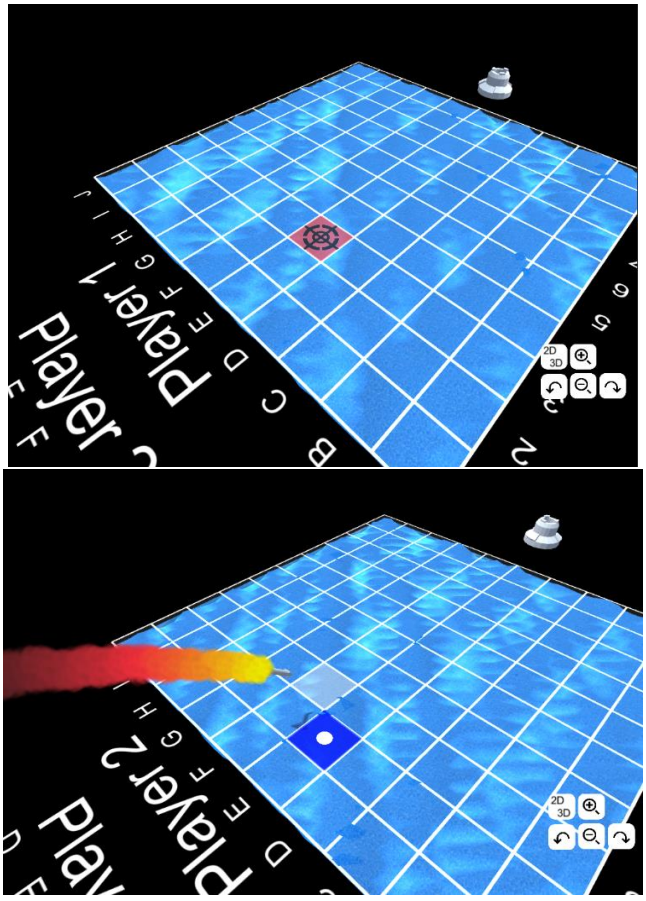
- **Player 1 places its boats.** Thanks to the colliders, we are able to know when the boats are misplaced or overlap other ships. Boats are placed by clicking on the grid and rotating the boat in the four possible directions. A magenta line will appear around the ship.



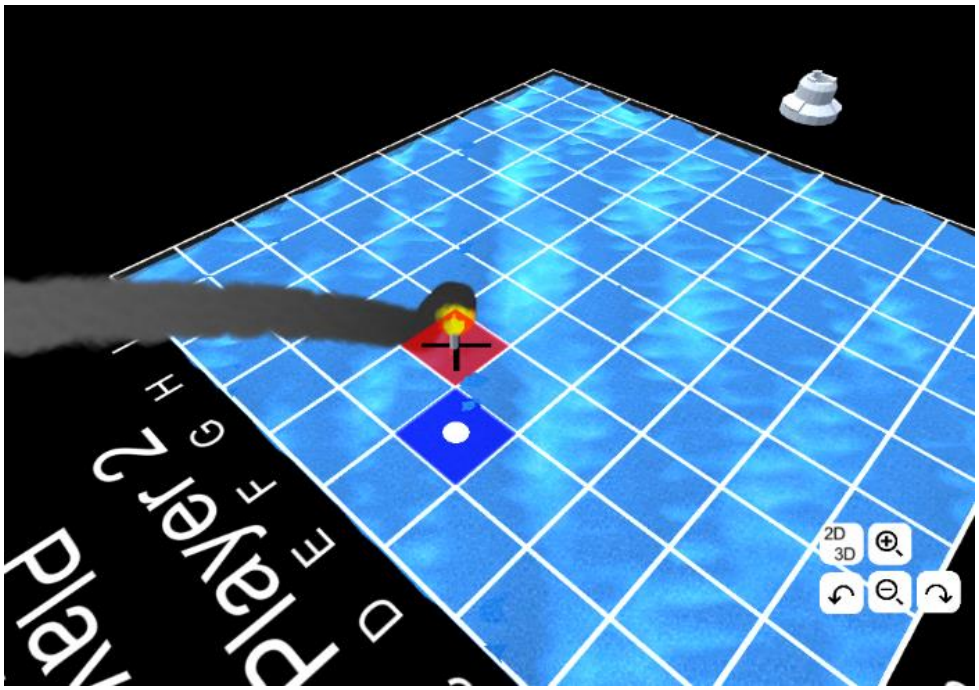
- **Player 2 places its boats.** Same process as shown before but this time it is player 2 who places the boats in the screen.



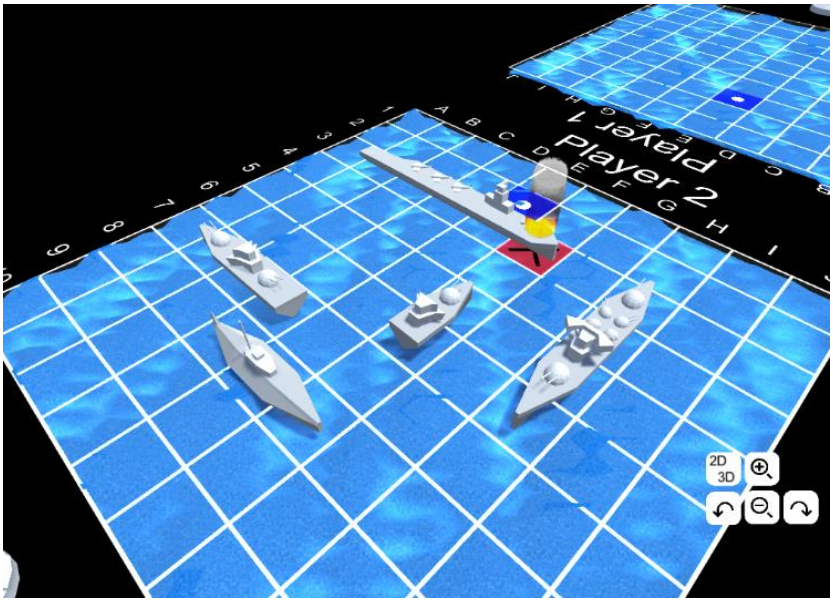
- **Player 1 turn.** The player can select a square from the enemy grid and it will show the “shooting button”. Once pressed, the missile will launch.



If the player hit the water a blue marker will stay over the grid, while a red marker will appear if the player hits an enemy boat.



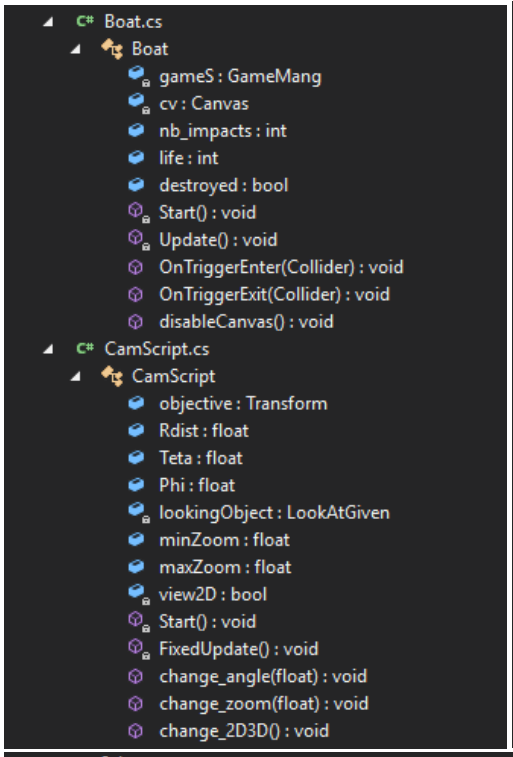
- **Player 2 turn.** The player can select a square from the enemy grid and it will show the "shooting button". Once pressed, the missile will launch. Same process than in player 1. In this case the player can check the state of his boats on the grid.

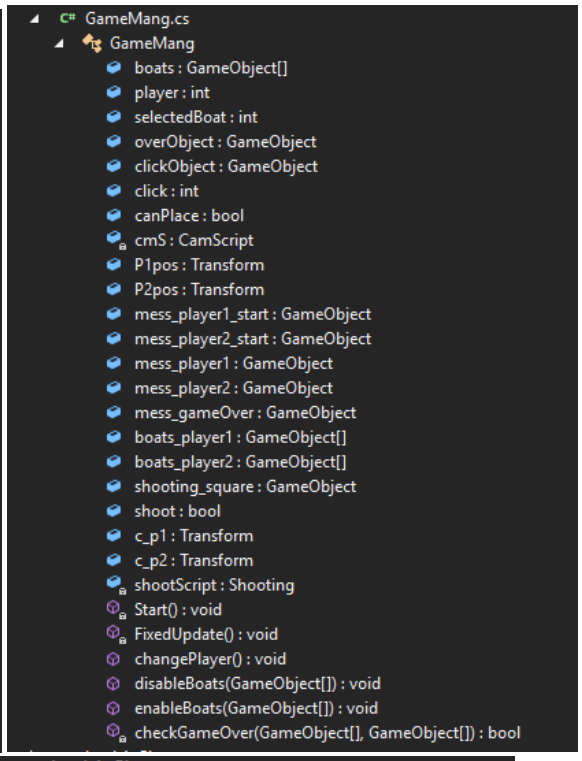


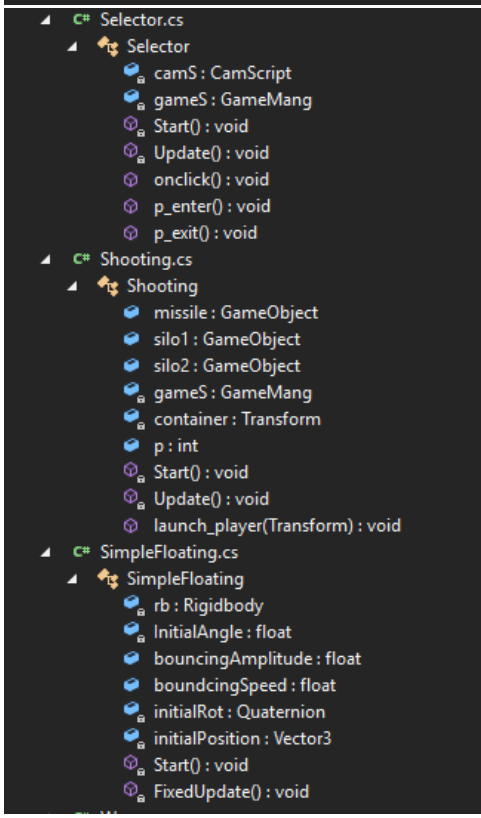
5. Scripts

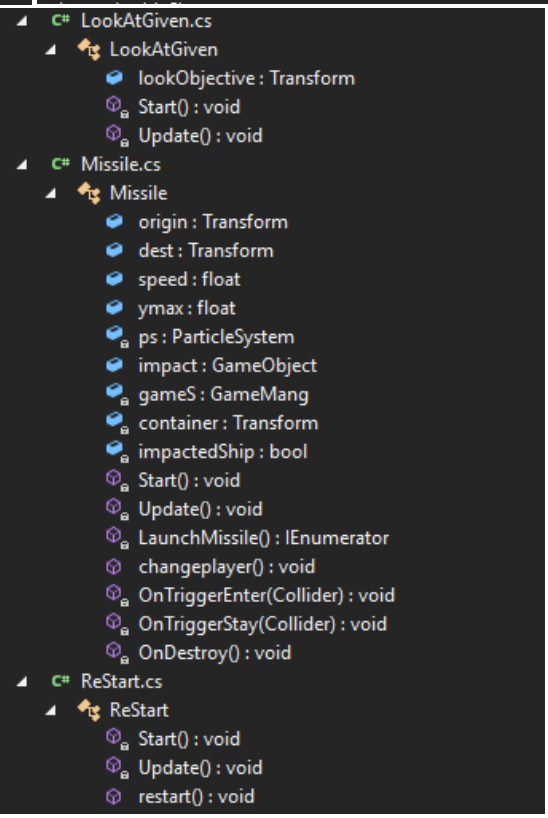
We have made a specific tutorial explaining how scripts and game dynamics work.

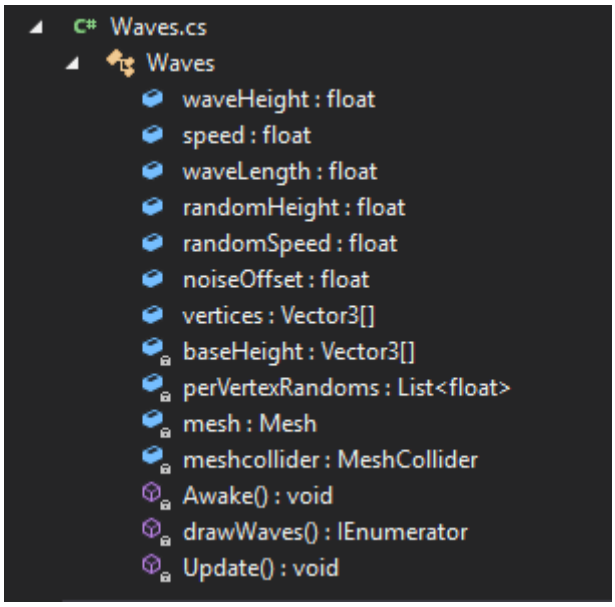
https://youtu.be/za5wvYbn7nM


Boat.cs
Boat
gameS : GameMang
cv : Canvas
nb_impacts : int
life : int
destroyed : bool
Start() : void
Update() : void
OnTriggerEnter(Collider) : void
OnTriggerExit(Collider) : void
disableCanvas() : void
CamScript.cs
CamScript
objective : Transform
Rdist : float
Teta : float
Phi : float
lookingObject : LookAtGiven
minZoom : float
maxZoom : float
view2D : bool
Start() : void
FixedUpdate() : void
change_angle(float) : void
change_zoom(float) : void
change_2D3D() : void


GameMang.cs
GameMang
boats : GameObject[]
player : int
selectedBoat : int
overObject : GameObject
clickObject : GameObject
click : int
canPlace : bool
cmS : CamScript
P1pos : Transform
P2pos : Transform
mess_player1_start : GameObject
mess_player2_start : GameObject
mess_player1 : GameObject
mess_player2 : GameObject
mess_gameOver : GameObject
boats_player1 : GameObject[]
boats_player2 : GameObject[]
shooting_square : GameObject
shoot : bool
c_p1 : Transform
c_p2 : Transform
shootScript : Shooting
Start() : void
FixedUpdate() : void
changePlayer() : void
disableBoats(GameObject[]) : void
enableBoats(GameObject[]) : void
checkGameOver(GameObject[], GameObject[]) : bool


Selector.cs
Selector
camS : CamScript
gameS : GameMang
Start() : void
Update() : void
onclick() : void
p_enter() : void
p_exit() : void
Shooting.cs
Shooting
missile : GameObject
silo1 : GameObject
silo2 : GameObject
gameS : GameMang
container : Transform
p : int
Start() : void
Update() : void
launch_player(Transform) : void
SimpleFloating.cs
SimpleFloating
rb : Rigidbody
InitialAngle : float
bouncingAmplitude : float
boundcingSpeed : float
initialRot : Quaternion
initialPosition : Vector3
Start() : void
FixedUpdate() : void


LookAtGiven.cs
LookAtGiven
lookObjective : Transform
Start() : void
Update() : void
Missile.cs
Missile
origin : Transform
dest : Transform
speed : float
ymax : float
ps : ParticleSystem
impact : GameObject
gameS : GameMang
container : Transform
impactedShip : bool
Start() : void
Update() : void
LaunchMissile() : IEnumerator
changeplayer() : void
OnTriggerEnter(Collider) : void
OnTriggerStay(Collider) : void
OnDestroy() : void
ReStart.cs
ReStart
Start() : void
Update() : void
restart() : void


Waves.cs
Waves
waveHeight : float
speed : float
waveLength : float
randomHeight : float
randomSpeed : float
noiseOffset : float
vertices : Vector3[]
baseHeight : Vector3[]
perVertexRandoms : List<float>
mesh : Mesh
meshcollider : MeshCollider
Awake() : void
drawWaves() : IEnumerator
Update() : void